Learning Visual Motion Segmentation using Event Surfaces

Anton Mitrokhin*, Zhiyuan Hua*, Cornelia Fermüller, Yiannis Aloimonos
University of Maryland, College Park
College Park, Maryland
amitrokh@umd.edu, howardh@terpmail.umd.edu, fermulcm@umd.edu, jyaloimo@umd.edu

Abstract

Event-based cameras have been designed for scene motion perception - their high temporal resolution and spatial data sparsity converts the scene into a volume of boundary trajectories and allows to track and analyze the evolution of the scene in time. Analyzing this data is computationally expensive, and there is substantial lack of theory on dense-in-time object motion to guide the development of new algorithms; hence, many works resort to a simple solution of discretizing the event stream and converting it to classical pixel maps, which allows for application of conventional image processing methods.

In this work we present a Graph Convolutional neural network for the task of scene motion segmentation by a moving camera. We convert the event stream into a 3D graph in (x, y, t) space and keep per-event temporal information. The difficulty of the task stems from the fact that unlike in metric space, the shape of an object in (x, y, t) space depends on its motion and is not the same across the dataset. We discuss properties of of the event data with respect to this 3D recognition problem, and show that our Graph Convolutional architecture is superior to PointNet++. We evaluate our method on the state of the art event-based motion segmentation dataset - EV-IMO and perform comparisons to a frame-based method proposed by its authors. Our ablation studies show that increasing the event slice width improves the accuracy, and how subsampling and edge configurations affect the network performance.

1. Introduction

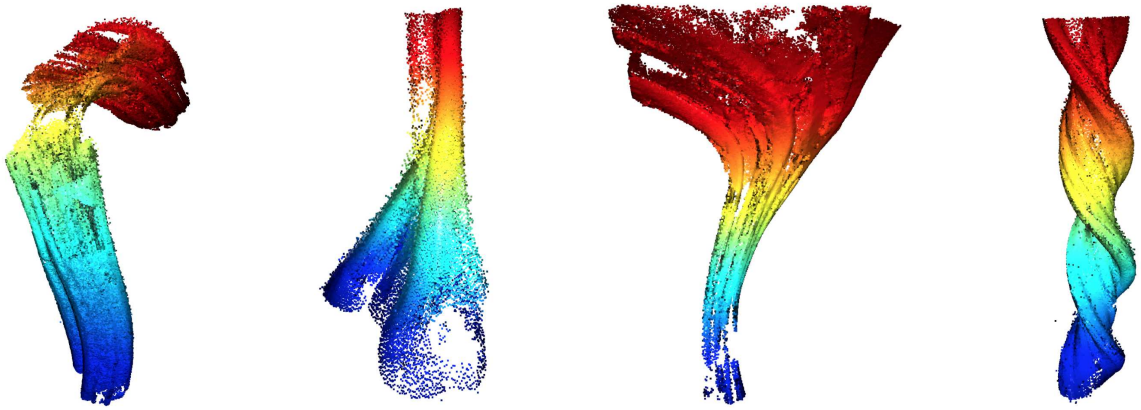
Scene motion analysis been studied for many years [16, 21, 1]. Lately, there has been an increased interest in these problems due to the applications in autonomous navigation [13]. The basic image representation for motion analysis is optical flow, representing pixelwise motion between two moments in time. Optical flow is subject to many ambiguities [33, 11]. On the other hand, feature point tracking allows for long-term estimation of pixel motion trajectories. That way, by analyzing pixel matches over large time intervals the ambiguity in motion can be resolved. Scene motion *exists in time*. Changes in the scene over short time intervals provide some information, but tasks, such as occlusion detection, segmentation of multiple moving objects, and detecting objects with a motion similar to the camera, require capturing the changes over longer time intervals to resolve ambiguities. Classic frame-based vision, however, is not naturally designed to provide temporal information.

