Unsupervised Learning of Spatiotemporal Feature Representations for Event-Based Vision

Rohan Ghosh, *Student Member, IEEE,* Anupam K. Gupta, Siyi Tang,
and Nitish V. Thakor, *Life Fellow, IEEE*

Abstract—Unlike conventional frame-based sensors, event-based visual sensors output information through spikes at a high temporal resolution. By only encoding changes in pixel intensity, they showcase a low-power consuming, low-latency approach to visual information sensing. To use this information for higher sensory tasks like object recognition and tracking, an essential simplification step is feature extraction. An ideal feature descriptor must be robust to changes involving (i) local transformations and (ii) re-appearances of a local event pattern. To that end, we propose a novel unsupervised spatiotemporal feature descriptor learning algorithm based on slow feature analysis (SFA). Using SFA, smoothly changing linear projections are learnt which are robust to local visual transformations. A feature point tracking task is chosen to evaluate the effectiveness of the feature representations. We found that our method performed significantly better than the other evaluated methods, as shown by the percentage of active trackers (over 80%) that stayed with the feature point even after 1.5 seconds of fast feature motion involving translational, scaling and pose changes. The adaptability of the method to different visual scenarios makes it a suitable precursor to high-speed tracking and recognition applications.

Index Terms—Event-based vision, Feature extraction, Slow feature analysis, Feature point tracking

1 Introduction

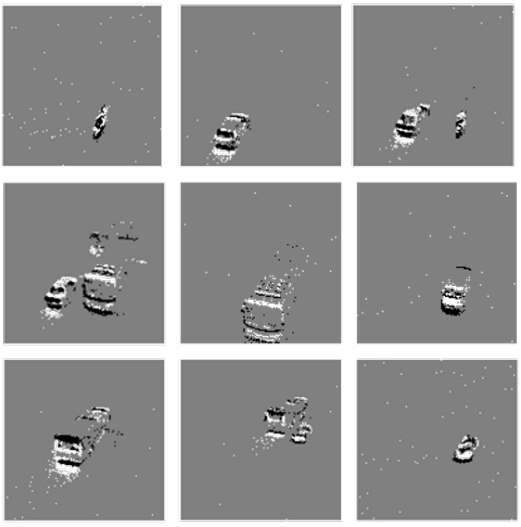


Fig. 1. Examples of binned event-data from the traffic dataset recorded for our point tracking experiments. Pixels of event-based cameras asynchronously respond to visual intensity changes at a temporal resolution of 1 μs. Spikes are produced when a pixel accumulates light intensity above a predetermined threshold. Spike polarity (white or black) indicates whether the change is an increase or decrease in intensity. Each image shown here contains 1000 spike-events recorded from an event-based vision sensor, facing the road from above.

Feature extraction has been the cornerstone of many computer vision algorithms. It is a common practice to compress the high dimensional information contained in digital images and videos into low dimensional features. This process aids in reducing the computational burden on subsequent algorithmic modules by only channelling relevant and interesting information. Feature descriptors obtained from local image regions around *interest points* (or feature points, e.g. [1]) act as robust transformation-invariant markers. Extraction and matching of feature descriptors are fundamental to applications such as simultaneous localization and mapping (SLAM) [2], depth estimation from stereo images [3], object recognition [4], motion segmentation [5], object tracking [6] and action recognition [7].

For the emerging field of event-based vision, robust feature extraction algorithms are needed. Event-based sensors [8], [9], [10] are primarily characterized by-

- Asynchrony: Each pixel responds in the form of *spike-events*, only when the pixel intensity changes by a predetermined threshold. The spike times are accurate to $\sim 10\mu$s, enabling almost continuous visual information acquisition.
- Low-power consumption: Pixel intensity changes usually only occur at object edges, thereby, limiting the number of spike-events.
- Low latency: Each spike is communicated almost as soon as it occurs (only tens of μs delay).

Therefore, the data generated from event-based sensors are inherently spatiotemporal in the form of sparsely located spike-events. Intensity changes at a pixel can occur either due to global illumination changes or motion of edges. If one assumes that global scene illumination varies only gradually, then most spike-events would occur at the moving edges. Figure 1 shows example images obtained by collecting hundreds of spike-events, from a static dynamic vision sensor (DVS [1]) facing a road. Feature learning and extraction with only dynamic edge information is a challenging problem. However, the high temporal resolution

1. The Dynamic Vision Sensor has a 128x128 spatial resolution, and a 1 μs temporal resolution.

associated with each spike-event, enhances the richness of information in a local spike-event pattern.

Local feature descriptors typically consider local image regions which have properties that distinguish them clearly from their neighbors [11]. Thus, often these desirable properties emerge from a non-uniform pixel intensity structure. Interesting and informative image regions usually form around spatial edges created by local intensity differences. Since event-based sensors only respond to intensity changes, the patterns present in the spikes are naturally indicative of interesting features. Furthermore, their high temporal resolution can be used to add more detail to potential feature descriptors.

Despite the above positives, extracting meaningful features from event data poses several challenges. The spike times can often be inaccurate and plagued by noise, decreasing the *effective* temporal resolution [12]. False spike-events may be generated, caused by shot noise fluctuating the light intensities at the pixel photoreceptors [8]. The integrate-and-fire nature of neurons often leads to cases where the pixel intensities fall short of the threshold required to spike. These may be eventually triggered as noisy spikes and need to be filtered [13]. Furthermore event patterns are rarely consistent for multiple inputs of the same stimulus. Hence, feature estimation methods need to be robust to noise and variation.

An important characteristic of any feature estimation algorithm is the set of criteria that decides feature selection. In [14], a technique was proposed to predict event patterns using unsupervised echo-state networks, that estimates features as discrete classes or clusters. The criterion for feature learning was to favor echo-state networks which better predict how an event pattern unfolds in time. Recently, a feature estimation framework comprising a hierarchical approach to encoding time-surface information was described in [15]. The feature selection criterion prioritized features which occurred more frequently throughout the data. In [16] hierarchies of spiking neural networks were used to learn and extract temporally correlated features.

Although the estimated features in the previously mentioned methods act as useful markers to identify prominent spatiotemporal patterns, they were not trained for robustness to local transformations. Good feature representations must show minimal changes in their feature values when an input stimulus gradually changes in response to visual transformation [1]. This property helps preserve the identity of a feature point within the computed feature representations. Most common examples of such transformations include scaling (from going away/towards the camera), translation, rotation and change of pose. We first note that any event-based input sequence would typically contain multiple feature points transforming smoothly in time while following different trajectories. Therefore, simply constraining feature representations to be similar across reappearances of feature points would be a meaningful step towards achieving transformation invariance. Such an approach can be categorized under a *track and learn* technique, which has found use in other computer vision problems, and is also used in this paper.

In frame-based vision, there are several algorithms which use similar motivation to the track and learn strategy for learning feature representations [17], [18], [19], [20]. Unsupervised convolutional neural network based feature representations were learnt solely from tracked pairs of image regions across videos [17]. A Siamese network architecture [18] was trained to ensure similar responses to consecutive tracked instances of objects. A tracking-learning-detection (TLD) framework was proposed in [19], in which the tracker learns from matched pairs of tracked objects over consecutive frames. Similar to TLD, a tracking-learning-matching technique was also proposed (TLM, [20]), which is TLD's analogue for local feature points.

The contributions of this work are three-fold. First, a novel feature learning algorithm is described, which prioritizes smooth but informative representations. We use the track and learn approach to obtain feature representations which are (i) robust to slight spatiotemporal perturbations of event location, or noise and (ii) slowly changing when an event-pattern re-appears in the spatiotemporal vicinity of its earlier appearance. Specifically, slow feature analysis [21] is used to arrive at the optimal weight matrices. Second, a feature point tracking algorithm is proposed to evaluate the effectiveness of trained features. This helps assess whether the extracted feature representations are spatiotemporally smooth while preserving the identity of an event-pattern. Third, a dataset consisting of event-data recorded from a traffic scene, containing manually annotated trajectories of feature points is provided, which forms the ground truth for the feature point tracker.

This paper is organized as follows. Section 2 presents the theoretical details of the feature extraction method along with the proposed point tracking algorithm. Section 3 provides details about the experiments and analyses of our algorithm's performance. Section 4 analyzes our approach in detail and gives a discussion about our results and their implications. In section Section 5 we provide concluding remarks about our finds.

2 METHODS

2.1 Background

Spike-events generated from an event-based camera can be written as

$$E = \{e_i\}_{i=1}^{N} \quad \text{where} \quad e_i = (x_i, y_i, t_i, p_i). \tag{1}$$

Each spike-event is described by its spatial location (x_i, y_i), time of occurrence t_i and polarity p_i. We also define a spatiotemporal region around each spike as a box neighbourhood centered at a spatiotemporal location as

$$\mathcal{BN}_E(x, y, t, \Delta x, \Delta y, \Delta t) = \left\{ e_i \in E \middle| \begin{matrix} x_i \in \left(x - \frac{\Delta x}{2}, x + \frac{\Delta x}{2}\right) \\ y_i \in \left(y - \frac{\Delta y}{2}, y + \frac{\Delta y}{2}\right) \\ t_i \in \left(t - \frac{\Delta t}{2}, t + \frac{\Delta t}{2}\right) \end{matrix} \right\}. \tag{2}$$

Eqn. 2 describes a box neighbourhood, which is a set of events in 3D (x, y, t) space enclosed within a box with size units specified as $(\text{pixels} \times \text{pixels} \times \mu s)$. For the remainder of the paper, extracted feature vectors around any event will only be derived from the events within its box neighbourhood.

