# Neuromorphic Extended Object Tracking and Classification Using Spiking Neural Networks

Craig Hamilton
*School of Engineering and Physical Sciences*
*Heriot-Watt University*
Edinburgh, Scotland
ch40@hw.ac.uk

Yoann Altmann
*School of Engineering and Physical Sciences*
*Heriot-Watt University*
Edinburgh, Scotland
Y.Altmann@hw.ac.uk

***Abstract*—Extended object tracking systems are becoming more relevant for autonomous systems due to new innovative sensor systems and the cost reduction of existing sensors. Data from such sensors can be modelled as sets of spare asynchronous spikes, opening the door to the use of Spiking Neural Networks (SNNs). SNNs offer a more energy-efficient method of processing data when compared to Artificial Neural Networks (ANNs). By leveraging SNNs for extended object tracking, we illustrate in this paper how to create fast and robust tracking systems capable of directly processing data from multiple potential data sources. We also show how to easily enhance tracking systems with the ability to classify or estimate the sizes of objects. As an example, we propose an SNN architecture capable of jointly performing extended object detection, tracking and classification. We compare our architecture to a state-of-the-art Poisson multi-Bernoulli mixture (PMBM) tracker and discuss how SNNs can be applied in a wider set of scenarios.**

## I. Introduction

With the decreasing cost and increasing adoption of fast LiDAR systems for autonomous systems, there is still a critical need for fast, low-complexity object tracking algorithms, and in particular for extended objects. While long-range detection technologies typically produce a single observation per object/target, high-resolution LiDAR or millimetre-wave radars for instance, can generate multiple returns from objects [1]. Similarly, neuromorphic cameras can be used to capture event-like data that can in turn be used for object detection and tracking [2]. In the context of detection and tracking, such information can be used for object recognition/classification. Beyond this, such systems can be used for mapping, navigation or even vital sign monitoring.

In this paper, we investigate the problem of detection, recognition and tracking of extended objects. Most current extended object tracking algorithms operate in one of two modes. The first set of methods split the detection and tracking problems into two separate and subsequent problems. These methods are particularly well adapted when the data quality is high and the extraction of objects is relatively easy, e.g., based on measured signal strengths or using architectures for object detection such as YOLO [3]. Once estimated object positions are extracted, they can be fed to single or multi-object trackers, that often need to be able to handle misdetections and clutter. Although complex trackers (see next paragraph) can be deployed, object tracking in favourable scenarios can often be achieved using variants of the Kalman filter [4], using heuristic or simple rules to speed-up the data allocation problem. One of the main drawbacks of those methods is that the tracking performance is highly dependent on the object extraction method and the ability to limit clutter observations.

The second set of methods, solving jointly the detection and tracking problems, is more suitable for challenging tracking problems, e.g., when the clutter level is high, the signatures of interest are weak. These methods are primarily Bayesian trackers [4], in particular based on random finite sets e.g., the PMBM filter [5], and allow for principled uncertainty management and quantification, from data uncertainties to uncertainties about estimated tracks. While the latest Bayesian filters have shown excellent detection and tracking capabilities, they remain generally expensive computationally due to the data allocation and track storing problem. Moreover, pure model-based approaches can also be limited by the relative simplicity of the clutter, observation and motion models. Recently, data-driven approaches have been used within Bayesian filters to learn motion and observation models [6], [7] but to the best of our knowledge, not yet for extended object tracking.

Spiking neural networks (SNNs) appear as promising architectures to address tracking problems, as they build on the propagation of asynchronous events or spikes and measurements from neuromorphic [8] and radar sensors can be interpreted as event-like. SNNs can learn complex spatio-temporal features and due to the binary (quantised) nature of their signals, they can run rapidly on dedicated neuromorphic hardware. A number of recent studies addressed object detection using SNNs, e.g., [9]–[12]. However those methods handle the object detection, classification and tracking tasks separately.