With the growing enthusiasm for technologies such as VR and gesture recognition, many companies have started to invest in the development of event-based cameras [19, 28, 7, 32]. These sensors provide dense temporal information about changes in the scene; with every pixel acting as an independent electrical circuit such sensors are not driven by a common clock - every pixel reacts to motion independently, allowing for a more efficient, generic and at the same time accurate perception of the dynamic aspect of the scene. An additional benefit of these sensors is better tolerance to varying light conditions and sparse data encoding which makes event-based cameras useful for mobile devices.

For an event camera, every visible moving edge in the scene produces a trail of events, - an *event cloud*, which lies on a surface (called *event-surface* or *time-surface* [18]) in (x, y, t) space. The time-surface contains all the information about structure and motion. Unlike classical 3D processing (with RGB-D or Lidar sensors), the shape of the event clouds is constrained by the laws of physics and epipolar geometry. A certain shape of a cloud, even locally, might signify that two objects occlude each other, collide with each other or move closer to the camera.

Examples are shown in Fig. 1. The event clouds for four object motions are illustrated (the color of points corresponds to the event timestamp, in the range of 0 to 1 sec.): (a) translation parallel to the camera plane does not change the shape of the cloud across time, only its spatial coordinates;(b) roll, or rotation around the axis parallel to the camera plane reveals previously occluded parts of the ob-

*The authors contribute equally to this work.



(a) (b) (c) (d)

Figure 1. Illustration of event cloud shape properties depending on camera or object motion. Each point is an event, the timestamp is shown with color: blue - 0 sec, red - 1 sec. (a) - translation parallel to camera plane (along y axis), (b) - roll (rotation around y axis), (c) - translation along z axis (object moves closer to the camera), (d) - yaw (rotation around z axis)

ject, and the shape of the cloud cross-section changes; (c) with translation along the camera's optical axis, as the object approaches the camera its contour becomes larger; (d) rotation around the optical axis produces a discernible twist pattern. From these clouds, we also can see how natural the object tracking problem becomes, given the high temporal density of events. We discuss 3D motion pattern features in Sec. 3.

The time-surfaces have their origin in the Epipolar plane image analysis introduced in the late 80's [6]. As the camera moves along a linear path, images were taken in such rapid succession that they formed a solid block of data. The technique utilized knowledge of the camera motion to form and analyze slices of this solid. These slices directly encode not only the three-dimensional positions of objects, but also spatiotemporal events such as the occlusion of one object by another. For straight-line camera motions, these slices have a simple linear structure that makes them easy to analyze.

Generalizing this concept, we work with time-surfaces in (x, y, t) produced by an event camera to perform the task of foreground-background segmentation of moving objects. The complexity of this task comes from the enormous amount of asynchronous data which needs to be processed, high levels of noise produced by the event camera and, most importantly, the fact that *the variability of object and camera motion changes the shape of the event cloud*, making it more difficult to learn local 3D features. In summary, the contributions of this paper are:

- The first learning approach on 3D event clouds over large time intervals.
- We show theoretically and with experiments that larger temporal slices yield better performance.
- We perform comparisons to current state of the art, using *PointNet++* [30] and *EV-IMO* [25] as baselines, and show that our method is faster and yields better results.

2. Related Work

While most works process event data, by collapsing information into 2D image maps, a few approaches have adopted concepts from 3D processing. The best known flow techniques on event cameras compute normals to local time-surfaces [5, 10, 26, 27] to estimate normal flow. A block matching algorithm by Liu [20] uses similar ideas but can produce full flow for corner regions. The approach in [2] tracks events caused by contrast edges over multiple pixels to estimate normal flow, and [8, 3] use local frequency measurements defined on the event count maps. Recent motion compensation approaches use the temporal information as third dimension within a slice of events, to derive flow, and local or global motion models [24, 12].