2.2 Method Summary

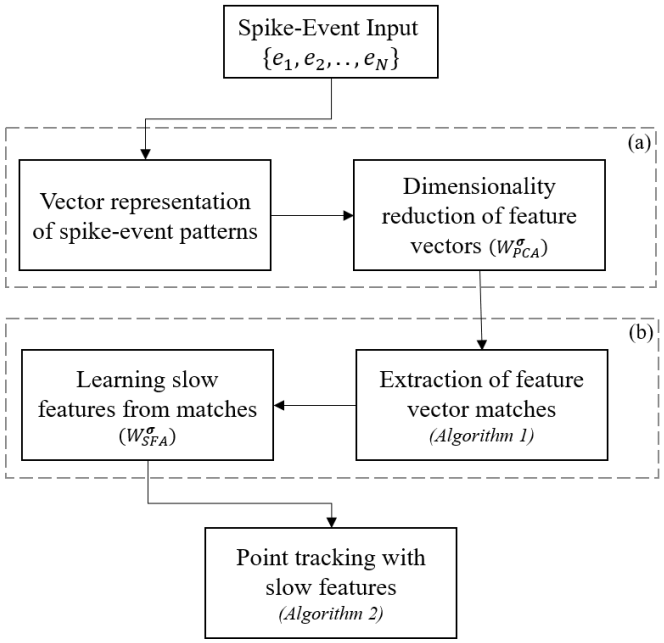


Fig. 2. A flow chart summary of the methodology proposed in this paper, including the feature match extraction and point tracking algorithms. The approach can be divided into two parts which are further explained in the text below.

Fig. 2 presents a flow chart of the steps involved in our proposed methodology. Each step is elaborated in a later section having the same name. The unsupervised feature learning method can be roughly divided into two parts, as demonstrated in the figure. We summarize each part as follows.

1) **Part (a)**: Here, an initial feature vector representation is generated for each event's box-neighbourhood. It simply consists of vectorizing the box neighbourhood while preserving the spatiotemporal topology. This culminates in pruning the feature vector to only account for the most informative components, generating a smaller, more compact feature representation. The features essentially take the form of simple linear projections on a smoothed spike-event array.
2) **Part (b)**: Here the feature representations are further enhanced by obtaining more stable linear projections using the track and learn approach. First, using the feature representations obtained in part (a), pairs of feature matches are generated throughout the data through tracking each feature point (Algorithm 1). Subsequently, slow feature analysis is performed to learn more robust projections which vary less within a tracked pair.

Once the final SFA based features are obtained, they are used in the point tracking algorithm (Algorithm 2).

2.3 Vector representation of spike-event patterns

Let the set of spike events $E = \{e_1, e_2, .., e_N\}$ consist of all events in a training dataset. The objective is to have a feature representation of spatiotemporal data using the distribution of spikes around each event e_i.

Initially, only events in a box neighborhood of size $(a \times a \times \delta t)$ centered about $e_i = (x_i, y_i, t_i)$ are considered. This region is denoted by the set $\mathcal{BN}_E(x_i, y_i, t_i, \Delta x, \Delta y, \Delta T)$. This cuboidal volume is partitioned into smaller box neighborhoods (voxels) of size $(a/p \times a/p \times \Delta T/M)$, where M denotes the number of temporal partitions and p the number of spatial partitions.

Due to the low spatial resolution of the DVS, the number of spatial partitions are increased to its maximum value, by setting $p = a$. Each voxel obtained is of dimension $(1 \times 1 \times \Delta T/M)$. A 3D matrix $\mathbf{C}_i \in \mathbb{R}^{(a \times a \times M)}$ is used to store the spike count at each voxel. The elements of $\mathbf{C}_i$ are estimated as

$$\begin{aligned}\mathbf{C}_i(m,n,p) \quad &= \Big|\mathcal{BN}_E\left(e_i(x) - \tfrac{a}{2} + m, e_i(y) - \tfrac{a}{2}\right. \\ &\quad \left. + n, e_i(t) - \tfrac{\Delta T}{2}, 0, 0, \tfrac{\Delta T}{M}\right)\Big| \end{aligned} \tag{3}$$

where $|\cdot|$ denotes the set cardinality operator. We denote the vectorized form of $\mathbf{C}_i$ by the one dimensional feature vector $\mathbf{c}_i \in \mathbb{R}^{(1 \times p^2 M)}$. For simplicity of notation, let $d = p^2 M$ be the dimensionality of each $\mathbf{c}_i$ vector. Note that for our experiments d typically takes on large values $(500 - 5000)$. In order to avoid a large feature vector dimension while retaining spike timing details, a dimensionality reduction step is needed. The following section describes this step in detail.

2.4 Dimensionality reduction of feature vectors

To aid with dimensionality reduction, the spike count matrix, $\mathbf{C}_i$, is smoothed with a 3D Gaussian kernel,

$$\mathbf{G}_{\sigma_x,\sigma_y,\sigma_t}(x,y,t) = g_x(x,\sigma_x) \times g_y(y,\sigma_y) \times g_t(t,\sigma_t) \tag{4}$$

where $g_x = e^{-x^2/\sigma_x^2}$, $g_y = e^{-y^2/\sigma_y^2}$ and $g_t = e^{-t^2/\sigma_t^2}$ are the one dimensional Gaussian kernel functions. The choice of a Gaussian smoothing kernel for dimensionality reduction is elaborated later in section 4. Variance parameters of the Gaussian kernels are compactly represented in vector notation as

$$\boldsymbol{\sigma} = [\sigma_x, \sigma_y, \sigma_t]. \tag{5}$$

A smoothed spike count matrix is computed by convolving it with the appropriate Gaussian kernel

$$\mathbf{C}_i^{\boldsymbol{\sigma}} = \mathbf{C}_i * \mathbf{G}_{\boldsymbol{\sigma}} \tag{6}$$

where $*$ denotes the convolution operator. $\mathbf{C}_i^{\boldsymbol{\sigma}}$ is vectorized to obtain the feature vector, $\mathbf{c}_i^{\boldsymbol{\sigma}}$. A Gaussian smoothing step allows $\mathbf{c}_i^{\boldsymbol{\sigma}}$ to retain some topological properties of the original space. Fig. 3 illustrates that this process desparsifies (compresses) the original feature vector. Large smoothing variance parameters lead to a loss of temporal and spatial information due to over-smoothing. In contrast small variance parameters retain temporal and spatial information, but generate a sparse feature vector. For our experiments we appropriately select the parameter values as $\sigma_x = \sigma_y = \sigma_t = 3$. The significance of this parameter is elaborated in section 4.

Each event e_i generates a corresponding feature vector $\mathbf{c}_i^{\boldsymbol{\sigma}}$. The set of extracted feature vectors from the training data is

$$\{\mathbf{c}_1^{\boldsymbol{\sigma}}, \mathbf{c}_2^{\boldsymbol{\sigma}}, ..., \mathbf{c}_N^{\boldsymbol{\sigma}}\}. \tag{7}$$

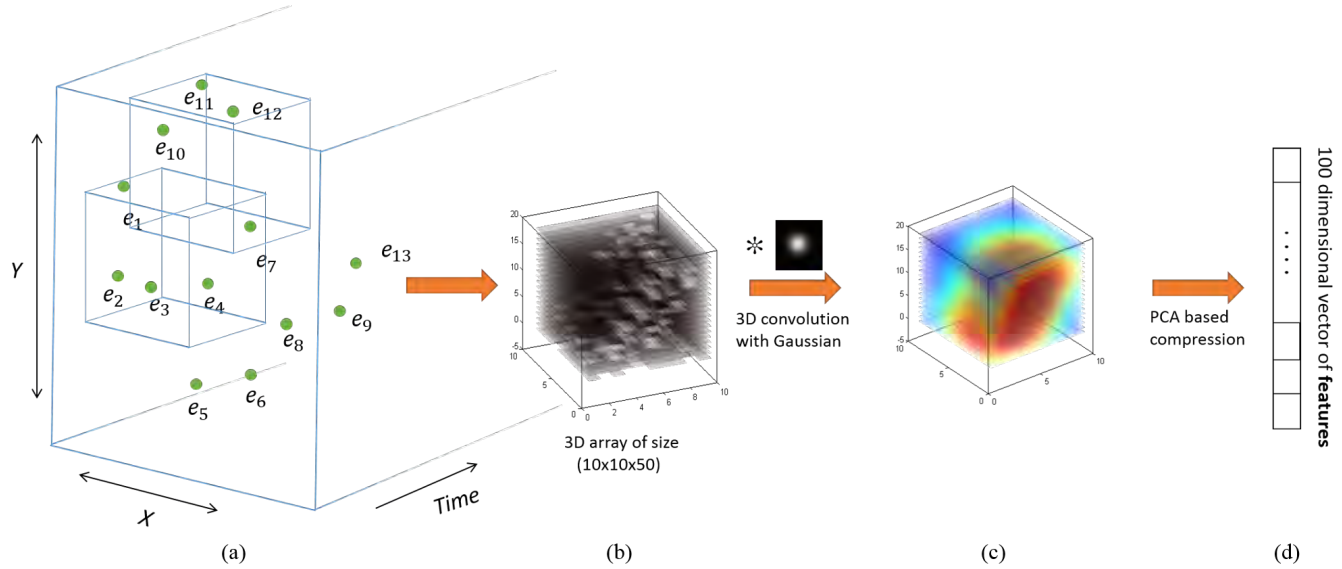


Fig. 3. An illustration of the steps involved in feature compression. (a) demonstrates the creation of box neighbourhoods with spike-event data; (b) example of a three dimensional spike count matrix for parameter values $\Delta T = 100$ ms, $M = 50$, and $a = 10$; (c) corresponding smoothed matrix with large smoothing parameters $\sigma_x = \sigma_y = \sigma_t = 10$ to emphasize the coarse distribution of events within the box neighbourhood; (d) schematic showing the pruning of the 5000 dimensional matrix to a much smaller, 100 dimensional feature vector using PCA.

