

GraphEnet: Event-driven Human Pose Estimation with a Graph Neural Network

Gaurvi Goyal
Maastricht University &
Istituto Italiano di Tecnologia
gaurvi.goyal@
maastrichtuniversity.nl

Pham Cong Thuong
Istituto Italiano di Tecnologia
cong.pham@iit.it

Arren Glover
Istituto Italiano di Tecnologia
arren.glover@iit.it

Masayoshi Mizuno
Sony Interactive Entertainment Inc.
masayoshi.mizuno@sony.com

Chiara Bartolozzi
Istituto Italiano di Tecnologia
chiara.bartolozzi@iit.it

Abstract

Human Pose Estimation is a crucial module in human-machine interaction applications and, especially since the rise in deep learning technology, robust methods are available to consumers using RGB cameras and commercial GPUs. On the other hand, event-based cameras have gained popularity in the vision research community for their low latency and low energy advantages that make them ideal for applications where those resources are constrained like portable electronics and mobile robots. In this work we propose a Graph Neural Network, GraphEnet, that leverages the sparse nature of event camera output, with an intermediate line based event representation, to estimate 2D Human Pose of a single person at a high frequency. The architecture incorporates a novel offset vector learning paradigm with confidence based pooling to estimate the human pose. This is the first work that applies Graph Neural Networks to event data for Human Pose Estimation. The code is open-source at <https://github.com/event-driven-robotics/GraphEnet-NeVi-ICCV2025>.

1. INTRODUCTION

Visual Human Pose Estimation (HPE) is the task of estimating major body joints of a human agent in a scene. It is a crucial module in systems involving human-machine interaction, with the location and motion of the body joints used by downstream tasks like human action recognition, motion detection, motion analysis, gesture recognition, emotion detection amongst others [12, 24, 34, 38]. The maximum action speed for accurate HPE is constrained by the camera

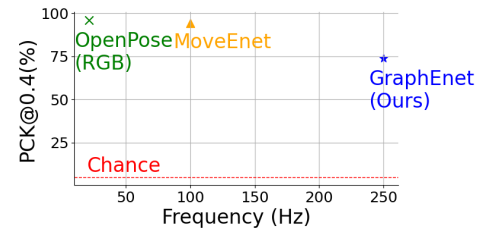


Figure 1. The proposed GraphEnet is an initial investigation into using GNNs for human pose estimation with event cameras, with $2.5\times$ faster than the previous state-of-the-art with a minor performance drop. The chance baseline is calculated assuming a random pixel in the image is chosen for each joint position.

in the system. Fast human motion (e.g. during sports) or fast camera motion (e.g. on a flying drone) induces motion blur in the image, reducing the information available to the algorithm to detect precise joint locations.

Event based cameras are bio-inspired visual sensors, consisting of an array of pixels that capture the *change* in intensity of light in the scene asynchronously. If no change occurs, no data is produced; instead lighting change results in a sequential stream of only those pixels that detected change. This gives rise to a number of interesting properties, such as low energy consumption, reduced data transmission and low latency, all of which can be exploited in a range of applications where power and computation are constrained, including mobile robots and portable electronics. Embedded algorithms require similar properties (low latency and resource computation) to be ideally positioned to bridge the gap from sensory input to higher level applications while leveraging the unique properties of event cameras. In fact, complex, higher level vision applications like

HPE are scarcely explored with event cameras because accurate estimation may require more computation, with high latency, negating the advantage of the event cameras. In this work, the authors explore the trade off between accuracy and latency, pushing the limit of operational frequency at the expense of accuracy.

This work proposes a Graph Neural Network (GNN) for the estimation of 2D human pose based on a stream of events. Graph Neural Networks [39] have gained traction in recent years, due to their flexibility with data input size and sparse computation, giving them an ability to adapt to a wide variety of applications, from drug discovery to social network recommendation systems; they can be applied to any type of data that can be encoded into a graph. In contrast, convolutional neural networks typically require dense data of constant size - e.g. as contained in images produced by traditional cameras. Event streams are sparsely populated, since events are only produced where and when there has been change in intensity leading to GNNs being more appropriate for such data.

This paper investigates the potential of a GNN to estimate human pose using event camera data, and the focus is on the trade off between accuracy and latency. Previous works using events as the input to a GNN created the graph directly from events [31], which can be computationally heavy due to the sheer number of data points. The graph creation step is often a bottleneck in the pipeline, which adds substantial latency to the full system. This paper proposes using a sparse line based event representation to build the initial input graph for the GNN, and investigates promising GNN processing methods to learn and extrapolate the relevant information from such an input graph. The proposed architecture obtains 74% PCKt@0.4 accuracy on event-Human 3.6 Million dataset with an update rate of 250 Hz, as shown in Fig. 1.

Overall, this work has the following contributions:

- Propose the use of line segment features to reduce the graph building time, and dimensionality, and later reduce the computational time in the GNN layers, leading to the first real-time GNN for event cameras.
- An architecture with lower latency than state-of-the-art methods for Human Pose Estimation with event cameras and the first GNN applied to this scenario.
- An in-depth ablation study to analyse the contribution of each component to the final results.

2. RELATED WORK

This work sits at the intersection between Graph Neural Networks, Human Post Estimation, and Event Cameras.

2.1. Human Pose Estimation

In recent years, numerous studies have been published on human pose estimation using RGB data, encompassing ap-

proaches from skeletal poses and localisation of specific poses to volumetric representations of the entire body [3, 12, 24, 34]. Bottom-up artificial neural network (ANN) approaches initially detect limbs before grouping them for each human, with OpenPose [7] being a seminal work that is still widely used because of its ease of use and accuracy, providing a solid baseline method. Top-down ANN approaches [28, 32] use single-person pose estimation for each individual detected in the raw input: MoveNet is such a system that runs at high-frequency, although it did not have a published paper at the time of writing [4].