Event-based feature detection research pursues similar ideas, but more global in temporal domain. Early works used the continuity of the event stream to bridge the gap between classical camera frames [34]. Later, Zhu *et al*. introduced a probabilistic, event-only approach to corner extraction [39]. Manderscheid *et al*. have significantly improved on existing methods by using deep learning [22]. Lagorce *et al*. [17, 18] addressed event cloud analysis in terms of time-surfaces by designing temporal features and demonstrated them in recognition tasks, and Chandrapla *et al*. [9] learned motion invariant space-time features. These works laid the foundations of the spatio-temporal feature analysis on event clouds and are precursors to the global event-cloud analysis.

Learning approaches on event data are quite numerous, with many authors emphasising the importance of encoding temporal information as input features for neural networks.

To name a few, [38] learns optical flow by constructing a discretized map with event timestamps, similar to the average time image used in [37]. An improved work by Zhu [40] uses multiple slices as input, retaining 3D structure of the cloud better. Barranco *et al*. [4] used different local spatio-temporal features to learn borders of objects. One of the most recent works - *EV-IMO* [25] presents a motion segmentation pipeline and a dataset which we use in this paper.

The closest to our approach is *EventNet* [31] – inspired by *Pointnet* [29], the first learning approach on 3D point clouds. This work analyzes events as points in 3D space, but only on slices up to $32ms$ wide, making it on par with image-based approaches in terms of data representation. Since the work uses a Multi Layer Perceptron, the spatial structure of events is not explicitly represented. Furthermore, EventNet has simple experimental results; it performs segmentation on planar shapes only, and shows marginal improvements of 0.1% in the mIoU (mean Intersection-Over Union) metric compared to the PointNet baseline. We build our approach around the *Graph Convolutional Network* [15, 36] structure, and we demonstrate that our networks performs favorably on the much more challenging *EV-IMO* dataset.

3. Event Clouds and Scene Motion

Event cameras record a continuous stream of brightness change events. Each event is encoded by its pixel position x, y, timestamp, t, accurate to microseconds, and an additional bit denoting whether brightness increased or decreased. We will refer to the events in a fixed-time interval as 'slice' of events. A single slice can contain millions of events (practically, 10^6 events per second in the EV-IMO [25] dataset collected with the DAVIS 346C sensor, and $6 * 10^6$ events with a newer Prophesee high resolution sensor [28]). The events are generated by the motion of image contours (object boundaries, or texture edges) over time. We can look at these events as points on trajectories $tj(t) = (x(t), y(t), t)$ - this makes it natural to regard events as points in 3D (x, y, t) space. The rest of this section is devoted to analyzing some geometric properties of event clouds. Since pixel motion is constrained by the laws of physics, the rigidity of bodies and the epipolar geometry, event clouds are significantly different from 3D point clouds in (x, y, z) space.

3.1. Surface Normals, Normal Flow, & Optical Flow

Earlier works [5, 10] have analyzed surface normals of event clouds in the context of optical flow estimation. With some abuse of notation, similar as in [5], we describe the event cloud as a function $t(x, y)$, with t the time and x, y the spatial coordinates, and consider it a surface (actually an (x, y) may map to multiple t violating the definition of a function). Then the partial derivates $\frac{\partial t(x,y)}{\partial x}$, $\frac{\partial t(x,y)}{\partial y}$ provide the inverse of the observed velocity components. Thus, from the normal to this surface n with components (n_x, n_y, n_t), the *instantaneous normal flow* at an image pixel can be obtained as $v_n = (\frac{n_t}{n_x}, \frac{n_t}{n_y})$.

More importantly though is the notion that the full flow of a point $p = (x, y, t)$ would lie in the plane with the normal n and passing through p. Now, we can impose an additional assumption, which is often called *smoothness constraint* - that flow in the local region is similar; this is not true for the boundaries, and we analyze boundary regions separately below.

