2023 IEEE International Conference on Image Processing (ICIP) | 978-1-7281-9835-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICIP49359.2023.10222777

INTRODUCING A FRAMEWORK FOR SINGLE-HUMAN TRACKING USING EVENT-BASED CAMERAS

Dominik Eisl[*] *Fabian Herzog*[*] *Jean-Luc Dugelay*[†] *Ludovic Apvrille*[‡] *Gerhard Rigoll*[*]

[*] Institute of Human-Machine Communication, Technical University of Munich, Germany
[†] Digital Security Department, EURECOM, Sophia Antipolis, France
[‡] LTCI, Telecom Paris, Institut Polytechnique de Paris, France

ABSTRACT

Event cameras generate data based on the amount of motion present in the captured scene, making them attractive sensors for solving object tracking tasks. In this paper, we present a framework for tracking humans using a single event camera which consists of three components. First, we train a Graph Neural Network (GNN) to recognize a person within the stream of events. Batches of events are represented as spatio-temporal graphs in order to preserve the sparse nature of events and retain their high temporal resolution. Subsequently, the person is localized in a weakly-supervised manner by adopting the well established method of Class Activation Maps (CAM) for our graph-based classification model. Our approach does not require the ground truth position of humans during training. Finally, a Kalman filter is deployed for tracking, which uses the predicted bounding box surrounding the human as measurement. We demonstrate that our approach achieves robust tracking results on test sequences from the Gait3 database, paving the way for further privacy-preserving methods in event-based human tracking. Code, pre-trained models and datasets of our research are publicly available [1].

Index Terms**—** Human Tracking, Event-based Cameras, Kalman Filtering

1. INTRODUCTION

Event-based cameras are bio-inspired vision sensors that recently attracted the attention of researchers for solving different problems in the domain of computer vision. They create instances of data denoted as *events* once the per-pixel brightness change exceeds a given threshold. The generation of events is steered by the dynamics of the captured scene and the overall output is a sparse and asynchronous stream of events. Event-cameras offer some appealing properties such as high temporal resolution, high dynamic range and low energy consumption [1]. Thus, for object tracking problems, they seem to be promising alternatives to standard frame-based cameras for scenarios with high-speed motion or extreme lighting conditions.

Recently, several approaches for event-based object tracking [2–4] have been proposed, which are mainly focused on the tracking of objects having simple shapes. In this work, we examine the tracking of humans using a single, static event camera. When designing systems with standard cameras for video surveillance tasks, oftentimes sensitive, personal information can be inferred from intensity images without the consent of the individual. As such, we believe that event cameras can be beneficial for these applications as they capture data based solely on a person's movement and therefore pose fewer privacy risks.

Thus, we propose a framework for tracking a single person using a static event camera. The system is comprised of the three core components *Human Classifier*, *Human Localizer* and *Human Tracker* which are illustrated in Figure 1. Our contributions can be summarized as follows: First, we introduce the *Human Classifier* which is a convolutional graph neural network trained on the task of binary object classification such that the model can distinguish a single human from background objects. We represent groups of events as spatio-temporal graphs in order to retain the high temporal resolution and sparse nature of events. Then, we present the *Human Localizer*, which is a module that can generate a bounding box surrounding the human within the stream of events. Therefore, we adopt class activation maps [5] for GNNs and exploit the method's weakly supervised object localization abilities. Hence, our approach doesn't depend on the availability of the human's ground truth location. Finally, we implement the *Human Tracker* which deploys a Kalman filter for tracking a detected human throughout the course of a given event stream.

2. RELATED WORK

While extensive research has been conducted on human tracking using frame-based cameras, only few works have been published on capturing human features in the domain of event-based vision. [6] presented an approach for event-based tracking of multiple persons. They focus on detecting persons in the case of occlusions by modeling humans as gaussian mixture models. Xu et al. [7] contribute a method for capturing high-speed human motions using a hybrid setting fusing information from intensity images and the event stream. Moreover, [8, 9] propose learning-based methods for 3D human pose estimation. In both approaches, batches of events are accumulated to form frame-like structures for training a CNN-model which predicts heatmaps of 2D body joint positions.