Event-based methods In the domain of event-based vision, research on HPE is relatively limited. EventCap [36] used a hybrid event and frame camera for high-frequency 3D volumetric pose. However, the accuracy relied on a backwards optimisation step between frames, making it an offline-only system. EventHPE [41] uses a greyscale image to estimate initial pose and events and optical flow to estimate pose changes. These methods are less practical as they require multiple cameras.

LiftMono-HPE [30] employs an ANN-based system to estimate a 3D skeleton pose at 2 Hz, from a monocular event camera using torso length as a depth prior. Event-PointPose [11] estimated 2D human pose with low latency by exploiting 3D point cloud processing techniques applied to events. The baseline method for DHP19 [6] used stereo event cameras to perform 3D HPE. MoveEnet [18] is the only network that operated directly on a single camera's event stream without prior information. It converts the input event stream to a time invariant event surface, from which a multi-headed CNN extracts human pose. Since then, [37] has proposed a Vision Transformer model for HPE.

EventGAN [40] presents a method to pre-train models designed for event based data in order for fast model convergence, and improving on object detection and the HPE task.

2.2. Graph Neural Networks

Graph Neural Networks (GNNs) are a class of neural networks designed to operate on graph-structured data which have found applications in vision [27], motion prediction [26] and human action recognition [1]. Nodes can represent superpixels (groups of pixels with a shared property) and edges are created based on region adjacency or K-nearest neighbours. There are a number of design choices: type of convolution, aggregate function, hierarchical vs flat pooling, node selection methods, number of layers and sequencing. Very few applications based on human input are explored with GNNs, most notably Human Action Recognition and motion prediction [2, 21], but in most works, the human skeleton or pose is first extracted by other methods, then the GNNs are applied on the downward task.

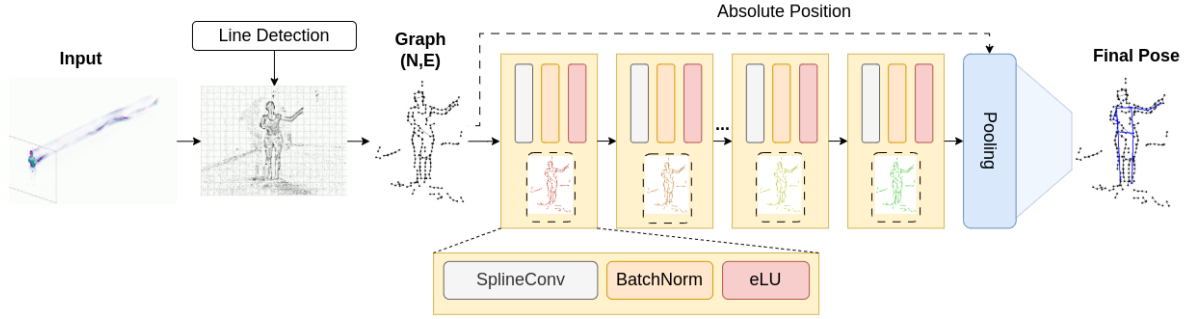


Figure 2. Pipeline of GraphEnet. The input is the continuous, asynchronous stream of events from an event camera. Line detection is performed on a velocity-invariant event accumulation, such that line segments are detected in each grid placed over the surface. The input graph is formed by connecting nearby detected line end-points. The GNN processes the graph, aggregating node features to represent each joint appearance. The final pooling layer extracts joint positions given learnt vector offsets from each node.

GNNs with event based cameras AEGNN [31] proposes an asynchronous event-based GNN for object recognition, creating a graph with each event as a node and which is iteratively updated as each event is produced. However, the graph building is a computationally expensive process in their system. HUGNet [13] proposes a lighter-weight graph building method by applying constraints to the node connection search area, performing optical flow and object recognition [22, 33]. EDGCN [14] uses distillation from frame data to the event-based graph to inform the edges for object detection and classification. [17] merges CNN based features from intensity images with high temporal resolution events to achieve a low latency estimation of pedestrians walking onto a busy road.

To the best of the authors’ knowledge there are no research articles applying event-based GNNs for any tasks related to human input, such as action, pose or motion estimation.

2.3. Line segments based methods

Line segments have frequently been explored as an intermediate representation for vision applications with RGB cameras [23]. They have also been explored for a few tasks with event data. LECalib [25] detects line segments in events integrated over a fixed period of time for camera calibration. Another work [9] uses a line-based representation to track the change in pose of a camera and later for a full SLAM pipeline [10]. Powerline has been tracked with line estimation onboard drones fitted with event cameras [15], by detecting planes in the spatio-temporal space of events. Other works have also proposed general methods to extract line segments from event data [29, 35].

3. Methods

The GNN consists two main components: (1) building the initial graph from the event stream, and (2) processing the

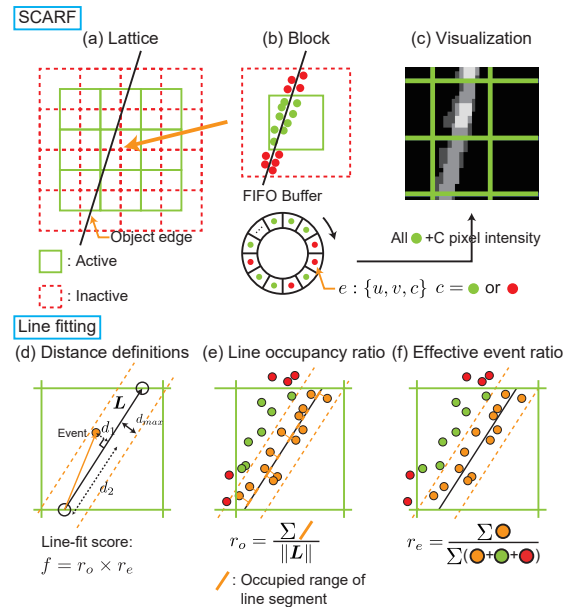


Figure 3. Algorithm overview to extract line segment features from raw events in real time [19]. (a) Lattice structure consists of active and (overlapping) inactive regions. (b) Each block stores active/inactive events together based on a FIFO principle. (c) Image-like representations can be generated by adding pixel intensities to all active events. (d) Line-fit score is calculated with (e) line occupancy ratio and (f) effective event ratio.

graph to aggregate information and extract most likely joint locations. An overview of the pipeline is shown in Fig. 2.