To objectively determine how to reduce data dimensionality, principal components analysis (PCA) is performed on this set of data points. This yields d orthogonal linear projections (eigenvectors) represented by the matrix

$$\mathbf{W} = [\mathbf{w}_1\ \mathbf{w}_2\ ....\ \mathbf{w}_d] \quad l = 1, 2, 3 \ldots, d \tag{8}$$

The columns of the weight matrix $\mathbf{W}$ are ordered w.r.t. decreasing eigenvalues $(\lambda_1 > \lambda_2 > ... > \lambda_d)$.

To minimize dimensionality but retain maximum information content, we choose the first $n_{PCA} \ll d$ projections, which prunes the weight matrix to

$$\mathbf{W}_{PCA} \quad = \quad [\mathbf{w}_1\ \mathbf{w}_2\ ....\ \mathbf{w}_{n_{PCA}}]\,. \tag{9}$$

Hence, the feature vector $\mathbf{c}_i^{\boldsymbol{\sigma}} \in \mathbb{R}^d$ is mapped to $\mathbf{W}_{PCA}^T \mathbf{c}_i^{\boldsymbol{\sigma}} \in \mathbb{R}^{n_{PCA}}$. $\mathbf{W}_{PCA}^T$ is the matrix transpose of $\mathbf{W}_{PCA}$.

Let $\mathbb{W}_i$ be a 3D matricized[2] form of each weight vector $\mathbf{w}_i$. The feature vector $\mathbf{W}_{PCA}^T \mathbf{c}_i^{\boldsymbol{\sigma}}$ can be rewritten as

$$\begin{aligned} \mathbf{W}_{PCA}^T \mathbf{c}_i^{\boldsymbol{\sigma}} \quad &= [sum\,(\mathbb{W}_1 \circ (\mathbf{C}_i * \mathbf{G}_{\boldsymbol{\sigma}}))\ \ ... \\ &... \ sum\,(\mathbb{W}_{n_{PCA}} \circ (\mathbf{C}_i * \mathbf{G}_{\boldsymbol{\sigma}}))]. \end{aligned} \tag{10}$$

The symbol $\circ$ represents the hadamard product. Given a matrix $\mathbf{X}$, the function $sum(\mathbf{X})$ returns the sum of all the elements of $\mathbf{X}$.

Theorem 1. *Given 3D matrices* $\mathbf{A}$, $\mathbf{B}$ *and* $\mathbf{C}$ *where* $\mathbf{C}$ *is a 3D Gaussian kernel,* $sum\,(\mathbf{A} \circ (\mathbf{B} * \mathbf{C})) = sum\,((\mathbf{A} * \mathbf{C}) \circ \mathbf{B})$.

Proof. The given 3D matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are zero padded to make their size equal. Using the associative property of convolutions we can write,

$$\mathbf{A} * \mathbf{B} * \mathbf{C} = \mathbf{A} * \mathbf{C} * \mathbf{B}.$$

Let us consider 3D matrices $\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3$, each of size $(l_1 \times l_2 \times l_3)$, such that $\mathbf{Y}_1 = \mathbf{Y}_2 * \mathbf{Y}_3$. The central element of $\mathbf{Y}_1$ can be simply written as

$$\begin{aligned} \mathbf{Y}_1\left(\frac{l_1}{2}, \frac{l_2}{2}, \frac{l_3}{2}\right) &= \sum_{i,j,k}^{l_1,l_2,l_3} \mathbf{Y}_2\left(\frac{l_1}{2} - i, \frac{l_2}{2} - j, \frac{l_3}{2} - k\right) \mathbf{Y}_3\,(i,j,k) \\ &= sum\,((\sim \mathbf{Y}_1) \circ \mathbf{Y}_2) \end{aligned} \tag{11}$$

where $(\sim \mathbf{Y}_1)$ as the mirror image matrix of $\mathbf{Y}_1$, obeying $(\sim \mathbf{Y_1})\,(i,j,k) = \mathbf{Y}_1\left(\frac{s_1}{2} - i, \frac{s_2}{2} - j, \frac{s_3}{2} - k\right)$. Observe that $(\sim (\sim \mathbf{Y}_1)) = \mathbf{Y}_1$. Conversely, $sum\,(\mathbf{Y}_1 \circ \mathbf{Y}_2)$ is the central element of the convolution $(\sim \mathbf{Y}_1) * \mathbf{Y}_2$.

Therefore, $sum\,(\mathbf{A} \circ (\mathbf{B} * \mathbf{C}))$ is essentially the central element of the matrix $(\sim \mathbf{A}) * (\mathbf{B} * \mathbf{C})$. Using the associative property of convolutions, this matrix can be rewritten as $(\sim \mathbf{A} * \mathbf{C}) * \mathbf{B}$. Applying (11), its central element can simply be obtained as $sum\,((\sim (\sim \mathbf{A} * \mathbf{C})) \circ \mathbf{B})$.

It is trivial to show that $\sim (\mathbf{A} * \mathbf{B}) = (\sim \mathbf{A}) * (\sim \mathbf{B})$. Using this property we continue with the previous steps as follows

$$\begin{aligned} sum\,(\mathbf{A} \circ (\mathbf{B} * \mathbf{C})) &= sum\,((\sim (\sim \mathbf{A} * \mathbf{C})) \circ \mathbf{B}) \\ &= sum\,(((\sim\sim \mathbf{A}) * (\sim \mathbf{C})) \circ \mathbf{B}) \\ &= sum\,((\mathbf{A} * \mathbf{C}) \circ \mathbf{B})\,. \end{aligned}$$

The last step substitutes $\sim \mathbf{C} = \mathbf{C}$ because $\mathbf{C}$ is the 3D Gaussian distribution kernel matrix.

Note that this result holds for any $\mathbf{C}$ which is equal to its mirror matrix $\sim \mathbf{C}$. A more general result is

$$sum\,(\mathbf{A} \circ (\mathbf{B} * \mathbf{C})) = sum\,((\mathbf{A} * (\sim \mathbf{C})) \circ \mathbf{B})\,. \tag{12}$$

□

A direct application of Theorem 1 on (10) yields

$$\begin{aligned} \mathbf{W}_{PCA}^T \mathbf{c}_i^{\boldsymbol{\sigma}} &= [sum\,((\mathbb{W}_1 * \mathbf{G}_{\boldsymbol{\sigma}}) \circ \mathbf{C}_i)\,, ... \\ &..., sum\,((\mathbb{W}_{n_{PCA}} * \mathbf{G}_{\boldsymbol{\sigma}}) \circ \mathbf{C}_i)] \\ &= (\mathbf{W}_{PCA}^{\boldsymbol{\sigma}})^T \mathbf{c}_i. \end{aligned} \tag{13}$$

This result suggests that by reordering the gaus-

2. Matricization is referred to converting a $(1 \times n^2)$ vector into its original 2D matrix form. Here we extend the definition to 3D matrices.

Algorithm 1 Point match extraction (for the i^{th} eligible event)

Input: All eligible spike-events $E = \{e_1, e_2, ..., e_N\}$, Event to be matched $e_i = (x_i, y_i, t_i)$, $T_\delta = [\delta t_1, \delta t_2, .., \delta t_k]$
Overlap parameter r
Output: Event displacement parameters $(\delta x_i, \delta y_i, \delta t_i)$, Matched pair of spike-count vectors $(\mathbf{p}_i, \mathbf{q}_i)$
for $(\delta x, \delta y, \delta t) \in \{-1, 1\} \times \{-1, 1\} \times T_\delta$ **do**
$\mathbf{D}(\delta x, \delta y, \delta t) \leftarrow \left\{(\mathbf{W}^{\boldsymbol{\sigma}}_{PCA})^T F_E\left(x_i, y_i, t_i, \frac{\delta t}{1+r}\right) - (\mathbf{W}^{\boldsymbol{\sigma}}_{PCA})^T F_E\left(x_i + \delta x, y_i + \delta y, t_i + \delta t, \frac{\delta t}{1+r}\right)\right\}^2$
end for
$\delta x_i, \delta y_i, \delta t_i \leftarrow_{argmin\ \delta x, \delta y, \delta t} \mathbf{D}(\delta x, \delta y, \delta t)$; $\mathbf{p}_i \leftarrow F_E\left(x_i, y_i, t_i, \frac{\delta t_i}{1+r}\right)$; $\mathbf{q}_i \leftarrow F_E\left(x_i + \delta x_i, y_i + \delta y_i, t_i + \delta t_i, \frac{\delta t_i}{1+r}\right)$

sian smoothing, it is possible to associate $\mathbf{W}^T_{PCA}\mathbf{c}^{\boldsymbol{\sigma}}_i$ to $(W^{\boldsymbol{\sigma}}_{PCA})^T\mathbf{c}_i$. It then follows that the result

$$\mathbf{W}^T\mathbf{c}^{\boldsymbol{\sigma}}_i = (\mathbf{W}^{\boldsymbol{\sigma}})^T\mathbf{c}_i \tag{14}$$

holds for any weight matrix $\mathbf{W}$. $\mathbf{W}^{\boldsymbol{\sigma}}_{PCA}$ contains the modified weights obtained from $\mathbf{W}_{PCA}$ by convolving each matricized weight vector $\mathbb{W}_k$ with the 3D Gaussian kernel $\mathbf{G}_{\sigma_x,\sigma_y,\sigma_t}$. Therefore instead of repeating the Gaussian convolution operation on each spike count matrix C_i, only a one time convolution with the matricized weight vectors of the columns of $\mathbf{W}_{PCA}$ is required. This step just modifies the weight vectors to their smoothed versions. For ease of notation, we parametrize $\mathbf{c}_i$ as a function of the variables involved as

$$\mathbf{c}_i = F_E(x_i, y_i, t_i, a, \Delta T, M). \tag{15}$$

Hence, the final feature vector is denoted as

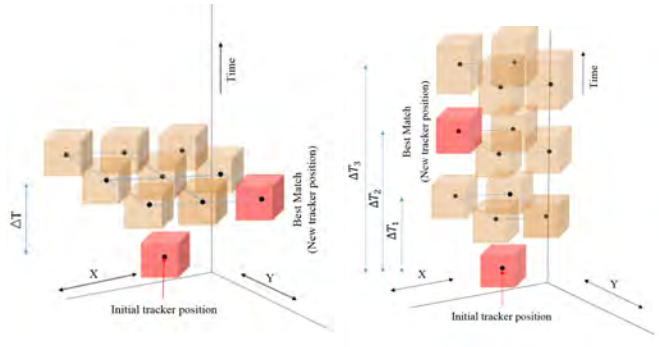
$$(\mathbf{W}^{\boldsymbol{\sigma}}_{PCA})^T F_E(x_i, y_i, t_i, \Delta T, M). \tag{16}$$

Finally, since we keep M unchanged throughout, the feature vector representation is further simplified as

$$(\mathbf{W}^{\boldsymbol{\sigma}}_{PCA})^T F_E(x_i, y_i, t_i, \Delta T), \tag{17}$$

removing M from the function arguments.