Grouping events into dense event-frame representations is a popular choice for processing the event stream as this has the advantage that established deep learning models and libraries can be leveraged. Hence, [10, 11] solve the problem of event-based object detection by redesigning CNN models to work with events while Perot et al. [12] present a recurrent architecture with ConvLSTM layers to implement a memory mechanism. However, processing a batch of events as dense structure, eliminates their sparse and asynchronous nature. Recently, a line of research has emerged [13–17] which models groups of events as spatio-temporal graphs and thereby enables a sparse representation of events while retaining their high temporal resolution. As a consequence, GNNs have been trained for event-based object detection setting new benchmarks in terms of computational efficiency and latency reduction [13, 16].

[1] Code and data will be shared after review.

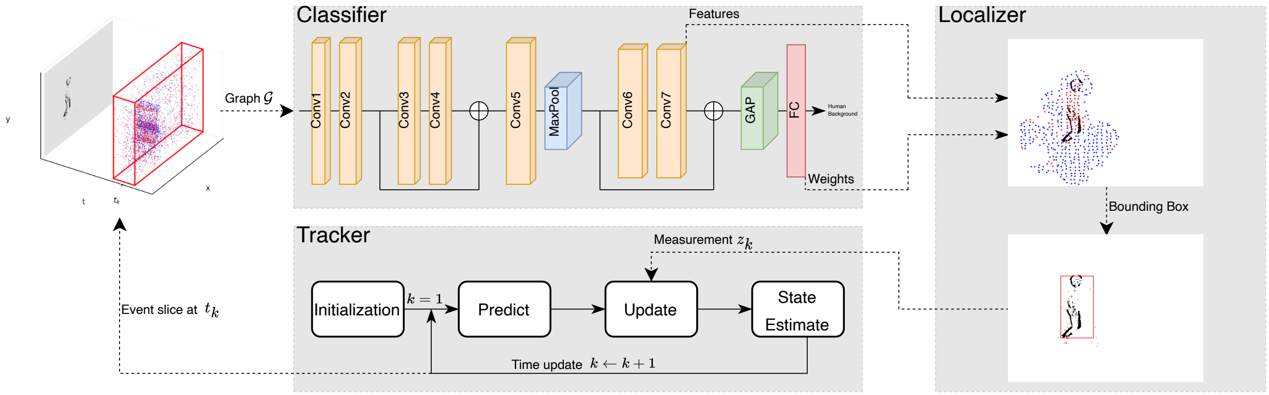

Fig. 1. Our proposed single human tracking framework: The *Human Classifier* is a binary classifier that predicts whether the event graph contains a human or background object. In the case of a predicted human label, the *Human Localizer* produces a bounding box which encodes the location of the human. The *Human Tracker* deploys a Kalman filter for tracking the human which models the generated bounding box as a sensor measurement.

3. METHODOLOGY

3.1. Event Processing

In order to exploit the sparsity of events, we represent them as spatio-temporal graphs and hence follow the approach of [13] and [16]. The event stream is sliced along the temporal dimension into groups of events, each of them corresponding to a time interval of Δt. Therefore, we also refer to the obtained batch of events as event slice. Each obtained subset of the event stream is represented as a set of tuples given by

$$\{e_i\}_N = \{(x_i, y_i, t_i, p_i)\}_N. \quad (1)$$

Thus, an event e_i, is uniquely defined by its position in the pixel coordinate system, x_i and y_i, its time of occurrence t_i and its polarity $p_i \in \{+1, -1\}$ indicating whether the pixel brightness has increased $(+1)$ or decreased (-1).