In this work, we investigate a flexible SNN framework capable of jointly detecting, tracking and classifying extended object measurements from LiDAR and event camera-like sensors. We build on the preliminary work by A. Abdulaziz et al. [8] and extend it by enabling 2D tracking and object classification. This is achieved by introducing different loss functions to allow end-to-end supervised training of a single network. Our investigation aims at assessing whether simple SNNs can perform similarly or better than model-based Bayesian trackers. While the SNN considered here does not

(yet) enable uncertainty quantification, it could in the future using probabilistic SNNs and Bayesian learning [13]–[16]

The remainder of this paper is structured as follows: Section II introduces the spiking neuron models considered and describes the architecture used and associated training initialisation strategy. Section III discusses the tracking problem we considered and compares the performance of the SNN-based tracker to a state-of-the-art extended object tracking algorithm. Section IV finally summarises the contributions of this work and discusses future work.

## II. SNN FOR JOINT DETECTION, CLASSIFICATION AND TRACKING

### A. Spiking neuron models

Although SNNs handle asynchronous events, such networks discretise data over their temporal dimension. For any given time $t$ a spiking neuron $i$ produces a binary output or spike $s_{i,t} \in \{0, 1\}$. In a layered architecture as considered here, each neuron receives as input the signals emitted by neurons from the previous layer and the links between neurons are referred to as synapses. These neurons of the previous layer are referred to as pre-synaptic neurons, whose set is denotes $P_i$, for the post-synaptic neuron $i$. The internal state of a spiking neuron $i$ at any given time $t$ is described by a so-called membrane potential $u_{i,t}$ [17]. Different models exist to describe the dynamics of spiking neurons, including the Spike Response Model (SRM) [18], the Hodkin-Huxley neuron model [19], the Leaky Integrate-and-fire (LIF) neuron model and recurrent spiking neuron model [17].

In this work, we employ two different neuron models, the LIF neuron and the recurrent LIF [17]. The LIF model mimics the behaviours found in RC circuits and the membrane potential can be described as

$$u_{i,t} = \beta_i u_{i,t-1} + \sum_{j \in P_i} w_{j,i} s_{j,t} - \theta_i s_{i,t-1}. \quad (1)$$

The membrane potential consists of the weighted sum of the input synapses and incorporates a leakage term analogous to an RC circuit [9]. Here $\beta_i$ represents the decay rate of the membrane potential for the neuron $i$, $w_{j,i}$ denotes the synaptic weight of the link between the pre-synaptic neuron $j$ to the post-synaptic neuron $i$. Both the decay rate $\beta_i$ and the synaptic weights $w_{j,i}$ are learnable parameters of the LIF neuron Model. When the membrane potential crosses a trainable threshold value $\theta_i$, the LIF neuron "fires" and generates a spike, i.e., $s_{i,t} = 1$. The neuron may also be reset by subtracting $\theta_i$ from the membrane potential ensuring regulated spiking activity [20] (see last term in (1)). Without this reset mechanism, the spiking neuron continues to fire until membrane potential decreases below the threshold $\theta_i$.

The recurrent LIF neuron model operates similarly to the LIF neuron with the addition of the output spikes of the neuron being fed back in as input the neuron's own input. The recurrent LIF neuron can be expressed as

$$u_{i,t} = \beta_i u_{i,t-1} + \sum_{i \in P_i} w_{j,i} s_{j,t} + w_i s_{i,t-1} - \theta_i s_{i,t-1}, \quad (2)$$

where the recurrent weight $w_i$ is an additional trainable parameter.

### B. Network Architecture

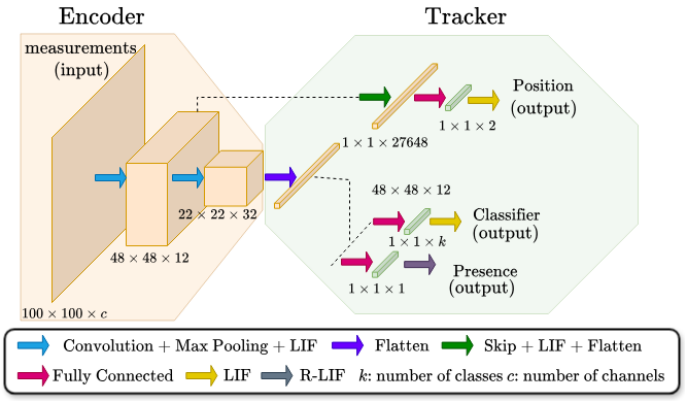


Fig. 1. Diagram of the proposed SNN-based tracking system. Arrows of different colours indicate the types of neural network layers, as shown in the colour-coding in the legend.