2.5 Extraction of feature vector matches



(a) Search space for a fixed temporal displacement, ΔT.

(b) Our proposed approach with variable temporal displacements.

Fig. 4. Contrast between two approaches to updating tracker position is shown. In (a), lower speeds of the feature point would lead to smaller spatial displacements and vice versa. Whereas in (b), a fixed spatial displacement implies that temporal displacement controls the speed of the tracker. Note the increase in temporal dimension of box neighbourhoods as ΔT increases, corresponding to time-scaling of the pattern in response to variable speeds. For simplicity overlapping box neighborhoods are not shown.

For each eligible event $e_i = (x_i, y_i, t_i)$, we seek its event displacement parameters $(\delta x_i, \delta y_i, \delta t_i)$. These provide the location of the event $(x_i + \delta x_i, y_i + \delta y_i)$, at a later time $t_i + \delta t_i$ $(\delta t_i > 0)$. To estimate them, a dissimilarity measure between a pair of feature vectors extracted at different (x, y, t) locations is needed. Given arbitrary displacement parameters $(\delta x, \delta y, \delta t)$, we choose the Euclidean distance

$$\left\| \begin{array}{c} (\mathbf{W}^{\boldsymbol{\sigma}}_{PCA})^T F_E(x_i, y_i, t_i, \Delta T) - \\ (\mathbf{W}^{\boldsymbol{\sigma}}_{PCA})^T F_E(x_i + \delta x, y_i + \delta y, t_i + \delta t, \Delta T) \end{array} \right\|^2, \tag{18}$$

as the measure of dissimilarity between feature vectors from locations (x_i, y_i, t_i) and $(x_i + \delta x, y_i + \delta y, t_i + \delta t)$.

The search space of probable $(\delta x, \delta y, \delta t)$ must permit variable speeds of feature points. One approach to finding these parameters involves doing a local grid search over possible values of $(\delta x, \delta y)$ as shown in Fig. 4 (a), with δt kept fixed. Instead, here we assign $|\delta x| = 1$ and $|\delta y| = 1$ and vary δt to account for different speeds. Let

$$T_\delta = \{\delta t_1, \delta t_2, .., \delta t_k\}$$

be the set of potential temporal displacements. The set of possible event displacements is represented by the cartesian product

$$\Delta_E = \{-1, 1\} \times \{-1, 1\} \times T_\delta. \tag{19}$$

The optimum displacement parameters $(\delta x_i, \delta y_i, \delta t_i)$ are the $(\delta x, \delta y, \delta t) \in \Delta_E$ for which the Euclidean distance

$$\left\| \begin{array}{c} (\mathbf{W}^{\boldsymbol{\sigma}}_{PCA})^T F_E\left(x_i, y_i, t_i, \frac{\delta t}{1+r}\right) - \\ (\mathbf{W}^{\boldsymbol{\sigma}}_{PCA})^T F_E\left(x_i + \delta x, y_i + \delta y, t_i + \delta t, \frac{\delta t}{1+r}\right) \end{array} \right\|^2, \tag{20}$$

is minimum. Note that the temporal dimension of the box neighborhood changes proportionally to the temporal displacement δt as $\Delta T = \frac{\delta t}{1+r}$. The parameter $r \in (0, 1)$ controls the overlap between the two box neighbourhoods.

The search space spans a region as shown in Fig. 4 (b). For the sake of simplicity non-overlapping box neighborhoods are shown. Formally the point matching algorithm is summarized in Algorithm 1. The algorithm outputs one matching pair of feature vectors. To extract all matching pairs across the training data, Algorithm 1 is repeated over all eligible events. Observe that the matches are solely decided by the choice of weights $\mathbf{W}_{PCA}$.

The matching pairs $\{\mathbf{p}_i, \mathbf{q}_i\}_{i=1}^K$ are used to optimize the weights to incorporate slowness in features across each match. This is described in the following section.

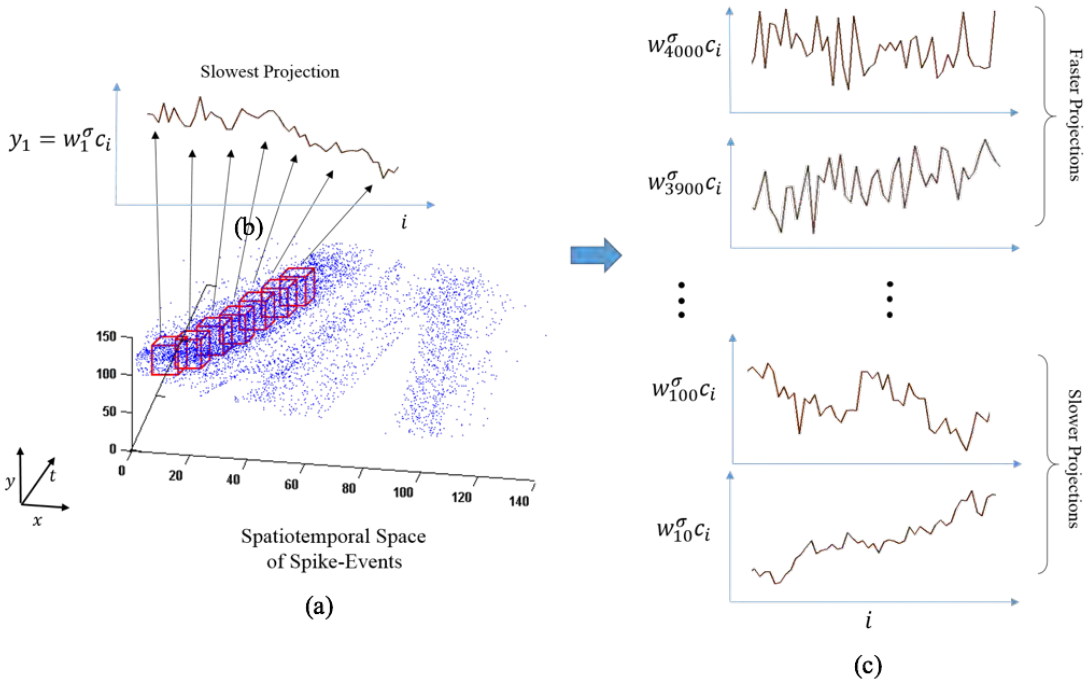


Fig. 5. An example of the slow feature extraction process on spike-event data (blue dots) obtained from the traffic dataset. (a) the boxes highlight the spatio-temporal path of the tracker, which in this case was made to follow a corner point. c_i represents the vectorized spike-count matrix obtained from the spatiotemporal bounding boxes (in red) of size $10 \times 10 \times 200$ms, (b) the sequence of values of the projection of w_1^σ is shown on the successive boxes formed by the tracker trajectory, numbered by the variable i, (c) shows contrast between sequences formed by weight vectors having different values of slowness. Note that the weight vectors w_k^σ, $k = 1$ to 4000, are ordered w.r.t decreasing values of the slowness parameter.

2.6 Learning slow features from matches

Once all the matched pairs of features $(\mathbf{p}_i, \mathbf{q}_i)_{i=1}^K$ have been extracted from the training data, we wish to estimate robust linear projections which exploit the correspondence information. Note that the matches here are the vectorized form of the unsmoothed 3D spike count matrix. First, the matricized form of these vectors are smoothed with the Gaussian kernel $\mathbf{G}_{\sigma_x,\sigma_y,\sigma_t}$, similar to Section 2.4. This leads to modified, smoothed feature point matches $(\mathbf{p}_i^{\boldsymbol{\sigma}}, \mathbf{q}_i^{\boldsymbol{\sigma}})_{i=1}^K$.