To compensate for the difference in spatial and temporal resolution, the timestamps are normalized such that $\tilde{t}_i = \beta \cdot t_i$. Moreover, M events are sampled randomly from $\{e_i\}_N$ for saving memory and computation expenses [13]. Subsequently, a spatio-temporal graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$ representing a given event slice in a time interval Δt, is constructed with $\mathcal{N}, \mathcal{E}$ being the set of nodes and edges, respectively. A node i is allocated for every event in $\{e_i\}_M$ and the polarity p_i is assigned as the initial node feature with $f^{(0)}(i) = p_i \in \{+1, -1\}$ as proposed in [13]. The spatio-temporal position $\mathbf{x_i} = (x_i, y_i, \tilde{t}_i)$ [16] of node i is used to find neighboring nodes in the $(x, y, \tilde{t})$–space and to connect them via an edge in $\mathcal{G}$. An edge connecting two nodes i and j is created and added to $\mathcal{E}$ if the Euclidean distance $d_{i,j}$ between $\mathbf{x}_i$ and $\mathbf{x}_j$ is below a fixed threshold R such that

$$d_{i,j} = \sqrt{|x_i - x_j|^2 + |y_i - y_j|^2 + |\tilde{t}_i - \tilde{t}_j|^2} < R. \quad (2)$$

In particular, nodes are connected temporally which is ensured by time normalization. Events in disjoint slices are not connected with each other.

3.2. Tracking Framework

Classification. Inspired by [16], we present the *Human Classifier*, a convolutional GNN, which takes event graphs as input and learns to distinguish a human from background objects within the given slice of events. Its architecture is visualized in Figure 1 and consists of a series of graph convolution blocks and pooling layers that act as a feature extractor as well as a single fully-connected (FC) layer. Within a convolutional block, the feature vectors of all graph nodes are updated by aggregating information from their graph neighborhood. For a given node i, all of its neighboring nodes contribute to $f(i)$ through their node features which are weighted based on their relative positioning to i. Graph pooling is applied for coarsening the graph and obtaining more expressive node features. Ultimately, the FC-layer generates an output based on the extracted features and predicts class probability scores for the labels *human* and *background*.

Localization. In this section we introduce the *Human Localizer* which builds on top of the previously introduced *Human Classifier* and aims at finding the human's location within an event slice once it has been classified as human. One option for implementation is to extend the classification model by a detection head as has been done by [16]. However, this requires the availability of the object's ground truth location which is often hard to obtain. Therefore we apply the well known concept of *class activation maps* (CAM) [5] on our graph-based classification model and exploit its localization capabilities. Pope *et al.* [18] showed a proof-of-concept for adopting CAM as explainability method on convolutional GNNs. Yet, to the best of our knowledge, this is the first work which makes use of CAM's localization ability in the domain of convolutional GNNs for object tracking.

First, we designed the *Human Classifier* such that it meets the requirements for applying CAM. We deployed a *global average pooling* (GAP) layer after the last *conv*-block followed by a single FC-layer using Softmax. Pope *et al.* [18] derived a node-wise computation scheme for obtaining the CAM heat-maps which is described in the following. Let $\mathbf{F} \in \mathbb{R}^{N \times K}$ be the feature matrix of the graph which has just been processed by the last *conv*-block of the classifier where F_{nk} denotes the k-th feature of node n. Further, let $w_k^{(c)}$ denote the weight of the FC-layer activated by the k-th feature and corresponding to class c. Then the CAM score of node n with respect to class c can be calculated as

$$L_{CAM}[n] = \sum_k w_k^{(c)} F_{nk}. \quad (3)$$

We observe that one can compute the CAM scores for all nodes in the graph by a simple matrix vector multiplication. Let $\mathbf{w}^{(c)}$ be a vector holding the weights of the FC-layer belonging to class c. Then we can compute a vector $\mathbf{l}^{(c)} \in \mathbb{R}^N$ with class-specific discriminative scores for all nodes by