3.1. Building the graph

3.1.1. Event stream to line segment features

Raw events are converted to line segment features at a very high frequency (> 1 kHz) using line detection and tracking with perturbation as in [19] and overviewed in Fig. 3. The

proposed method first uses a velocity invariant event accumulation named the Set of Centre Active Receptive Fields (SCARF), which is fast, asynchronous, and performed for each and every event. Line detection and tracking cannot be performed “event-by-event”, and is instead run as-fast-as-possible using the most recent events in each grid-block of the SCARF representation. A single line is fit to the pixels in each block, using a random sample consensus method to fit a 2 parameter line model. Lines are rejected if there are not enough events in the block, or if the line-fit score is below a threshold.

Each of the two processes, building the event surface (μs latency) and extracting line segment features (ms latency) from the event surface, can be processed simultaneously and in parallel (on a separate computational core) to the GNN components, thereby enabling a high event throughput ($> 20e^6$ events per second) in real-time, without blocking for GNN processing. The line features can be sampled at a millisecond temporal resolution, and as required by the respective downstream steps.

3.1.2. Features to graph

The group of extracted line segment features is then converted into a graph representation ($G(N, E)$ where N is the set of nodes and E is the set of edges), where the line segment features form the initial set E . A processing step is required to explicitly connect E and place nodes at the connection points. The initial connection is trivial as neighboring lines have identical end-point positions. E are assigned a feature describing relative position of the two nodes they connect, and N are assigned features describing the pixel position on the image plane.

As there are a fixed number of blocks in the line feature method, but not each block necessarily contains a line, there is a variable number of nodes and edges in the graph. However, GNNs, differently to convolutional neural networks, are conducive to processing with variation on input size.

3.1.3. Graph Augmentation

When a graph is processed by a GNN, the features of each node are passed to neighbouring nodes connected by edges. A perfectly built graph from the input data would theoretically produce a well connected graph contouring objects in the scene, allowing each node to represent the shape of the contour at that point. Due to noise and inaccuracy, lines can be missing or edges may not align perfectly, severing the information flow through the graph. To reduce the effect of such problems, the edges in the graph have been augmented, based on the following condition: an edge is added between 2 previously unconnected nodes $N_1(x_1, y_1)$ and $N_2(x_2, y_2)$ if:

$$dist(N1, N2) < \zeta, \quad (1)$$

where $dist()$ is the Euclidean distance between the points in pixels and ζ is a hyperparameter empirically set to a value

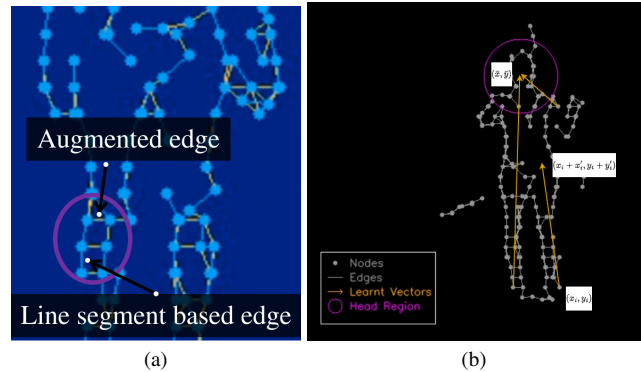


Figure 4. Graph building and joint regression, using (a) missing edges in the graph are added if the end-points are nearby (Equation 1), and (b) offset vectors are learned that point from node positions, to expected joint positions, given a learned confidence weighting.

slightly larger than the block size of the line segment detection step. Fig. 4 show the difference in connectivity of a graph with augmentation applied.

3.2. Processing the graph

The absolute and relative position of the nodes was strongly informative for the task, thus SplineConv layers [16] were integrated into the network as the primary message passing and aggregating layers. SplineConv function consists of a continuous spatial convolutional kernel using B-spline bases, and incorporates position data in the feature aggregation that occurs during GNN processing. As with any deep neural network, a number of SplineConv layers are stacked along with batch normalisation and non-linearity. The SplineConv kernel size is changed across the layers. Two different variation of these kernel sizes were tested: biconic (increase and then decrease, with first and last layer at similar size) and conic (gradual increase with maximum size at the last layer).

Intuitively, the process undertaken by the network layers is similar to that of a convolutional neural network. However, information is passed to the local neighbourhood of each node in a sparse manner. The receptive field of each node, or n-hop neighbourhood increases as the the graph is processed by the layers, enabling nodes to accumulate positional information of more distant nodes. Therefore, node features accumulate information that can be consistently representative of, and unique to, a specific body joint.

3.2.1. Pooling layer

The final layer of the network is a pooling later that clusters the nodes to obtain the 13 body joints of the human agent. To this end, we implemented a custom pooling method.

Each node, i , is trained to estimate a potential location for each joint, j , as an offset to its own position ($x'_{i,j}, y'_{i,j}$)

(see Fig. 4). Additionally, a confidence($c_{i,j}$) is associated to each offset. We tested two different variations of the confidence value: (1) a single confidence per node-joint pair, and (2) each node-joint pair has independent confidence for x and y dimensions (labeled: axis separated).