This section estimates linear projections represented by the columns of the matrix $\mathbf{W}_{SFA}$, which operate on the smoothed and vectorized spike-count matrices $\mathbf{c}_i^{\boldsymbol{\sigma}}$. To estimate the columns of $\mathbf{W}_{SFA}$, we apply the principles involved in slow feature analysis (SFA). First, we give a brief description of the steps in slow feature analysis. Given a temporal sequence of d dimensional vectors

$$(\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_i, .., \mathbf{X}_l)\,,$$

we define a *slowness* parameter

$$S(\mathbf{w}) = \frac{\mathbb{E}_i\left[\mathbf{w}^T\mathbf{X}_{i+1} - \mathbf{w}^T\mathbf{X}_i\right]^2}{\mathbb{E}_i\left[\mathbf{w}^T\mathbf{X}_i - \mathbb{E}\left[\mathbf{w}^T\mathbf{X}\right]\right]^2} \tag{21}$$

for a linear projection vector $\mathbf{w}$. Similar to PCA, the SFA method returns an orthogonal set of d linear projections $[\mathbf{w}_1\ \mathbf{w}_2\ ...\ \mathbf{w}_d]$ such that their slowness values $(S(\mathbf{w}_1), S(\mathbf{w}_2), .., S(\mathbf{w}_d))$ are in ascending order, with $\mathbf{w}_1$ having the smallest value among all possible projections. As mentioned in [21], the weights can be simply obtained by sphering the data followed by finding the eigenvectors ordered by the sorted d eigenvalues of the difference vectors $\mathbf{X}_{i+1} - \mathbf{X}_i$. Here, since we only have pairwise matches, the slowness parameter is modified as

$$S(\mathbf{w}) = \frac{\mathbb{E}_i\left[(\mathbf{w}^T\mathbf{q}_i^{\boldsymbol{\sigma}} - \mathbf{w}^T\mathbf{p}_i^{\boldsymbol{\sigma}})^2\right]}{\mathbb{E}_i\left[\mathbf{w}^T\mathbf{p}_i^{\boldsymbol{\sigma}} - \mathbb{E}_k\left[\mathbf{w}^T\mathbf{p}_k^{\boldsymbol{\sigma}}\right]\right]^2}. \tag{22}$$

Note that the numerator in the above equation is the difference in the feature value within a matching pair. The denominator is the variance of that particular value across all matches. Clearly, we want projections which have small intra-match differences and large inter-match differences. This would mean smaller $S(\mathbf{w})$ is preferred over larger. Thus we choose only $n_{SFA} << d$ projections having the lowest values of $S(\mathbf{w})$. The n_{SFA} linear projections are portrayed as the columns of the matrix

$$\mathbf{W}_{SFA} = [\mathbf{w}_1\ \mathbf{w}_2\ ...\ \mathbf{w}_{n_{SFA}}]\,,$$

where $\mathbf{W}_{SFA} \in \mathbb{R}^{(d \times n_{SFA})}$. Applying the result in (14), we generate the Gaussian smoothed equivalent of those weights as follows.

$$\mathbf{W}_{SFA}^T\mathbf{p}_i^{\boldsymbol{\sigma}} = (\mathbf{W}_{SFA}^{\boldsymbol{\sigma}})^T\mathbf{p}_i. \tag{23}$$

Therefore, the projections can be also obtained through the product of the smoothed weight matrix $\mathbf{W}_{SFA}^{\boldsymbol{\sigma}}$ with the vectorized spike count matrix $\mathbf{p}_i$. Like in Section 2.4, a one-time convolution operation on the 3D matrix representation of each weight vector column $\mathbf{w}_i$ generates the modified weights $\mathbf{W}_{SFA}^{\boldsymbol{\sigma}} = [\mathbf{w}_1^\sigma\ \mathbf{w}_2^\sigma\ ...\ \mathbf{w}_{n_{SFA}}^\sigma]$ which directly operate on the spike count matrices. Fig. 5 shows an example of the projections obtained from the trained weights changing with time. This step reduces the dimensionality of the feature representation by removing the noisy projection

Algorithm 2 Feature point tracking

Input: All Events $E = \{e_1, e_2, ..., e_N\}$
Initial tracker position: (x_0, y_0, t_0)
$T_\delta = \{\delta t_1, \delta t_2, .., \delta t_k\}$
Overlap parameter: r
Spatial dimension of neighborhood: a
Minimum event threshold for tracker displacement: N_0
Output: Tracker positions over k iterations $\{(x_0, y_0, t_0), (x_1, y_1, t_1), .., (x_{k-1}, y_{k-1}, t_{k-1})\}$
for $i \in \{0, 1, .., k-1\}$ **do**
for $(\delta x, \delta y, \delta t) \in \{-1, 1\} \times \{-1, 1\} \times T_\delta$ **do**
$\mathbf{D}(\delta x, \delta y, \delta t) \leftarrow \left\{(\mathbf{W}^{\boldsymbol{\sigma}}_{SFA})^T F_E\left(x_0, y_0, t_0, \frac{\delta t}{1+r}\right) - (\mathbf{W}^{\boldsymbol{\sigma}}_{SFA})^T F_E\left(x_i + \delta x, y_i + \delta y, t_i + \delta t, \frac{\delta t}{1+r}\right)\right\}^2$
$Count(\delta x, \delta y, \delta t) \leftarrow \left|\mathcal{BN}_E(x_i + \delta x, y_i + \delta y, t_i + \delta t, a, a, \frac{\delta t}{1+r})\right|$
end for
end for
$(\delta x_{min}, \delta y_{min}, \delta t_{min}) \leftarrow_{argmin\ \delta x, \delta y, \delta t} \mathbf{D}(\delta x, \delta y, \delta t)$
$minCount = \min_{\delta x, \delta y, \delta t}\{Count(\delta x, \delta y, \delta t)\}$
if $minCount > N_0$ **then**
$(x_{i+1}, y_{i+1}, t_{i+1}) \leftarrow (x_i + \delta x_{min}, y_i + \delta y_{min}, t_i + \delta t_{min})$
else
$(x_{i+1}, y_{i+1}, t_{i+1}) \leftarrow (x_i, y_i, t_i + \delta t_{min})$
end if

components which vary too much within a matching feature pair. Hence, the only purpose of $\mathbf{W}^{\boldsymbol{\sigma}}_{PCA}$ is to obtain matches across the event-data which are subsequently utilized to learn more robust projections in $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA}$. Note that the method of obtaining these weights is unsupervised.

2.7 Point tracking with slow features

Trackers are initialized at feature points throughout the event-data. Each tracker updates its position in time according to the motion of the feature point, and maintains a feature vector representation at all times. The objective is to accurately track feature points as they gradually undergo various transformations. The tracking method is elaborated in Algorithm 2. In a way, it can be visualized as the repeated application of the point matching algorithm described in Algorithm 1. The weight matrix used to extract features is the one learned from the matches, $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA}$. As shown in an example presented in Fig. 5 (a), the tracker simply updates its spatial and temporal position based on the best matches obtained at each iteration with a reference feature vector. The reference feature vector is extracted at the initial location of the tracker (x_0, y_0, t_0). Additionally, the tracker also maintains the count of the number of events inside each box neighbourhood.

It undergoes spatial displacement only when the event count in the box neighbourhood of the best match is greater than a threshold number. This we refer to as the *stopping criterion*, intended to keep the tracker static when the feature point stops all motion.

3 Experiments and Results

3.1 Pre-processing

A noise filtering routine [22] was applied to the obtained spike-events from each recording. The filtering process eliminates spike events which only have a single event in the box neighbourhood $\mathcal{BN}_E(e_i(x), e_i(y), e_i(t), 2, 2, 3 \times 10^4)$.

3.2 Tracker performance analysis

To test tracker performance, multiple ground truth trajectories of feature points were annotated across event-data. The feature points which were chosen for annotation were mainly corner-like points. Points on straight line edges were avoided as their motion cannot be estimated due to the aperture problem. We denote each i^{th} ground truth trajectory by the set $\left\{x^i_g(t), y^i_g(t), t \,|0 < t < T^i_{max}\right\}$, where t is the time since each tracker was initialized. Similarly, the i^{th} estimated trajectory from our method is represented by the set $\left\{x^i_e(t), y^i_e(t) \mid 0 < t < T^i_{max}\right\}$. The tracking performance is quantified using the following metric.

$$\tau(t) = \frac{\left\{\begin{array}{c} \#_i \sqrt{(x^i_g(t) - x^i_e(t))^2 + (y^i_g(t) - y^i_e(t))^2} \leq 7 \\ \mid T^i_{max} > t \end{array}\right\}}{\#_i \left\{T^i_{max} > t\right\}} \tag{24}$$

$\tau(t)$ represents the proportion of trackers which are within a distance of seven pixels from their corresponding ground truth locations at time t since their initialization. The plot of $\tau(t)$ over a fixed time interval of 0 (initialization) to 1.5 seconds is used as a quantifier of tracking performance for all our experiments. We refer to this curve as the *accuracy plot*. Most of the analyses presented in the following sections involves monitoring accuracy plots and their response to changes in algorithm parameters.

3.3 Traffic recording

The DVS was placed on the handrail of a pedestrian overpass facing the road below. The event data generated from vehicular motion in the direction towards and away from the DVS was recorded. The recording was done over a period of 15 minutes in broad daylight. The level of traffic ranged from one to five vehicles at any time instant, with a