$$\mathbf{l}^{(c)} = \mathbf{F} \cdot \mathbf{w}^{(c)}. \tag{4}$$

The scores in $\mathbf{l}^{(c)}$ can be mapped to their respective nodes' pixel coordinates to obtain a heat-map $\mathbf{M}(x, y)$ that is inherently sparse compared to the dense CAM from [5]. Thus, we adopted the procedure for generating a bounding box using our sparse heat-map while still following the original idea. In analogy to [5], our approach finds the largest connected component within the subset of points that have a score in $\mathbf{l}^{(c)}$ which lies above the 80-th percentile. First, DBSCAN clustering [19] is applied for grouping points into clusters which is equivalent to identifying connected components within this set of points. Subsequently, we developed a heuristic which examines whether multiple clusters are in a meaningful configuration. That is, based on the clusters' density and their relative positioning, multiple clusters can be combined to form a larger cluster. Then, a bounding box is generated from the smallest rectangle surrounding all cluster points.

Tracking. We design a single human tracking framework for event-based cameras which is depicted in Figure 1. The *Human Tracker* deploys a Kalman filter which models the movement of the person inside the stream of events as a discrete time-controlled process. The discretization along the temporal axis is obtained by slicing the event stream into groups of events from constant time intervals with Δt. At each iteration step, we want to track the bounding box surrounding the human given that a person is present in the respective event slice. Similar to [20], we define the system state vector holding the bounding box in the two-corner representation along with a velocity component for each position parameter such that

$$\mathbf{x} = [x_{\min}, \dot{x}_{\min}, y_{\min}, \dot{y}_{\min}, x_{\max}, \dot{x}_{\max}, y_{\max}, \dot{y}_{\max}]^T. \tag{5}$$

The physical process is governed by the state transition function as described in [21]:

$$\mathbf{x_{k+1}} = \mathbf{F}\mathbf{x_k} + \mathbf{w_k} \tag{6}$$

We choose a linear motion model for the person with independent motion of the individual dimensions and thus $\mathbf{F}$ is a block-diagonal matrix implementing $x_{k+1} = x_k + \Delta t \cdot \dot{x}_k$ for all components of $\mathbf{x}$. The vector $\mathbf{w_k}$ models the process noise which is sampled from a normal distribution with covariance matrix $\mathbf{Q}$. In each iteration step, the Kalman filter first predicts the system state based on the previous state and the process model and then updates its belief according to the current measurement $\mathbf{z_k}$ and its notion of the sensor's noise. The measurement vector $\mathbf{z}$ holds the bounding box generated by the *Human Localizer* in two-corner parameterization such that $\mathbf{z} = [x_{\min}, y_{\min}, x_{\max}, y_{\max}]^T$. As formulated by [21], the measurement update function links the state to the current measurement as

$$\mathbf{z_k} = \mathbf{H}\mathbf{x_k} + \mathbf{v_k}, \tag{7}$$

where $\mathbf{H}$ is the observation matrix and $\mathbf{v_k}$ is the measurement noise vector.

4. EXPERIMENTAL RESULTS

Datasets. We use the Gait3 database [22] to create training samples for the classification model and to obtain test sequences for evaluating the performance of the tracker. Gait3 is a multimodal database for gait recognition unifying an event-based, an RGB and a thermal camera. It features recordings of humans performing three variations of walking, that is normal and quick walking along with walking while carrying a backpack. For the purpose of creating training samples for the *human* class we only used the normal and quick walking event camera recordings from Gait3 [22]. The database does not provide human ground truth bounding boxes and thus we labeled groups of events manually based on event-frame representations. We recorded the data for the *background* label of the classification training set ourselves by capturing a series of background objects with a DAVIS346 event camera which is the same sensor that was used for creating the Gait3 database. Finally, the data samples in the training dataset are distributed evenly such that half of the samples belong to the *human* label and the other half consists of *background* samples.