The proposed architecture, depicted in Fig. 1 is tailored for the task of detection, tracking and classification of clusters or swarms of events. The training processing will be detailed in Section II-C but we present here briefly the architecture and associated training loss. The network consists of an encoder (left) with two sets of Convolution -Max Pool – LIF blocks and a tracking head with three separate sets of fully connected output neurons. The first is a pair of non-spiking neurons dedicated to determining the position of the cluster within the field of view, the second group is a set of spiking neurons used to determine the object class, and finally a single recurrent-spiking neuron estimates the presence of an object within the field of view. It is interesting to note that the pair of non-spiking position neurons are connected to the output of the first convolution layer via a non-spiking LIF layer, skipping the Max Pool and second joined block. From preliminary runs, we observed that this configuration is more efficient than connecting to the output of the second convolution layer (for the problem we considered). The input of the encoder can be configured to have several input channels $c$, such as RGB images or events from a neuromorphic camera, where events are polarised.

For any given time $t$, the network loss function is expressed as

$$\begin{aligned} \mathcal{L}(t) &= BCE\left(u_{out_{1,t}}, l_t\right) + CE\left(u_{out_{2,t}}, c_t\right) \\ &+ \delta\left(l_t - 1\right) MSE\left(u_{out_{3,t}}, p_t\right), \end{aligned} \quad (3)$$

where $\delta(.)$ denotes the Kronecker delta function. The Binary Cross-Entropy (BCE) loss is calculated between the spiking outputs membrane potential $u_{out_{1,t}}$ and the actual binary presence label $l_t$. The Cross-Entropy (CE) loss is evaluated between the spiking classification outputs membrane $u_{out_{2,t}}$ and the one-hot encoded target classification $c_t$ (assumed to be the same over time for a given training sample/sequence). The

last term in the loss computes the mean squared error (MSE) between the ground truth and estimated object positions, denoted $p_t$ and $u_{out_{3,t}}$, respectively. The delta function acts as a mask and ensures that the MSE term is only included when $l_t = 1$, i.e. when the cluster is present in the scene. The BCE and CE losses are constantly calculated at each time $t$. The MSE loss quantifies the difference between the non-spiking output membrane potential and the centroid position of events.

### C. Network description and training process

The network structure used is that described in the previous Section II-B and Fig. 1. To accelerate the training process, the 12 filters of the first convolutional layer are initialised using Gabor filters [21], primarily used for texture and feature extraction. They are also believed to approximate the response of simple cells in the mammalian visual cortex [21] [22] to oriented bars and from preliminary runs, we observed that they help convergence with small training sets. This observation is in line with previous work showing it can improve the performance and accelerate the convergence of conventional convolutional neural networks [23]. The 32 filters of the second layer are initialised randomly.

The network input at each time step t consists of a matrix of events of size 100 x 100, which corresponds to discretising the spatial field-of-view into $100^2$ cells. Training of the SNN employs back-propagation through time (BPTT) [24]. This method works backwards from the final output of the network, the gradient is propagated backwards through the network. This way the computation of the gradients of the SNN closely mirrors the method used for recurrent neural networks (RNNs) by iterative use of the chain rule. The thresholding of the membrane potential results in a non-differentiable loss akin to a Heaviside step function. To overcome the so-called "dead neuron problem" and the non-differential nature of the spiking functions, a surrogate gradient method is employed, whereby the Heaviside step function is replaced with a continuous and differentiable function during the backward pass (a sigmoid function here). BPTT is run with a batch size of 48 as larger batches did not have a significant effect on training speed or model accuracy for our experiments. Adam [25] was selected as optimiser and the learning rate was decreased from $0.5e^2$ after 20 epochs, by 5% at the end of each epoch until the end of the training (50 epochs in total).