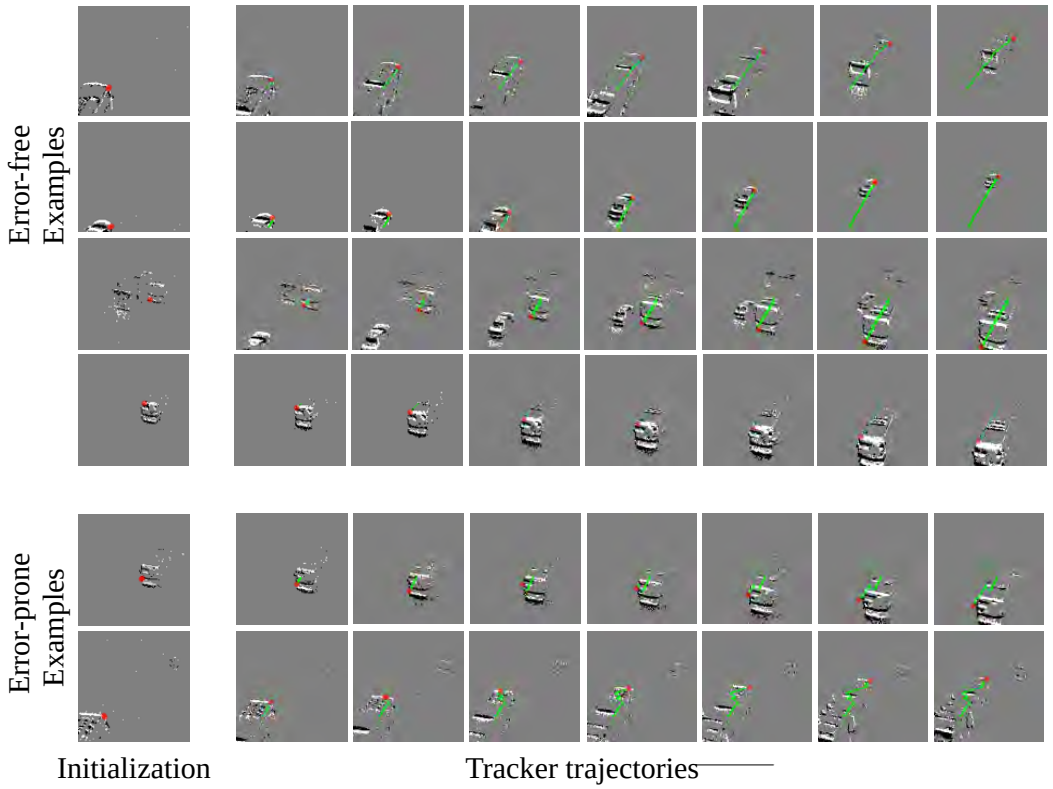


Fig. 6. Selected tracker trajectories computed by our algorithm ($M = 40$, $\Delta T = 200ms$, $a = 10$) on the testing data. On a given row, the left most frame shows the initial location of the tracker (in red). For the images to the right, the past trajectory of the tracker is overlayed onto each image in green, with the latest location of the tracker being shown by the red dot.

few instances showing occlusion. The curved road resulted in non-linear tracker trajectories. Fig. 1 shows example data from the recording. The initial five minutes of the data was used for training and the rest (10 minutes) for testing. The annotated data consists of 130 tracker trajectories evenly spaced throughout the data[3]. Fig. 6 displays selected tracker trajectories estimated by our algorithm across the testing data, separated by error-free and error-prone performance computed w.r.t the respective ground truth trajectories.

3.3.1 Varying temporal dimension

The temporal dimension of the box neighbourhoods, ΔT, used for obtaining the initial set of feature vectors in training was varied. To maintain the dimensionality of the feature vector, the temporal dimension corresponding to the temporal displacements was also scaled accordingly. As Fig. 7(b) shows, the performance is usually worse for lower values of ΔT. For instance when $\Delta T = 10\ ms$ not enough events are captured to obtain any meaningful information, which leads to a significant drop in performance compared to other values. Increasing ΔT indefinitely does not affect performance, as the weights adapt and recover relevant information. In fact, $\Delta T = 200\ ms$ returns the highest proportion of correct trackers $\tau(t)$.

3. The data along with the annotations are available online at *https://files.fm/u/933uurrr?k=90a4c919*.

3.3.2 Varying temporal partitions

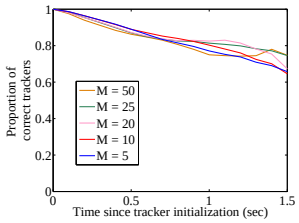
The number of temporal partitions M was varied. The temporal dimension of the box neighbourhoods used to obtain $\mathbf{W}_{PCA}$ was fixed at $\Delta T = 100\ ms$. The accuracy plots were studied for different values of M, and are shown in Fig. 7(a). The lowest number of partitions ($M = 5$, equivalent to 50 FPS) shows a steady decline of performance with time. In contrast, higher values such as $M = 25$ (equivalent to 250 FPS) demonstrate stability with time. A very high number of temporal partitions ($M = 100$, equivalent to 1000 FPS) makes performance suffer noticeably.

3.3.3 Varying the number of PCA projections

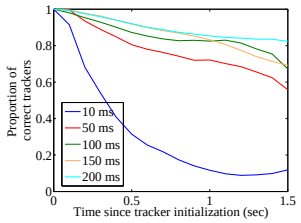
The number of projections n_{PCA} of $\mathbf{W}_{PCA}$ used to extract matches in Algorithm 1 was varied, and the subsequent accuracy plots obtained with the learned SFA weights were compared. For this analysis, parameter values chosen were $\Delta T = 100\ ms$ and $M = 25$. Fig. 7(c) contains the relevant accuracy plots. Overall, the effect of changing n_{PCA} is not very noticeable. However, a very high value, which preserves most of the information in the original spike-count vector, makes performance suffer. Using very few number of PCA components ($n_{PCA} = 2$) is however not ideal. For a slightly higher value $n_{PCA} = 10$, the most stable performance of the tracker in time is observed.

3.3.4 Varying the number of SFA projections

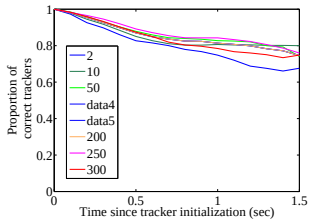
Another free parameter in our algorithm is the choice of n_{SFA}, the number of slow projections used in Algorithm 2.



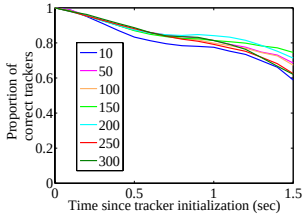
(a) Accuracy plots for different values of temporal partitions M.



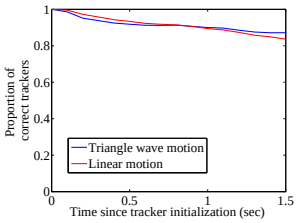
(b) Accuracy plots for different values of temporal dimension ΔT.



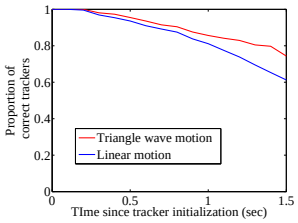
(c) Accuracy plots for different values of the number of principal components n_{PCA}.



(d) Accuracy plots for different values of the number of slow projections n_{SFA}.



(e) Accuracy plots tested on the MNIST linear motion, with weights from triangle wave motion



(f) Accuracy plots tested on the MNIST triangle wave motion, with weights from linear motion

Fig. 7. Shown are some of the accuracy plot analyses from our experiments. Figures (a)-(e) shows all accuracy plots containing the analyses of tracker performance in the traffic dataset. In short, (a) indicates that preserving more time information of each event leads to better performance; (b) highlights that acquiring more information about the event pattern around a feature point, by increasing the ROI, leads to better performance; (c) indicates that using fewer principal components for the initial tracker matches is more beneficial; (d) shows about 150 slowest projections can be considered as the optimum number for best performance in the traffic dataset; Figures (e) and (f) are the accuracy plots for the MNIST digits experiment. They show how the features adapt to the spatiotemporal patterns present in data.

We compare the accuracy plots with different values of n_{SFA} in Fig. 7(d). First note that a small value of $n_{SFA} = 10$ expectedly yields low accuracy (60% at $t = 1.5s$). Using only 10 slow features therefore does not provide enough information for the tracker to accurately identify the event patterns across time. A relatively larger value of $n_{SFA} = 150$ yields stable performance. Still larger values of n_{SFA} leads to a decrease in accuracy to almost as low as that obtained for $n_{SFA} = 10$. As the projections are arranged in ascending order of their slowness, the subsequent projections are noisier and therefore can worsen tracking accuracies.

3.3.5 Comparisons with other methods

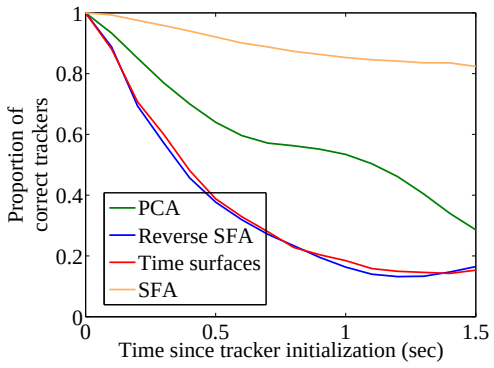


Fig. 8. Accuracy plots comparing point tracking performance of our method with other feature estimation strategies.

We compare other methods with SFA weights $\mathbf{W}_{SFA}$ obtained with $M = 20$ and $\Delta T = 200ms$. Three other ways of generating feature vectors were used, which are detailed below. The accuracy plots obtained from each method is shown in Fig. 8.

- PCA: Instead of $\mathbf{W}_{SFA}$, $\mathbf{W}_{PCA}$ was used as the weight matrix in Algorithm 2. The number of projections n_{PCA} was fixed at 150. Projections from our method achieve much higher accuracies than the principal component based projections $\mathbf{W}_{PCA}$. This is because PCA does not consider temporal contiguity in obtaining linear projections.
- Reverse SFA: Instead of using the $n_{SFA} = 150$ slowest projections to construct $\mathbf{W}_{SFA}$, we take the 150 projections having the highest value of slowness parameter $S(w)$. The accuracy plot for these weights demonstrate its poor performance. This result reinforces the contrast in information encoded between the slower and faster weights. Clearly, the projections having least temporal contiguity also show very poor performance in point tracking applications.
- Time surfaces: Time surfaces represented as $T_s(x, y, t)$ were used as the feature vectors. All the parameters and steps involved in Algorithm 2 were unchanged, except $\mathbf{D}(\delta x, \delta y, \delta t)$ which was changed to $\left(e^{-(t_0 - T_s(x_0,y_0,t_0))/\tau} - e^{-(t_i - T_s(x_i+\delta x, y_i+\delta y, t_i+\delta t))/\tau}\right)^2$. The parameter τ was set at $50ms$. Note that this approach yields an unbiased tracker which is independent of the training data, as it does not have any element of learning. Not surprisingly, the accuracy plots obtained from this method perform poorly, very similar to reverse SFA projections.