The pooled joint position (\hat{x}_j, \hat{y}_j) for joint j (where j is removed for simplicity) given each node, i , is calculated by:

$$(\hat{x}, \hat{y}) = \left(\sum_{i=0}^{N-1} c_{x_i}(x_i + x'_i), \sum_{i=0}^{N-1} c_{y_i}(y_i + y'_i) \right), \quad (2)$$

where \hat{x}, \hat{y} : estimated joint position, N : number of nodes, (x_i, y_i) : position of the i^{th} node, (x'_i, y'_i) : learnt offset of the i^{th} node, and (c_{x_i}, c_{y_i}) : confidence for the i^{th} node. For the single confidence value ($c_{y_i} = c_{x_i}$).

3.2.2. Losses and Training

The final model loss, l_{total} incorporates a target loss and a node loss according to $l_{total} = \alpha l_{target} + \beta l_{node}$, where each loss is calculated using the mean squared error (MSE):

$$l_{target} = \text{MSE}((\hat{x}, \hat{y}), (\bar{x}, \bar{y})), \quad (3)$$

$$l_{node} = \left(\sum_{i=0}^{N-1} \text{MSE}((x_i + x'_i, y_i + y'_i), (\bar{x}, \bar{y})) \right), \quad (4)$$

where (\bar{x}_j, \bar{y}_j) is the ground truth position of joint j (j removed for readability), and α and β are tunable parameters.

Various training paradigms were tested (results reported in Tab. 4). The basic paradigm, labelled *together*, trained the full architecture with l_{total} loss, with varying values of α and β . Paradigms *node-only* and *target-only* used only the l_{node} ($\alpha = 0, \beta = 1$) and l_{target} ($\alpha = 1, \beta = 0$) respectively. Finally, the *staggered* paradigm first trained only on the l_{node} ($\alpha = 0, \beta = 1$) for 20 epochs, thus training the nodes to point to the joints. Then l_{target} is included ($\alpha = 1, \beta = 1$) for 50 epochs, to train the confidence values as well.

4. EXPERIMENTS

Experiments perform a comparison to the state-of-the-art as well as ablation studies, for both speed and accuracy.

4.1. Datasets

The architecture was tested on two different datasets, to enable a comparison with state-of-the-art methods. Each dataset has the ground truth joint positions for HPE using motion capture systems.

The **e-Human 3.6M dataset (eH36M)** is a semi-synthetic dataset with a spatial resolution of 640x480. It is created by converting the large-scale benchmark video dataset Human 3.6 million [20] to events (process described in [18]). The dataset consists of 7 subjects (3 female, 4

male), each performing 15 actions. 5 subjects are used in training and 2 for validation. The results shown in the paper are on the validation dataset.

The **DHP19 dataset** [6] is a large-scale event-based dataset for human pose estimation. Its 4 cameras have the resolution of 346x280, acquiring data from 17 subjects (12 female, 5 male), each performing 33 movements. The data is split by subject, 12 for training and 5 for validation.

4.2. Metrics

Two standard metrics have been adopted.

Percentage of correct keypoints (PCK) defines a keypoint (i.e. joints) as correct if it is within some tolerance of the ground-truth position. The metric is defined as:

$$\frac{100}{M} \sum_{j=1}^M \delta(pT - d_j), \quad \text{where } \delta(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (5)$$

where d_j is the Euclidean distance between the predicted and Ground Truth positions of the j^{th} joint, $M=13$ is the number of joints/keypoints, T is a threshold equal to the diagonal torso size of the human agent in pixels and p is a scaling factor in the range (0,1]. Since the PCK threshold is extracted from the sample that is being evaluated, it is independent of the input resolution, the height of the subject and the distance of the subject from the camera. Using the diagonal length of the torso makes it robust to change in orientation of the camera and most camera viewpoints.

Mean Per Joint Position Error (MPJPE) is calculated as the average Euclidean distance between ground-truth and prediction as follows:

$$\text{MPJPE} = \frac{1}{M} \sum_j \|x_j - \hat{x}_j\|, \quad (6)$$

where N is the number of skeleton joints, x_j and \hat{x}_j are, respectively, the ground truth and predicted position of the j^{th} joint in the image space. MPJPE is frequently used in HPE benchmarks and facilitates comparison with other works, but it is affected by factors like camera resolution.

4.3. Comparison to state-of-the-art

There are few works on 2D HPE with event-cameras, and fewer still that provide results on benchmark datasets. TORE[5] provides results on DHP19, but separate metrics on camera 2 and 3, foregoing the other two camera viewpoints entirely. EventPointPose [11], an event based HPE model for 3D Pose also provides 2D results on DHP19 with the MPJPE metric and latency information. [37] neither has results on any benchmark datasets, nor the code available at the time of writing. MoveEnet [18] provides results for both DHP19 and eH36M. Results on DHP19 dataset are available from DHP19 method [6] (which also uses greyscale

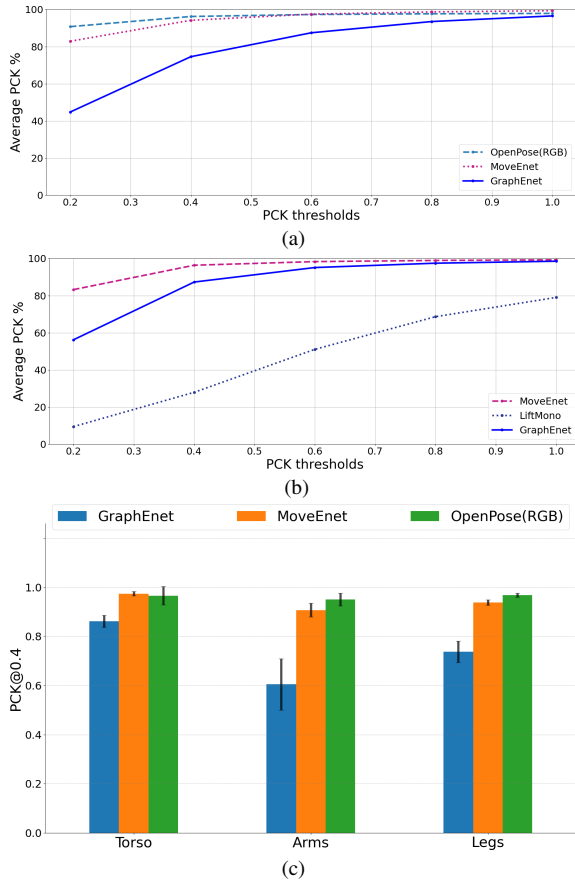


Figure 5. Comparison with state-of-the-art using a variation on PCK thresholds on the (a) eH36M dataset and (b) DHP19 dataset. The eH36M results are clustered by body-part type in (c).

intensity values that they do not provide with the dataset) and LiftMono-HPE[30] (in [18]).