## III. Experiments

### A. Data Generation

To evaluate our model, synthetic data set was created, illustrating 2D gaussian-like objects appearing and disappearing in the field of view (at most one object per sequence). Three main scenarios have been investigated, departing only by the expected number of observations per object at each time stamp, ranging from 5 (hard scenario) to 50 (easy scenario). Three sets of 10,080 samples each were created as part of the training set and a further 144 samples were generated to test and evaluate the model. These samples were created using a linear constant velocity motion and measurement model as detailed in [26]. This method of data generation favours the tracker we have compared our model to. Each sample consists of measurements spanning over 50 timesteps. Each timestep contains an expected number of clutter measurements defined by the clutter rate $\lambda_c$. The "birth" timestep is randomly generated to be in the first half of the sequence and the "death" timestep in the second half of the time series. Upon reaching the birth timestep a random starting point and velocity are generated. For each subsequent timestep, the motion model modifies the position of the object, until the "death" timestep is reached, where it is subsequently removed from the field of view. The track generated by the motion model describes the centroid of the extended object and is used as our ground truth position, times of birth and death can be used to determine presence during the sequence. The measurement model takes each of these ground truth positions and generates a set number of measurements within a Gaussian cluster around the centroid, with covariance $\Sigma_r$ if the object belongs to class $r \in \{1, \ldots, k\}$. For our dataset, the field of view was set to a range of $\{-1000, 1000\}$, the clutter rate $\lambda_c = 5$ and we used $k = 3$ classes, which corresponds to three object sizes/shapes. More precisely, we used for $\Sigma_r$ diagonal covariance variances whose diagonals are $\{(50^2, 50^2), (75^2, 75^2), (50^2, 75^2)\}$, respectively.

### B. Results

To measure the performance of the network the same data used during the testing process was also passed through an optimised extended target Poisson multi-Bernoulli mixture (PMBM) tracker [27] [28]. The PMBM model was optimised by cross-validation to ensure a reasonable trade off between the Precision and False Positive Rate.To emulate the classification abilities of the SNN we attempted to use the provided extents of the PBMB tracker to assess if those could be clustered for classification purposes. However, this did not work due to a significant amount of noise in the estimated extents. Conversely, the SNN does not directly provide size or extent only a classification of group size. As such classification performance has been omitted from the results for the PBMB tracker, only presence and position metrics can be provided. Nonetheless, the SNN tracker was able to achieve a classification accuracy of 84.03% for the 5-event dataset, 97.62% for the 10-event dataset and 100% for the 50-event dataset.

Table I provides a quantitative comparison of the SNN and PMBM trackers. These trackers have been evaluated on True Positive Rate (TPR), False Positive Rate (FPR), accuracy, precision, and F1 score for presence. For the position errors, the MSE has been provided, alongside its 5th and 95th quantiles. The PMBM tracker shows very high performance across all metrics when a large number (50) measurements are available for the target. The PMBM tracker presents a TPR of 98.43%, a low FPR of 0.24%, an accuracy of 98.97%, a precision of 99.71% and an F1 score of 99.05 %. However, for the same data, the SNN-based tracker is able to marginally

| | | Detection | | | | | Localisation |
|---|---|---|---|---|---|---|---|
| | | TPR | FPR | Acc. | Prec. | F1 | MSE |
| 5 events | SNN | **96.45**% | 0.79% | **98.06**% | 99.05% | **97.67**% | 0.020 (0.0044, 0.7680) |
| | PMBM | 93.71% | **0.13**% | 97.18% | **99.77**% | 96.48% | **0.014** (0.0002, 1.0000) |
| 10 events | SNN | **97.38**% | 1.19% | **98.29**% | 99.17% | **98.22**% | **0.013** (0.0022, 0.3755) |
| | PMBM | 94.23% | **0.00**% | 97.25% | **100.00**% | 96.76% | 0.016 (0.0002, 1.0000) |
| 50 events | SNN | 98.22% | 0.80% | 98.60% | 99.25% | 98.71% | **0.007** (0.0011, 0.2482) |
| | PMBM | **98.43**% | **0.24**% | **98.97**% | **99.71**% | **99 .05**% | 0.011 (0.0001, 0.6930) |