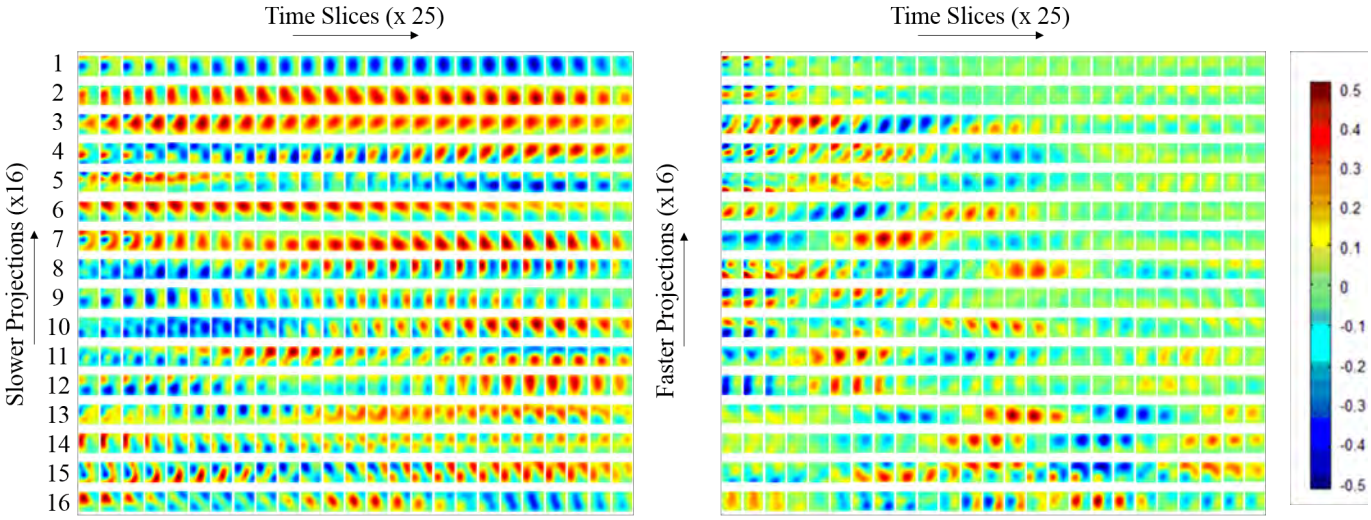


Fig. 9. (a) shows 16 spatiotemporal weight vectors which have the most slowly changing projections, from the weight matrix $W^{\boldsymbol{\sigma}}_{SFA}$. (b) shows 16 weight vectors with most rapidly changing projections. On a given row, each image is obtained by keeping the temporal index fixed of the 3D weight matrices. Each image represents a time slice of the 3d weight matrix of size 10×10. A total of 25 time slices are shown for each weight matrix. The color coding of the pixel values is shown to the right. The images are ordered from left to right w.r.t increasing temporal index.

3.3.6 Visualization of the SFA weight matrix

The smoothed weights $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA}$ obtained from the traffic data with parameter values $M = 25$ and $\Delta T = 100ms$ is shown in Fig. 9. Only the fifteen projections with smallest values of slowness parameter $S(\mathbf{w})$ are shown in Fig. 9 (a). Similarly, the fifteen projections with the largest values of their slowness parameters are shown in Fig. 9 (b). Each 3D matricized weight vector from the columns of $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA}$ is of size $(10 \times 10 \times 25)$, shown in the form of twenty five (10×10) images. The pixels are color coded with $blue$ being the most negative weights and red the most positive. The observations are summarized below.

- The weight vectors with rapidly changing projections exhibit high degree of sparseness compared to the slowly changing projections. As such, the slowly changing and rapidly changing projection weights can be clearly differentiated.
- The slowest changing projection weights are shaped as big 3D blobs which count the number of events inside them. Some of them have patterns which translate with time, indicating responsiveness to translation.
- In Fig. 9 (a), most weights show smoothly shifting patterns across time. Some of them have corner-like shapes (4,7 and 13). A few weights respond to event rate changes (10 and 12), indicating that they encode whether the feature point is moving towards or away from the camera (corresponding to vehicles moving in either direction). There are also weights which respond to specifically oriented edges moving in a certain direction (6 and 8). These properties indicate similarities of these weights to visual cortical receptive fields of different kinds (simple, complex and hypercomplex [23]).

3.4 MNIST digits motion

Images from the MNIST digits dataset were taken, with two examples selected for each digit. Two sets of training and testing data were recorded with the DVS, each consisting of the digits moving in a different motion profile as shown in Fig. 10 (a) & (b). The two motion profiles used are described below.

- Linear motion: The digits were simply moved horizontally in a straight line.
- Triangle wave motion: The digits were moved horizontally along a triangular wave trajectory.

From the training data for each motion profile, two weight matrices $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA,L}$ and $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA,T}$ were obtained. The following sections summarize the testing procedures. The parameters used in this experiment were $n_{PCA} = 10$, $n_{SFA} = 150$, $M = 25$ and $\Delta T = 100ms$.

3.4.1 Linear motion testing

Both $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA,L}$ and $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA,T}$ were used to obtain accuracy plots on testing data with linear motion. $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA,L}$ is expected to perform better on this data, as it was trained with similar patterns. The results are shown in Fig. 7(e).

3.4.2 Triangle wave motion testing

Similar to the previous experiment, both sets of weights were used to obtain accuracy plots on testing data with triangle wave motion. The results are shown in Fig. 7(f).

From the plots, we notice that the cross-dataset tracking performance only suffers a little, despite two different motion profiles. It is because the spatial patterns remain consistent across the two datasets. Unsurprisingly, we notice better tracking accuracy when the trained weights are applied to their own datasets. However, the cross-dataset performance of the triangle wave motion trained weights eventually

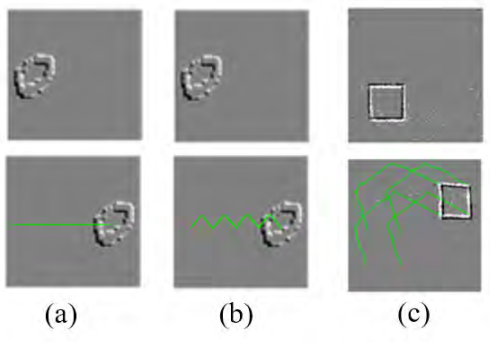


Fig. 10. Example motion profiles for the MNIST digits and the square corner tracking experiments. (a) Linear motion profile and (b) triangle-wave motion profile of an example digit from the MNIST dataset is shown. The top and bottom images constitute the initial and final locations of each digit respectively, with its trajectory overlaid in green. Similarly shown in (c) are the initial and final locations of the square along with the superimposed corner trajectories.

matches the linear motion weights (as shown in Fig. 7(e)) on the linear motion data itself. This can be explained as follows. Recall that the DVS only responds to moving edges, with greater spike-events generated when the edge moves along its normal direction. The triangular wave motion data on which the corresponding $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA,T}$ is trained, clearly has a wider range of motion direction than the linear motion data. Therefore, $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA,T}$ is subject to input where more edge information is revealed than its counterpart $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA,L}$. Naturally $\mathbf{W}^{\boldsymbol{\sigma}}_{SFA,T}$ learns to better encode the spike-event patterns than its counterpart.

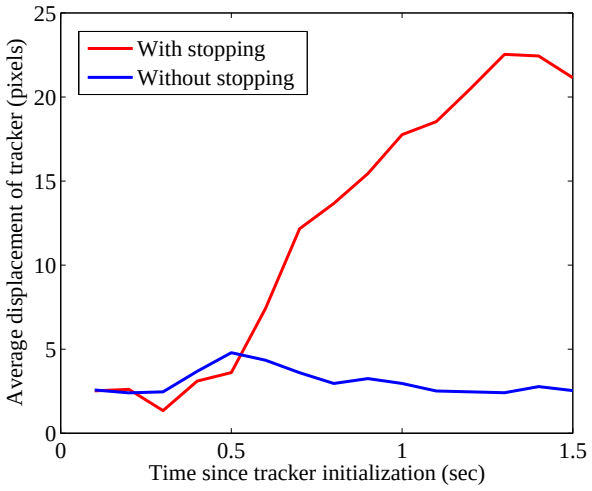


Fig. 11. Accuracy plots comparing Algorithm 2 performance with the stopping condition and without.

3.5 Intermittent motion

In this experiment a square shaped stimulus was translated in a piecewise linear curve. As the square moved, it was paused for 0.1 seconds at $t = 0.5s$ and $t = 1s$ before changing motion direction. The stopping criterion used in Algorithm 2 is pivotal to ensure that the tracker stays put as all motion ceases. Fig. 10 (c) shows the starting position of the square and the point trajectories annotated. To compare, we quantify tracking performance w.r.t the average tracker displacement which is defined as follows.

$$dist(t) = \frac{\sum_{i,T^i_{max}>t} \sqrt{(x^i_g(t) - x^i_e(t))^2 + (y^i_g(t) - y^i_e(t))^2}}{\#_i \{T^i_{max} > t\}} \tag{25}$$

Note that tracking performance does not suffer for the algorithm without the stopping criterion until $t = 0.5s$. For $t > 0.5$, the average displacement of the trackers without the stopping criterion increases rapidly, eventually reaching 20 pixels. The tracker with the stopping criterion however maintains performance as desired.