Our accuracy comparison results are shown in Table 1 and Fig. 5. Since eH36M is derived from RGB data, this enables a comparison with RGB based models. The conversion is generally a lossy process that only recreates high temporal resolution through interpolation, thus an event based method on eH36M would likely not supersede the accuracy or precision of a SOTA RGB based method on the original dataset, but can provide an upper bound for context and comparison. Thus we also show results from OpenPose [7].

4.4. Joint-wise results

Table 2 show the performance of GraphEnet on each joint in the eH36M dataset as compared to MoveEnet and OpenPose. The head is a strongly stable joint for all networks, in addition to joints on the torso. Instead, limb joints are less stable, with the line features representing them less consistently over time and body poses. To further analyse this, the

| Model | PCK@0.4 \uparrow | PCK@0.6 \uparrow | MPJPE _{2D} \downarrow |
|--------------------|--------------------|--------------------|----------------------------------|
| eH36M | | | |
| OpenPose((RGB) [7] | 0.96 | 0.97 | 17.07 |
| MoveEnet [18] | 0.94 | 0.97 | 19.68 |
| GraphEnet(Ours) | 0.74 | 0.88 | 36.21 |
| DHP19 | | | |
| MoveEnet [18] | 0.97 | 0.98 | 6.28 |
| LiftMono [30] | 0.28 | 0.51 | 26.79 |
| DHP19 [6] | - | - | 7.03 |
| GraphEnet(Ours) | 0.87 | 0.95 | 12.90 |

Table 1. PCK comparison.

| Joint | GraphEnet | | MoveEnet [18] | | OpenPose [7] | |
|----------------|-----------|-------|---------------|-------|--------------|-------|
| | Events | | Events | | RGB Frames | |
| Input Type | Accu | Error | Accu | Error | Accu | Error |
| head | 0.90 | 24.73 | 0.99 | 13.84 | 0.90 | 18.15 |
| shoulder_right | 0.86 | 29.60 | 0.97 | 17.54 | 0.99 | 15.23 |
| shoulder_left | 0.87 | 27.82 | 0.97 | 16.19 | 0.99 | 13.00 |
| hip_left | 0.84 | 29.01 | 0.97 | 15.30 | 0.97 | 16.93 |
| hip_right | 0.84 | 29.93 | 0.97 | 17.00 | 0.98 | 20.32 |
| elbow_right | 0.71 | 42.35 | 0.94 | 23.03 | 0.98 | 18.02 |
| elbow_left | 0.68 | 39.34 | 0.92 | 20.48 | 0.96 | 15.79 |
| wrist_right | 0.52 | 54.73 | 0.89 | 30.15 | 0.94 | 21.01 |
| wrist_left | 0.51 | 50.83 | 0.88 | 27.18 | 0.92 | 16.22 |
| knee_right | 0.79 | 33.16 | 0.95 | 18.99 | 0.98 | 14.41 |
| knee_left | 0.77 | 32.41 | 0.94 | 17.75 | 0.97 | 15.69 |
| ankle_right | 0.70 | 39.09 | 0.94 | 20.27 | 0.97 | 17.68 |
| ankle_left | 0.69 | 37.77 | 0.92 | 20.57 | 0.96 | 19.52 |
| Average | 0.74 | 36.21 | 0.94 | 19.87 | 0.96 | 17.07 |

Table 2. Joint-wise breakdown of the accuracy (PCKt@0.4) and error (2D MPJPE) on the eH36M dataset

| Model | Latency \downarrow | Frequency \uparrow |
|-------------------|----------------------|----------------------|
| OpenPose(RGB) [7] | 46ms | 20Hz |
| MoveEnet [18] | 10ms | 100Hz |
| GraphEnet(Ours) | 4ms | 250Hz |

Table 3. Table comparing with SOTA for eH36M dataset

13 joints considered in this work are separated into three clusters: (1) Torso: head, shoulders and hips, (2) Legs: knees and ankles, and (3) Arms: elbows and wrists. Fig. 5c shows the PCK accuracy on joint clusters. GraphEnet estimates the torso joints with a higher accuracy, very close to the state of the art methods, and instead has more difficulty correctly detecting the joint positions of the legs and arms.

4.5. Latency

Appropriate algorithms can leverage the low latency of event cameras without adding excessive latency themselves, in order to create a high frequency pipeline. With the parallelisation of processes, the frequency of the full pipeline can be defined by the slowest thread. Table 3 reports the processing time and effective latency of the slowest process for each algorithm. Components of the pipeline are considered different modules if they can be parallelised and

| Experiment/Model | Conf. | ζ | Feature | Training | PCK@0.4 \uparrow | PCK@0.6 \uparrow | $MPJPE_{2D}$ \downarrow |
|-------------------|-----------------------|-----------|-------------|--------------------|--------------------|--------------------|---------------------------|
| Baseline | single | 15 | Biconic | Staggared | 0.50 | 0.67 | 62.47 |
| Dual-contribution | axis-separated | 15 | Biconic | Staggared | 0.45 | 0.63 | 68.02 |
| No Augmentation | single | 0 | Biconic | Staggared | 0.28 | 0.47 | 83.16 |
| More Augmentation | single | 20 | Biconic | Staggared | 0.49 | 0.63 | 67.88 |
| Feature-shape | single | 15 | Cone | Staggared | 0.57 | 0.71 | 56.89 |
| Training | single | 15 | Biconic | node-only | 0.21 | 0.37 | 93.01 |
| Training | single | 15 | Biconic | target-only | 0.55 | 0.71 | 57.06 |
| Training | single | 15 | Biconic | together | 0.49 | 0.66 | 64.85 |