TABLE I

COMPARATIVE PERFORMANCE METRICS OF THE SNN AND PMBM TRACKERS EVALUATED OVER A NUMBER OF EVENTS. METRICS ASSESSED INCLUDE THE TRUE POSITIVE RATE (TPR), FALSE POSITIVE RATE (FPR), ACCURACY (ACC), PRECISION (PREC) AND F1 SCORE FOR DETECTION. MEAN SQUARED ERROR (MSE) HAS BEEN PROVIDED WITH LOCALISATION ALONGSIDE (5TH/95TH) QUANTILES.

provide a more accurate track with an MSE of 0.007 compared to the PMBMs MSE of 0.01. Considering the 5th and 95th quantiles of the MSE helps differentiating the two methods. In all instances, the 5th quantiles for the PMBM tracker are lower than the SNN quantiles, but the 95th quantiles are larger. This shows that whilst the SNN tracker may not perform locally as favorably as a well-calibrated PMBM tracker it does not lead to excessively larger positioning errors. The SNN-based tracker is not significantly far behind and keeps within 1% point of the PMBM tracker for all detection criteria (bottom rows of Talbe I).

In the scenario where a small number of measurements (5) is available for the target at each timestamp, the SNN becomes more competitive. For the 5-measurement scenario, the SNN tracker has a TPR of 96.45%, an FPR of 0.79%, an accuracy of 98.06%, a precision of 99.05% an F1 score of 97.67% and an MSE of 0.02. Whilst the SNN only outperforms the PBMB in terms of TPR, accuracy and F1 score, it is not significantly behind in the remaining metrics and remains again within 1% point. This shows that the SNN tracker can perform similarly if not better than the PMBM tracker where measurements are limited for presence detection, and it is still able to classify measurements with an 84.03% accuracy. We also again see the same behaviour with the position estimation where the SNN is more robust and does not rely on the calibration. This better reflects real-world scenarios where a limited number of measurements may be received amidst the clutter and noise of the sensory data, and it may perform better with more complex scenarios.

Fig. 2 shows a reconstruction of the position estimate of the two trackers. Whilst the SNN-based position may appear noisier at times these spikes in noise are also where the PMBM tracker completely loses the object and is no longer able to provide a position estimate demonstrating the more robust nature of the SNN. The position estimate of the SNN tracker is directly linked to the membrane potential of the output neuron, this results in an oscillating behaviour as the pre-synaptic neurons fire due to high local activity causing a jump in membrane potential before the membrane can naturally decay back and follow the movement of the target.

Finally, Fig. 3 shows another example of the position estimation. Whilst both results are noisy and neither closely tracks the measurement, the SNN-tracker can more closely follow the general movements. Both the SNN and PMBM

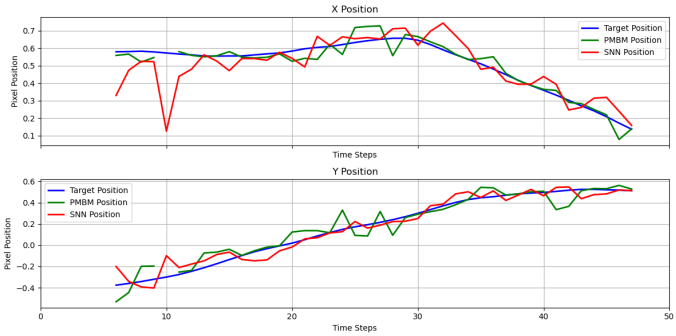


Fig. 2. Reconstructed position results from the PMBM tracker (Green) and the SNN tracker (Red). Ground truth is shown in Blue. The top panel illustrates the X-axis position estimation whilst the bottom illustrates the Y-axis position estimation.