4 Discussion

There can be a multitude of ways to learn and extract spatiotemporal features, as is evident from the plethora of work done in frame-based vision (discussed in [24]). However, a commonality in all approaches is compression of the information present in a local spatiotemporal region, which is often high-dimensional. This is also observable in event-based feature extraction literature. Time-surfaces used in [15] essentially compress the three dimensional spike-event cloud into a two dimensional image over a local area. In [14], the spike-event pattern is also compressed into a discrete dictionary involving one of nine classes. Our approach essentially describes an intelligent compression algorithm, which works with the information present only within a local cloud of spike-events. However, here we go beyond the main objective of compression, which is to preserve as much input information with as few output components as possible. In addition we minimize the variation of those components in response to spatio-temporal perturbation, citing transformation invariance as the main motivation behind this step. Robustness to transformations is not achievable through simple data compression methods (for e.g. PCA). When we do use PCA based representations, tracking performance considerably suffers as observed in Fig. 7(c).

The feature extraction and feature point tracking methods proposed here have a few aspects that require further discussion and explanation. They are listed as follows.

1) **3-D spike count matrix creation:** Each box neighborhood is split into multiple cuboidal voxels. The temporal dimension $\left(\frac{\Delta T}{M}\right)$ of each voxel determines the recoverable spike timing information. A small temporal dimension preserves spike timing details but also increases the feature vector dimension. In this case, the computed feature vectors are very sparse since most voxels are zero in value (unoccupied). Large values would lead to greater loss of time information present in each spike-event. Choosing just the right value of M leads to spike-count matrices which are robust to slight perturbation of spike times, while simultaneously preserving collective information present in the group of spikes within.
2) **3-D Gaussian Smoothing of spike-count matrices:** Without Gaussian smoothing, the Euclidean distance between two spike-count vectors containing two spikes

in different locations would be the same regardless the spatiotemporal distance between the spikes. Gaussian smoothing ensures that two spike-count vectors containing spikes in neighboring locations are closer to each other in the p^2M dimensional space, as compared to the case where the spikes are not neighbors. Hence, it preserves some of the topological properties of the spatiotemporal (X, Y, T) space in the initial feature vector representation.

3) **Time dimension scaling of box neighborhoods while tracking:** An important aspect of our tracking and matching algorithms is how the temporal dimension ΔT of the box-neighborhoods adjusts to its temporal displacement. We choose smaller temporal dimensions for short temporal displacements (faster speed) and vice-versa. This can be intuitively reasoned as follows. Let us consider patterns of events in the spatiotemporal domain generated from a feature point at different speeds v_1 and v_2 where $v_2 = 2v_1$. The time taken for the pattern to surface for v_2 is exactly half compared to the former. Therefore, the temporal dimension must adapt to the speed at which the pattern is translating across the spatial domain. Let the temporal dimension ΔT relate to the temporal displacement t_δ as $\Delta T = \alpha t_\delta$. Since the speed is $v = \frac{1}{t_\delta}$ the displacement of a feature point within the box neighbourhood is $v\Delta T = \alpha$, which is a constant. Therefore, α (5 in our experiments) can be tuned to control the extent of pixel displacement of the feature point within a box neighbourhood.

The results highlight the ability of the proposed approach to find meaningful trends in the patterns of spike-events in a particular visual setting. For example, few of the SFA weight matrices learnt from traffic data (number 10 and 12 in fig. 9) simply compute the differential of rate of occurrence of spike-events. This differential essentially informs us of the motion direction (towards or away from camera) of the corresponding vehicle that the point is a part of. Note that this property of a feature point will remain unchanged throughout the duration of its existence, thereby proving important for its continual detection and tracking. The linearity of the feature computation process reduces the number of parameters that need to be learnt. This in turn allows the algorithm to generalize well over unseen data, with relatively few minutes of training samples. This is highlighted in the traffic dataset experiments, where only acquiring five minutes of data gives desirable point tracking performance on the next ten minutes of unseen data.

5 Conclusion

An unsupervised feature learning method for obtaining spatiotemporally smooth and informative features is proposed. The feature learning criteria enforced are (i) smooth responses to individual spike-event perturbations, and (ii) slowly changing responses to change of spatio-temporal location attributing to re-appearance of feature points. We evaluate the obtained feature representations by the trackability of feature points across multiple event-data recordings. The method outperforms other ways of obtaining spatiotemporal features including time surface based representations. The features are seen to specialise towards the spatial and temporal motion patterns dominant in the training data, while simultaneously showing good generalisation across datasets containing different motion patterns.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *CoRR*, 2015.

[3] W. Hoff and N. Ahuja, "Surfaces from stereo: integrating feature matching, disparity estimation, and contour detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 2, pp. 121–136, Feb 1989.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[5] E. Elhamifar and R. Vidal, "Sparse subspace clustering." in *CVPR*. IEEE, 2009, pp. 2790–2797.

[6] D. N. Ta, W. C. Chen, N. Gelfand, and K. Pulli, "Surftrac: Efficient tracking and continuous object recognition using local feature descriptors," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 2937–2944.

[7] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition," in *Proc. BMVC*, 2009, pp. 124.1–124.11.

[8] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 times; 128 120 db 15 956;s latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb 2008.

[9] C. Posch, D. Matolin, and R. Wohlgenannt, "An asynchronous time-based image sensor," in *2008 IEEE International Symposium on Circuits and Systems*, May 2008, pp. 2130–2133.

[10] C. Brandli, R. Berner, M. Yang, S. C. Liu, and T. Delbruck, "A 240 x00d7; 180 130 db 3 x00b5;s latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct 2014.

[11] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Found. Trends. Comput. Graph. Vis.*, vol. 3, no. 3, pp. 177–280, Jul. 2008.

[12] H. Akolkar, C. Meyer, Z. Clady, O. Marre, C. Bartolozzi, S. Panzeri, and R. Benosman, "What can neuromorphic event-driven precise timing add to spike-based pattern recognition?" *Neural Computation*, vol. 27, no. 3, pp. 561–593, March 2015.

[13] R. Benosman, C. Clercq, X. Lagorce, S. H. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 407–417, Feb 2014.

[14] X. Lagorce, S.-H. Ieng, X. Clady, M. Pfeiffer, and R. B. Benosman, "Spatiotemporal features for asynchronous event-based data," *Frontiers in Neuroscience*, vol. 9, p. 46, 2015.

[15] X. Lagorce, G. Orchard, F. Gallupi, B. E. Shi, and R. Benosman, "Hots: A hierarchy of event-based time-surfaces for pattern recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2016.

[16] O. Bichler, D. Querlioz, S. J. Thorpe, J.-P. Bourgoin, and C. Gamrat, "Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity," *Neural Networks*, vol. 32, pp. 339 – 348, 2012.

[17] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 2794–2802.

[18] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 539–546 vol. 1.

[19] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, July 2012.

[20] G. Nebehay and R. Pflugfelder, "Tlm: Tracking-learning-matching of keypoints," in *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*, Oct 2013, pp. 1–6.

[21] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural Comput.*, vol. 14, no. 4, pp. 715–770, Apr. 2002.

[22] D. Czech and G. Orchard, "Evaluating noise filtering for event-based asynchronous change detection image sensors," in *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, June 2016, pp. 19–24.
[23] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurons in the cat's striate cortex," *Journal of Physiology*, vol. 148, pp. 574–591, 1959.
[24] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Proceedings of the 14th International Conference on Computer Communications and Networks*, ser. ICCCN '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 65–72. [Online]. Available: http://dl.acm.org/citation.cfm?id=1259587.1259830

Rohan Ghosh received the B.Tech. and M.Tech. degrees in Electronics and Electrical Communication Engineering from Indian Institute of Technology, Kharagpur, India in 2013 and 2014 respectively. He is currently working towards the PhD degree in Neuromorphic Vision in the Department of Electrical and Computer Engineering at National University of Singapore, Singapore.

Anupam K. Gupta received the B.Tech. degree in Mechanical Engineering from Institute of Engineering and Technology, Lucknow, India, in 2008, M.Sc. in Mechanical Engineering from Masachussetts Institute of Technology, Cambridge, USA, and Nanyang Technological University, Singapore in 2010 and PhD degree in Mechanical Engineering from Virginia Tech, Blacksburg, USA in 2016. He is currently a postdoctoral research fellow in the Singapore Institute for Neurotechnology (SINAPSE), National University of Singapore where his research focuses on bio-inspired sensing and algorithms for Vision and Tactile modalities. He has previously worked in IBM Research, Yorktown, N.Y., USA on bio-inspired speech recognition.

Siyi Tang received the B.Eng. degree in Electrical Engineering from National University of Singapore in 2016. She was a recipient of National University of Singapore Science and Technology Undergraduate Scholarship.

Nitish V. Thakor is the Director of the Singapore Institute for Neurotechnology (SINAPSE) at the National University of Singapore, as well as the Professor of Biomedical Engineering at Johns Hopkins University in the USA. Dr. Thakors technical expertise is in the field of Neuroengineering, where he has pioneered many technologies for brain monitoring to prosthetic arms and neuroprosthesis. He is the author of 340 refereed journal publications, 20 patents, and co-founder of 3 companies. He is currently the Editor in Chief of Medical and Biological Engineering and Computing, and was the Editor in Chief of IEEE TNSRE from 2005-2011. Dr. Thakor is a recipient of a Research Career Development Award from the National Institutes of Health and a Presidential Young Investigator Award from the National Science Foundation, and is a Fellow of the American Institute of Medical and Biological Engineering, IEEE, Founding Fellow of the Biomedical Engineering Society, and Fellow of International Federation of Medical and Biological Engineering. He is a recipient of the award of Technical Excellence in Neuroengineering from IEEE Engineering in Medicine and Biology Society, Distinguished Alumnus Award from Indian Institute of Technology, Bombay, India, and a Centennial Medal from the University of Wisconsin School of Engineering.