Table 4. Ablation Study on the e-H3.6M validation set. Conf.: Confidence. ζ : Threshold for additional augmented edges. Feature: Shape of features over layers

the reported latency is the maximum of such a module. Tests for OpenPose [7] and MoveNet [18] are done on an Intel Core i9-10980HK @2.40GHz x 16 and NVIDIA GeForce RTX 2070 while GraphEnet was tested on an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz x 12 with a NVIDIA GeForce GTX 1650. The slight difference in hardware was due to implementation constraints, but it is expected that GraphEnet will actually run faster if tested using the more powerful i9 machine.

GraphEnet is the first event-based GNN that can perform the target task in real time, and the pipeline can be parallelised to obtain a runtime frequency of 250Hz. The trade off between the accuracy and latency of the various methods is visualised in Fig. 1.

4.6. Ablation studies

Tab. 4 reports the various network architectures and their resulting accuracies. Each model was trained on a subset of eH36M for 50 epochs.

Since a large amount of training was necessary to produce the ablation comparison, for smarter use of resources, the training was conducted on a defined fraction of the training split eH36M dataset. Based on the observations of the ablations, the features indicating highest accuracy include: single confidence, augmentation threshold of 15, cone shaped node features, with target-only training. The above system was trained on the full training split, and used for results in Section 4.4 and Section 4.5.

4.7. Qualitative Results

Fig. 6 and 7 show the qualitative results of GraphEnet from DHP19 and eH36M, respectively. The prediction on DHP19 superimposes on the ground truth substantially, as is expected from the quantitative results. Results are consistent even with self occlusion. Fig. 7 shows that the model performs reasonably well even where there are noisy line segments present in the scene (see row 3) but tends to fail when the noise superimposes on the person (see row 4) or the information from the joints is too sparse. The level of granularity and detail in the extremities (hands and feet) varies in the graph, thus leading to a lower robustness in

the estimation of resulting pose, consistent with the quantitative results.

5. DISCUSSION

GraphEnet performs well on DHP19 samples and shows promising results on eH36M, but with a lower latency and higher frequency compared to other state-of-the-art methods.

GraphEnet can distinguish the body from the noisy background based on the line connectivity. The torso joints are estimated well and in a stable manner. However, arms and legs are more challenging for the GNN to detect correctly. The input graphs show that they have less consistent input information about the limbs. The area occupied by the limbs is lower compared to the torso, leading to less nodes, and the intricacy of the joints (e.g. hands and fingers) is not well represented by line segments. The low spatial resolution of the two datasets is also detrimental to finer details.

Finding a balance between speed (i.e. graphs from line segments) and detail (i.e. graphs directly from events) is needed in the graph-making process. The line segments features build a graph, lowering latency, but a hierarchical model can be developed to obtain more detailed graph edges around the regions of the hands to improve accuracy at a relatively small cost of latency, in the future. The novelties proposed in the paper made it possible to achieve the target task. The ablation study indicates that the model with a single confidence, with a connectivity slightly larger than the block size, 10 layers, conic features size and trained on the target loss performs the best. Pretraining maybe also mitigate these issues with methods like [40] or [8].

While the proposed model is promising, there is scope for improvement. Heterogeneous graphs can be adopted, bridging the gap between line segment features and events, creating more detailed nodes where needed. An asynchronous approach can also be adopted, updating the input graph when there is change in the scene, and updating only relevant parts of the GNN (like in [31]).

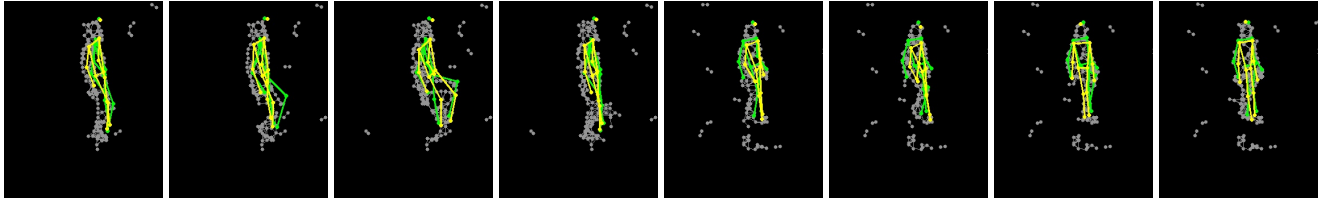
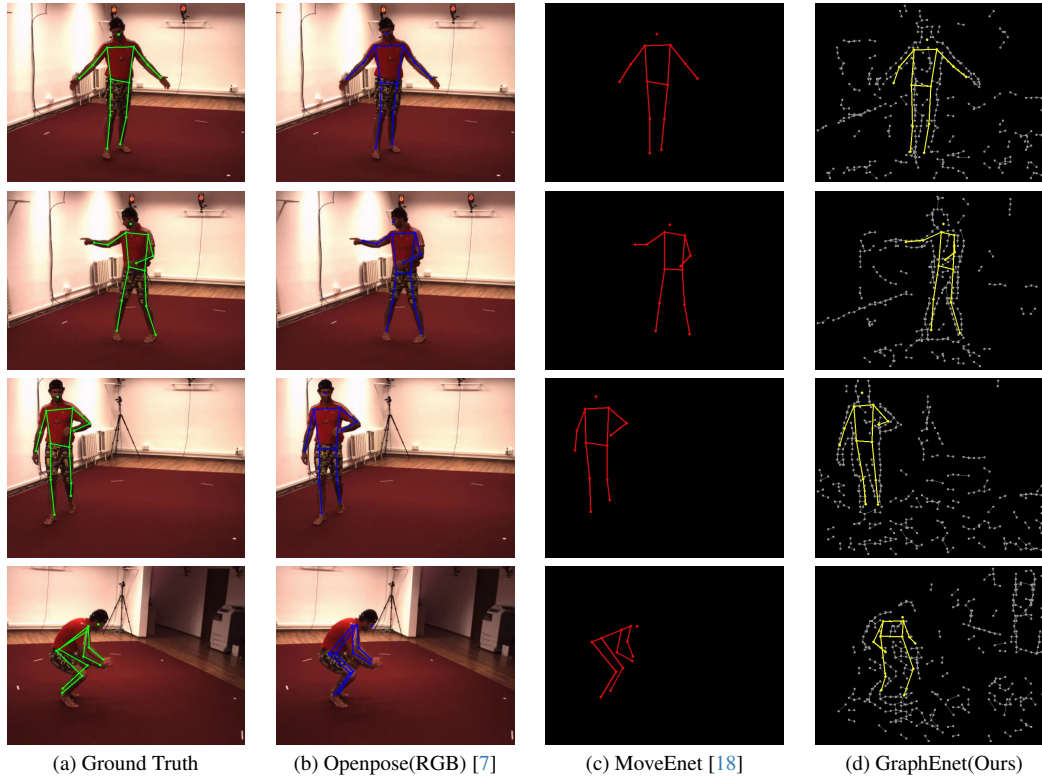