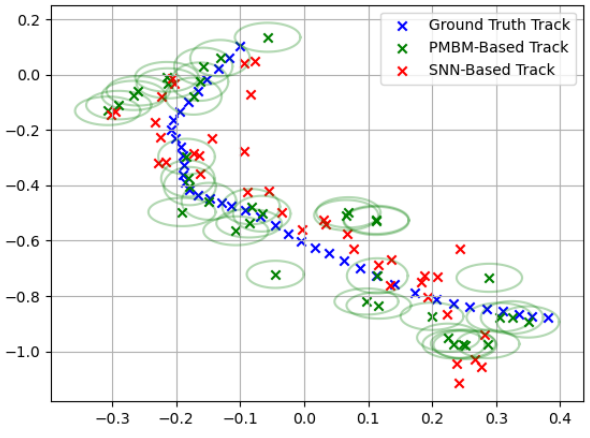


Fig. 3. 2D view of the reconstruction results of the PMBM tracker (Green) and the SNN tracker (Red). Ground truth is shown in Blue. Also shown is the extent of the PMBM tracker.

tracker suffer from a poor initialisation, but the SNN can respond more quickly and correct itself as the measurements move down the through the field of view.

Quantitative comparison of the computational cost of the two methods is challenging as it depends on the implementation adopted. Here, the SNN was trained and run using a single NVIDIA A5000 (24GB) GPU. The full 50-epoch training took 1.25 hours to complete - this increased by only

15 minutes when training was performed on a consumer-grade NVIDIA RTX 3080 (10GB) GPU. A single pass of the 144 elements test set took a little over 1.8 seconds irrespective of the number of measurements in each sample or timestep. The PMBM tracker was run on an AMD Ryzen 7 5800H CPU. The time taken to run the PMBM tracker varied with the number of measurements in each sample ranging from 0.4 seconds up to 2.3 seconds. However, it is worth noting that the SNN was implemented using the optimised snnTorch library [9], whilst the PBMB tracker was adapted from code found in [27] running in MATLAB and may not be fully optimised. Whilst these results are not directly comparable, it does highlight the better scaling of the SNN.

## IV. Conclusion

In this paper, we presented a Spiking Neural Network (SNN) based joint extended object tracking and classification system. SNNs and compare their performance against a state-of-the-art Poisson multi-Bernoulli mixture (PMBM) tracker. The two tracking systems were compared using synthetic datasets consisting of sequences of 50 timestamps with different detection levels. Overall, the SNN tracker provides marginally better results when compared to the PMBM tracker with an average presence accuracy of 98.31%. In contrast, the PMBM tracker has an average presence accuracy of 97.80%. PMBM provides better position MSEs that SNN, but its position errors can be quite large. Whilst the SNN does not significantly outperform the PMBM tracker, it demonstrated a more robust response to a wide variety of input scenarios. This behaviour makes it more suitable for use in more complex scenarios where there may not be enough data or where the ability to discriminate the object from non-visual data is beneficial. Larger SNNs may be able to discriminate between objects with more complex geometry as well as providing tracking abilities where other trackers may struggle. Future work should include the development of Bayesian inference methods to train tracking SNNs as Bayesian networks, as they would then be able to also provide uncertainties about the network outputs.

## References


[1] M. Baerveldt, A. G. Hem, and E. F. Brekke, "Comparing Multiple Extended Object Tracking with Point Based Multi Object Tracking for LiDAR in a Maritime Context," *Journal of Physics: Conference Series*, vol. 2618, p. 012011, Oct. 2023.

[2] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos, "Event-based moving object detection and tracking," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, 2018.

[3] M. Saada, C. Kouppas, B. Li, and Q. Meng, "A multi-object tracker using dynamic Bayesian networks and a residual neural network based similarity estimator," *Computer Vision and Image Understanding*, vol. 225, p. 103569, Dec. 2022.

[4] K. Bell, T. Corwin, L. Stone, and R. Streit, "Bayesian multiple target tracking, second edition," 2013.

[5] K. Granstrom, M. Fatemi, and L. Svensson, "Poisson multi-bernoulli mixture conjugate prior for multiple extended target filtering," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, p. 208–225, Feb. 2020.

[6] R. G. Krishnan, U. Shalit, and D. Sontag, "Deep kalman filters," 2015.