Training Details. For training the classification model introduced in subsection 3.2, data augmentation is performed by randomly flipping positions of graph nodes along two axes with probabilities 0.2 and 0.3 as well as scaling them with a random factor between 0.95 and 0.999 [16]. The *Human Classifier* is trained for 14 epochs, using Adam as optimizer, a learning rate of 0.001, Negative Log-Likelihood as loss function, and a batch size of 64. Each *conv*-block of the classification model comprises of a graph convolution layer followed by an ELU activation function and batch normalization. In analogy to [13, 16], we employ a spline-based convolution operator [23] for the convolution layers which have a kernel size of $k = 8$ and output dimensions $M_{out}^{(i)} = (16, 32, 32, 32, 128, 128, 128)$. Graph pooling is effectuated using a max pooling and a global average layer (after *Conv5* and *Conv7* respectively). The *Human Classifier* is implemented with the PyTorch Geometric library [24]. Training samples have been constructed by using a temporal window with $\Delta t = 50ms$ for slicing the event stream and a radius filtering technique for point clouds [25] is applied in order to eliminate noisy outlier events.

Evaluation Metrics. To date, no advanced object tracking benchmark exists for event-based cameras and hence also no corresponding set of standard evaluation metrics. Therefore, we are introducing the following compact set of evaluation metrics in order to quantify the performance of our proposed tracker.

Precision. Inspired by [26], we measure the precision as the average center error (ACE) between the center of the estimated and the ground truth bounding boxes. Moreover, we adopt the representative precision score (RPS) which indicates the percentage of samples for which the estimated location is below a threshold distance of 20 pixels [26] to the ground truth.

Accuracy. We utilize the average overlap score (AOS) for quantifying the tracking accuracy [27] [26]. This is based on the *intersection over union* score (IoU) which relates the area of overlap and union of the estimated and the ground truth bounding box.

Robustness. Similar to [27], we conduct a basic robustness analysis which measures how often the tracker drifts away from the target. A tracking failure is produced once the estimated bounding box has a zero overlap with the ground truth and the robustness is defined as the average number of failures. In contrast to [27], our tracker doesn't require initialization by ground truth and thus we consider all samples to contribute to the metrics.

Evaluating the Influence of a Kalman Filter. Subsequently, we present two versions of the *Human Tracker* and compare their performances on 11 test sequences using the previously introduced evaluation metrics. Firstly, we refer to the tracker that complies with the scheme from Figure 1 as *Human Tracker with Kalman* (HT w/ Kalman). Moreover, we evaluate a second tracker which naively

	HT w/o Kalman				HT w/ Kalman			
Seq.	AOS ↑	ACE ↓	RPS ↑	R ↓	AOS ↑	ACE ↓	RPS ↑	R ↓
1	0.524	16.7	63.0 %	1	0.567	**10.1**	**99.1%**	0
2	0.438	24.2	40.5%	1	0.446	23.1	29.7%	0
3	0.638	11.3	83.8%	0	**0.646**	10.9	97.1%	0
4	0.471	25.8	42.4 %	4	0.417	24.4	25.8%	0
5	0.477	22.6	54.7%	5	0.463	17.9	66.7%	0
6	0.420	26.4	55.8 %	10	0.391	23.5	55.8%	1
7	0.452	23.7	53.9%	5	0.452	12.9	89.5%	0
8	0.417	31.0	40.3%	5	0.374	24.7	38.9%	0
9	0.457	21.9	60.5%	8	0.434	16.8	71.1%	0
10	0.506	22.7	63.4%	7	0.441	15.9	74.6%	2
11	0.331	35.2	23.8 %	9	0.368	27.1	27.4%	0
Total	**0.466**	23.6	52.9%	0.065	0.456	**18.6**	**62.7%**	**0.004**

Table 1. Evaluation results of the Human Tracker (HT) with (w/) and without (w/o) deploying a Kalman filter with respect to the metrics AOS, ACE, RPS and R on given test sequences.

believes in the output generated by the *Human Localizer* without considering knowledge from prior time steps. Therefore, we denote this tracker as *Human Tracker without Kalman* (HT w/o Kalman). We quantify the performance of both trackers sequence by sequence and also compute scores measuring their total performances. It is important to note that the total scores represent the average over all event slices from all test sequences. Hence, in terms of the *robustness* R, the score signifies the risk that the respective tracker drifts away from the target at a given time step. The results of the evaluation are summarized in Table 1 and we will focus on comparing the trackers' total performances.