Figure 6. Qualitative results on a sample from DHP19. Ground Truth in green, GraphEnet in yellow



(a) Ground Truth

(b) Openpose(RGB) [7]

(c) MoveEnet [18]

(d) GraphEnet(Ours)

Figure 7. Qualitative results on samples from eH36M

6. CONCLUSION

We introduced GraphEnet for human pose estimation with a graph neural network. GraphEnet achieved reasonable (74% v.s. 94%) accuracy performance compared to the state-of-the-art, but with a lower latency (4 ms v.s. 10 m.s.) and higher output frequency(250 Hz v.s. 100 Hz). These preliminary results suggest that the use of GNNs can reduce latency and computational cost, while increasing the estimation frequency.

The confidence-based offset learning in the final pooling layer of the proposed architecture was necessary for strong performance. Such a pooling layer would also be useful for other tasks, such as object recognition. The comprehensive ablation study further contributed to understanding of the individual impact of the various architectural and training components.

GNNs, together with event cameras, enable fully sparse data pipelines realising an overall lower computational cost. Such systems can reduce the energy cost of AI systems and reduce the amount of GPU servers required, leading to positive impacts on the environment.

ACKNOWLEDGEMENTS

This work was funded and supported scientifically by Sony Interactive Entertainment.

References

- [1] Tasweer Ahmad, Lianwen Jin, Xin Zhang, Songxuan Lai, Guozhi Tang, and Luojun Lin. Graph convolutional neural network for human action recognition: A comprehensive survey. *IEEE Transactions on Artificial Intelligence*, 2(2): 128–145, 2021. 2