[7] B. Horvath, A. Kratsios, Y. Limmer, and X. Yang, "Deep kalman filters can filter," 2024.

[8] A. Abdulaziz, R. Hegedus, F. Macdonald, S. McLaughlin, and Y. Altmann, "Investigation of Spiking Neural Networks for Joint Detection and Tracking," in *2024 32nd European Signal Processing Conference (EUSIPCO)*, (Lyon, France), pp. 1681–1685, IEEE, Aug. 2024.

[9] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training Spiking Neural Networks Using Lessons From Deep Learning," *Proceedings of the IEEE*, vol. 111, pp. 1016–1054, Sept. 2023.

[10] R. Xiao, H. Tang, Y. Ma, R. Yan, and G. Orchard, "An Event-Driven Categorization Model for AER Image Sensors Using Multispike Encoding and Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, pp. 3649–3657, Sept. 2020.

[11] J. Lu, J. Dong, R. Yan, and H. Tang, "An Event-based Categorization Model Using Spatio-temporal Features in a Spiking Neural Network," in *2020 12th International Conference on Advanced Computational Intelligence (ICACI)*, (Dali, China), pp. 385–390, IEEE, Aug. 2020.

[12] R. Ghosh, A. Gupta, S. Tang, A. Soares, and N. Thakor, "Spatiotemporal Feature Learning for Event-Based Vision," Mar. 2019. arXiv:1903.06923 [cs].

[13] H. Jang, O. Simeone, B. Gardner, and A. Gruning, "An introduction to probabilistic spiking neural networks: Probabilistic models, learning rules, and applications," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 64–77, 2019.

[14] H. Jang, N. Skatchkovsky, and O. Simeone, "Bisnn: training spiking neural networks with binary weights via bayesian learning," in *2021 IEEE Data Science and Learning Workshop (DSLW)*, pp. 1–6, IEEE, 2021.

[15] N. Skatchkovsky, H. Jang, and O. Simeone, "Bayesian continual learning via spiking neural networks," *Frontiers in Computational Neuroscience*, vol. 16, p. 1037976, 2022.

[16] D. Shen, D. Yao, S. McLaughlin, and Y. Altmann, "Probabilistic spiking neural networks training with expectation-propagation," in *2023 IEEE 9th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 41–45, IEEE, 2023.

[17] W. Gerstner and W. M. Kistler, *Spiking Neuron Models Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[18] W. Gerstner, "Time structure of the activity in neural network models," *Phys. Rev. E*, vol. 51, pp. 738–758, Jan 1995.

[19] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *Bulletin of Mathematical Biology*, vol. 52, pp. 25–71, Jan. 1990.

[20] M. Gehrig, S. B. Shrestha, D. Mouritzen, and D. Scaramuzza, "Event-based angular velocity regression with spiking networks," *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[21] J. G. Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *Journal of the Optical Society of America A*, vol. 2, p. 1160, July 1985.

[22] S. Marĉelja, "Mathematical description of the responses of simple cortical cells*," *Journal of the Optical Society of America*, vol. 70, p. 1297, Nov. 1980.

[23] A. K. Jain, N. K. Ratha, and S. Lakshmanan, "Object detection using gabor filters," *Pattern Recognition*, vol. 30, pp. 295–309, Feb. 1997.

[24] D. Huh and T. J. Sejnowski, "Gradient descent for spiking neural networks," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, pp. 1–15, 12 2014.

[26] K. Granström, M. Baum, and S. Reuter, "Extended Object Tracking: Introduction, Overview, and Applications," *Journal of Advances in Information Fusion*, 2017.

[27] Y. Xia, K. Granström, L. Svensson, Ángel F. García-Fernández, and J. L. Williams, "Extended target poisson multi-bernoulli mixture trackers based on sets of trajectories," 2019.

[28] K. Granström, M. Fatemi, and L. Svensson, "Gamma gaussian inverse-wishart poisson multi-bernoulli filter for extended target tracking," in *2016 19th International Conference on Information Fusion (FUSION)*, pp. 893–900, 2016.