First of all, the deployment of the Kalman filter significantly improves the *precision* of the tracker. On average the estimated bounding box center is 5 pixels closer to the ground truth when compared to HT w/o Kalman. Moreover, HT w/ Kalman outperforms HT w/o Kalman in terms of total RPS with 62.7% to 52.9%. Also, it can be observed that the Kalman filter significantly increases the tracking robustness R, as the risk of a tracking failure is reduced from 0.065 to 0.004. Still, when comparing the *accuracy* of HT w/ Kalman and HT w/o Kalman, one can see that the AOS shrinks by a margin of 0.01.

A comparison of the success plots of the two trackers, illustrated in Figure 2, sheds more light on the influence of the Kalman filter. The success plot visualizes the percentage of successfully tracked event slices at different IoU-thresholds [26]. At a threshold of $t_{IoU} = 0$, HT w/o Kalman has a lower success rate than HT w/ Kalman which conforms to its lower robustness score. Besides, one can see that HT w/ Kalman dominates HT w/o Kalman until a threshold of $t_{IoU} = 0.45$. For larger thresholds $t_{IoU} \leq 0.45$, HT w/o Kalman has a higher success rate. Thus, deploying the Kalman filter in the tracking framework decreases the number of bounding box estimates with high IoU-scores while at the same time it eliminates predictions exhibiting a small overlap with the ground truth and hence leads to a much higher robustness compared to HT w/o Kalman. A similar observation can be made by examining the trackers' precision plots in Figure 3. According to [26], the precision plot displays the precision for different location error thresholds. For low thresholds with $t_{loc} < 14$, HT w/o Kalman is slightly more precise than HT w/ Kalman while for $t_{loc} > 14$ the opposite holds true. Overall one can obtain that the deployment of the Kalman filter in HT w/ Kalman yields significant improvements in *precision* and *robustness* compared to HT w/o Kalman and comes at a cost of a slightly lowered *accuracy*.

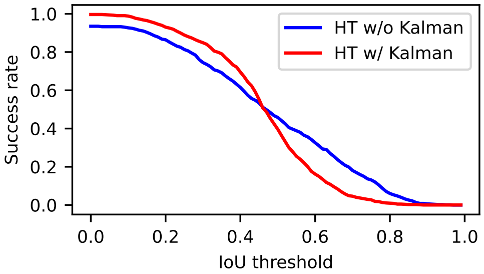

Fig. 2. Success plots [26] of the Human Tracker (HT) with (w/) and without (w/o) Kalman filter.

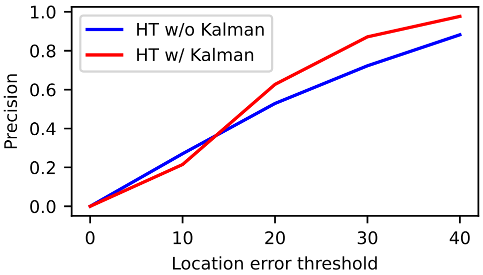

Fig. 3. Precision plots [26] of the Human Tracker (HT) with (w/) and without (w/o) Kalman filter.

5. CONCLUSION

We have demonstrated how human tracking can be realized using a static event-based camera by representing groups of events as spatio-temporal graphs. Thus, our approach makes use of the sparse nature of the event stream and moreover, it doesn't require ground truth data for predicting the human's position and shape. Therefore, we adopt the proven concept of Class Activation Maps (CAM) for our GNN-based classifier and exploit the method's localization capabilities. We deploy a Kalman filter which models the generated bounding box as measurement in order to implement an event-based human tracking framework with high *robustness*. To conclude, we have shown that human tracking is possible using a single event-based camera and we suggest that future research focuses on more advanced scenarios including multiple objects.

6. REFERENCES

[1] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza, "Event-based vision: A survey," *CoRR*, vol. abs/1904.08405, 2019.

[2] Arren Glover and Chiara Bartolozzi, "Robust visual tracking with a freely-moving event camera," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3769–3776.