- [2] Tasweer Ahmad, Lianwen Jin, Xin Zhang, Songxuan Lai, Guozhi Tang, and LuoJun Lin. Graph convolutional neural network for human action recognition: A comprehensive survey. *IEEE Transactions on Artificial Intelligence*, 2(2): 128–145, 2021. 2
- [3] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2
- [4] The TensorFlow Hub Authors. Movenet: Ultra fast and accurate pose detection model, 2021. <https://neurotechai.eu/resources/datasets/>, Accessed: 2022-05-20. 2
- [5] R Wes Baldwin, Ruixu Liu, Mohammed Almatrafi, Vijayan Asari, and Keigo Hirakawa. Time-ordered recent event (tore) volumes for event cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2519–2532, 2022. 5
- [6] Enrico Calabrese, Gemma Taverni, Christopher Awai Easthope, Sophie Skriabine, Federico Corradi, Luca Longinotti, Kynan Eng, and Tobi Delbruck. DHP19: Dynamic vision sensor 3D human pose dataset. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2019-June:1695–1704, 2019. 2, 5, 6
- [7] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 2, 6, 7, 8
- [8] Nicolò Carissimi, Gaurvi Goyal, Franco Di Pietro, Chiara Bartolozzi, and Arren Glover. [wip] unlocking static images for training event-driven neural networks. In *2022 8th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–4. IEEE, 2022. 7
- [9] William Chamorro, Juan Andrade-Cetto, and Joan Solà. High speed event camera tracking. *arXiv preprint arXiv:2010.02771*, 2020. 3
- [10] William Chamorro, Joan Solà, and Juan Andrade-Cetto. Event-based line slam in real-time. *IEEE Robotics and Automation Letters*, 7(3):8146–8153, 2022. 3
- [11] Jiaan Chen, Hao Shi, Yaozu Ye, Kailun Yang, Lei Sun, and Kaiwei Wang. Efficient human pose estimation via 3d event point cloud. *arXiv preprint arXiv:2206.04511*, 2022. 2, 5
- [12] Yucheng Chen, Yingli Tian, and Mingyi He. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, 192: 102897, 2020. 1, 2
- [13] Thomas Dalgaty, Thomas Mesquida, Damien Joubert, Amos Sironi, Pascal Vivet, and Christoph Posch. Hugnet: Hemispherical update graph neural network applied to low-latency event-based optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3953–3962, 2023. 3
- [14] Yongjian Deng, Hao Chen, Bochen Xie, Hai Liu, and Youfu Li. A dynamic graph cnn with cross-representation distillation for event-based recognition. *arXiv preprint arXiv:2302.04177*, 2023. 3
- [15] Alexander Dietsche, Giovanni Cioffi, Javier Hidalgo-Carrió, and Davide Scaramuzza. Powerline tracking with event cameras. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6990–6997. IEEE, 2021. 3
- [16] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877, 2018. 4
- [17] Daniel Gehrig and Davide Scaramuzza. Low-latency automotive vision with event cameras. *Nature*, 629(8014):1034–1040, 2024. 3
- [18] Gaurvi Goyal, Franco Di Pietro, Nicolo Carissimi, Arren Glover, and Chiara Bartolozzi. Moveenet: Online high-frequency human pose estimation with an event camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4023–4032, 2023. 2, 5, 6, 7, 8
- [19] Mikihiro Ikura, Arren Glover, Masayoshi Mizuno, and Chiara Bartolozzi. Lattice-allocated real-time line segment feature detection and tracking using only an event-based camera. In *Proceedings of the 2nd Workshop on Neuromorphic Vision: Advantages and Applications of Event Cameras (NeVi) in conjunctions with The International Conference on Computer Vision (ICCV)*, 2025. 3
- [20] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014. 5
- [21] Qin Li, Georgia Chalvatzaki, Jan Peters, and Yong Wang. Directed acyclic graph neural network for human motion prediction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3197–3204. IEEE, 2021. 2
- [22] Yijin Li, Han Zhou, Bangbang Yang, Ye Zhang, Zhaopeng Cui, Hujun Bao, and Guofeng Zhang. Graph-based asynchronous event processing for rapid object recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 934–943, 2021. 3
- [23] Xinyu Lin, Yingjie Zhou, Yipeng Liu, and Ce Zhu. A comprehensive review of image line segment detection and description: Taxonomies, comparisons, and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 3
- [24] Wu Liu and Tao Mei. Recent advances of monocular 2d and 3d human pose estimation: A deep learning perspective. *ACM Computing Surveys (CSUR)*, 2022. 1, 2
- [25] Zibin Liu, Banglei Guan, Yang Shang, Zhenbao Yu, Yifei Bian, and Qifeng Yu. Lecalib: Line-based event camera calibration. *Measurement*, 235:114900, 2024. 3
- [26] Kedi Lyu, Haipeng Chen, Zhenguang Liu, Beiqi Zhang, and Ruili Wang. 3d human motion prediction: A survey. *Neurocomputing*, 489:345–365, 2022. 2
- [27] Usman Nazir, He Wang, and Murtaza Taj. Survey of image based graph neural networks. *arXiv preprint arXiv:2106.06307*, 2021. 2

- [28] Lingteng Qiu, Xuanye Zhang, Yanran Li, Guanbin Li, Xiaojun Wu, Zixiang Xiong, Xiaoguang Han, and Shuguang Cui. Peeking into occluded joints: A novel framework for crowd pose estimation. In *European Conference on Computer Vision*, pages 488–504. Springer, 2020. [2](#)
- [29] David Reverter Valeiras, Xavier Clady, Sio-Hoi Ieng, and Ryad Benosman. Event-based line fitting and segment detection using a neuromorphic visual sensor. *IEEE Transactions on Neural Networks and Learning Systems*, 30(4): 1218–1230, 2019. [3](#)
- [30] Gianluca Scarpellini, Pietro Morerio, and Alessio Del Bue. Lifting monocular events to 3d human poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1358–1368, 2021. [2](#), [6](#)
- [31] Simon Schaefer, Daniel Gehrig, and Davide Scaramuzza. Aegnn: Asynchronous event-based graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12371–12381, 2022. [2](#), [3](#), [7](#)
- [32] Kai Su, Dongdong Yu, Zhenqi Xu, Xin Geng, and Changhu Wang. Multi-person pose estimation with enhanced channel-wise and spatial information. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5674–5682, 2019. [2](#)
- [33] Daobo Sun and Haibo Ji. Event-based object detection using graph neural networks. In *2023 IEEE 12th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 1895–1900. IEEE, 2023. [3](#)
- [34] Jinbao Wang, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He, and Ling Shao. Deep 3d human pose estimation: A review. *Computer Vision and Image Understanding*, 210:103225, 2021. [1](#), [2](#)
- [35] Xinya Wang, Haitian Zhang, Huai Yu, and Xianrong Wan. Evlsd-ied: Event-based line segment detection with image-to-event distillation. *IEEE Transactions on Instrumentation and Measurement*, 2024. [3](#)
- [36] Lan Xu, Weipeng Xu, Vladislav Golyanik, Marc Habermann, Lu Fang, and Christian Theobalt. Eventcap: Monocular 3d capture of high-speed human motions using an event camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4968–4978, 2020. [2](#)
- [37] Nannan Yu, Tao Ma, Jiqing Zhang, Yuji Zhang, Qirui Bao, Xiaopeng Wei, and Xin Yang. Adaptive vision transformer for event-based human pose estimation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 2833–2841, 2024. [2](#), [5](#)
- [38] Feng Zhang, Xiatian Zhu, and Chen Wang. Single person pose estimation: a survey. *arXiv preprint arXiv:2109.10056*, 2021. [1](#)
- [39] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020. [2](#)
- [40] Alex Zihao Zhu, Ziyun Wang, Kaung Khant, and Kostas Daniilidis. Eventgan: Leveraging large scale image datasets for event cameras. In *2021 IEEE international conference on computational photography (ICCP)*, pages 1–11. IEEE, 2021. [2](#), [7](#)
- [41] Shihao Zou, Chuan Guo, Xinxin Zuo, Sen Wang, Pengyu Wang, Xiaoqin Hu, Shoushun Chen, Minglun Gong, and Li Cheng. Eventhpe: Event-based 3d human pose and shape estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10996–11005, 2021. [2](#)