[3] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara,

and Yiannis Aloimonos, “Event-based moving object detection and tracking,” *CoRR*, vol. abs/1803.04523, 2018.

[4] Bharath Ramesh, Shihao Zhang, Hong Yang, Andrés Ussa, Matthew Ong, Garrick Orchard, and Cheng Xiang, “e-tld: Event-based framework for dynamic object tracking,” *CoRR*, vol. abs/2009.00855, 2020.

[5] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba, “Learning deep features for discriminative localization,” *CoRR*, vol. abs/1512.04150, 2015.

[6] Ewa Piatkowska, Ahmed Nabil Belbachir, Stephan Schraml, and Margrit Gelautz, “Spatiotemporal multiple persons tracking using dynamic vision sensor,” in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012, pp. 35–40.

[7] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020.

[8] 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,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 1695–1704.

[9] Gianluca Scarpellini, Pietro Morerio, and Alessio Del Bue, “Lifting monocular events to 3d human poses,” *CoRR*, vol. abs/2104.10609, 2021.

[10] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci, “Event-based convolutional networks for object detection in neuromorphic cameras,” *ArXiv*, vol. abs/1805.07931, 2018.

[11] Zhuangyi Jiang, Pengfei Xia, Kai Huang, Walter Stechele, Guang Chen, Zhenshan Bing, and Alois Knoll, “Mixed frame-/event-driven fast pedestrian detection,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8332–8338.

[12] Etienne Perot, Pierre de Tournemire, Davide Nitti, Jonathan Masci, and Amos Sironi, “Learning to detect objects with a 1 megapixel event camera,” *CoRR*, vol. abs/2009.13436, 2020.

[13] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos, “Graph-based object classification for neuromorphic vision sensing,” *CoRR*, vol. abs/1908.06648, 2019.

[14] 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 *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 914–923.

[15] Yongjian Deng, Hao Chen, Huiying Chen, and Youfu Li, “EV-VGCNN: A voxel graph CNN for event-based object classification,” *CoRR*, vol. abs/2106.00216, 2021.

[16] Simon Schaefer, Daniel Gehrig, and Davide Scaramuzza, “Aegnn: Asynchronous event-based graph neural networks,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[17] Daniel Gehrig and Davide Scaramuzza, “Pushing the limits of asynchronous graph-based object detection with event cameras,” 2022.

[18] Phillip E. Pope, Soheil Kolouri, Mohammad Rostami, Charles E. Martin, and Heiko Hoffmann, “Explainability methods for graph convolutional neural networks,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10764–10773.

[19] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” 1996, KDD’96, p. 226–231, AAAI Press.

[20] V. Girondel, A. Caplier, and L. Bonnaud, “Real time tracking of multiple persons by kalman filtering and face pursuit for multimedia applications,” in *6th IEEE Southwest Symposium on Image Analysis and Interpretation, 2004.*, 2004, pp. 201–205.

[21] Greg Welch and Gary Bishop, “An introduction to the kalman filter,” Tech. Rep. 95-041, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.

[22] Jean-Luc Dugelay, “Gait3: An event-based, rgb and thermal gait database,” in *BIOSIG 2022, 21st International Conference of the Biometrics Special Interest Group, 14-16 September 2022, Darmstadt, Germany*, IEEE, Ed., Darmstadt, 2022.

[23] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller, “Splinecnn: Fast geometric deep learning with continuous b-spline kernels,” *CoRR*, vol. abs/1711.08920, 2017.

[24] Matthias Fey and Jan Eric Lenssen, “Fast graph representation learning with pytorch geometric,” 2019, cite arxiv:1903.02428.

[25] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun, “Open3d: A modern library for 3d data processing,” *CoRR*, vol. abs/1801.09847, 2018.

[26] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, “Online object tracking: A benchmark,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418.

[27] Matej Kristan et al., “The visual object tracking vot2013 challenge results,” in *2013 IEEE International Conference on Computer Vision Workshops*, 2013, pp. 98–111.