More specifically, for a region of a small radius r around the point $p_0 = (x, y, t)$ we will assume that all points $p_i \in B_r(p_0)$ have the same (normalized) optical flow vector $v = (v_x, v_y, v_t)$. This results in a constraint: *the optical flow at p_0 lies at the intersection of all local planes given by normals n_i and passing through p_0*. If the assumption holds

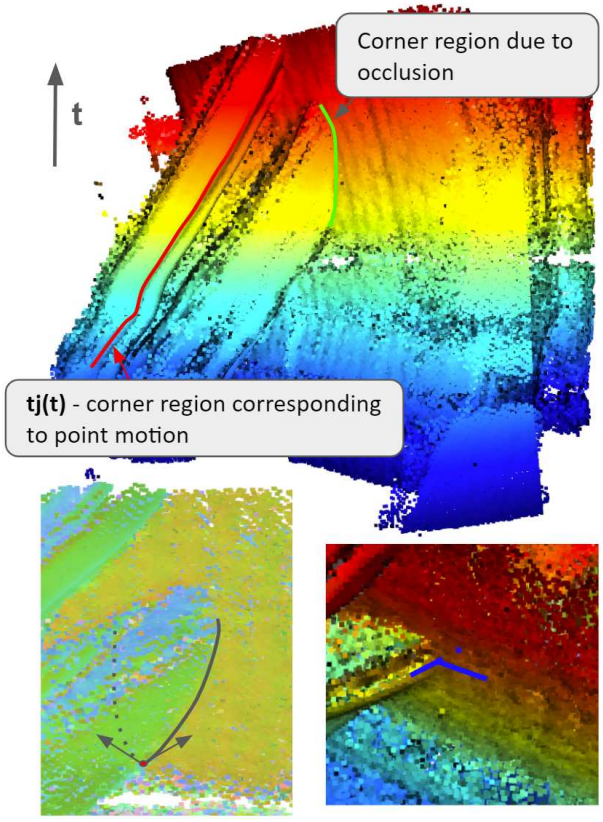


Figure 2. *Event cloud in (x,y,t) space; the color gradient corresponds to event timestamps. For a single rigid object, curvatures on the cloud would define a trajectory of a point of this object, while for a pair of objects such structure can identify an occlusion. On the bottom left - a point with two possible optical flow values, which only occur during occlusions. Bottom right: T-corner; during occlusion its shape defines which object is on the foreground and which is on the background (see Sec. 3.3)*

true, and the flow is the same for all local points, then all planes n_i will intersect on a single line v. The intersection of two planes given by n_i and n_j is simply $v = n_i \times n_j$. Given a set of planes in $B_r(p_0)$, we formulate the constraint on flow as a least squares problem:

$$v = min_v \sum ||v \times n_i||^2 \quad (1)$$

Here, we deviate from the common notion of optical flow as a $2D$ vector field; given the frame-less nature of the event stream, v should rather be thought of as the temporal derivative of the pixel trajectory, while instantaneous flow in the traditional sense can be written as $v_{xy} = (\frac{v_x}{v_t}, \frac{v_y}{v_t})$.

3.2. Continuity in Time

The flow is the derivative of the point trajectory $v = \frac{d(tj)}{dt} = (\frac{dx}{dt}, \frac{dy}{dt}, 1)$. Due to the *continuity in time* of the event stream, sometimes it is possible to recover the full trajectory of a point. A trajectory is unambiguously recoverable if all its points are *generalized corners* - that is, for all points Eq. 1 needs to have a single minima. Then, using a sequence of candidate points p_i and their corresponding flow v_i, it is possible to extract the full trajectories of these points. Even, if the data is discontinuous, we may be able to obtain the trajectory.

3.3. Boundary Regions and Occlusions

Corner detection for event cameras has been studied previously [22]. However, existing methods don't allow to distinguish between object corners, and structures caused by occlusions (see Fig. 2). Next we discuss how to distinguish these cases.

A consequence of Eq. 1 is that every point with nonzero curvature will have an unambiguous optical flow vector associated with it. If a point region $B_r(p_0)$ includes events from boundaries of two separate objects (which can happen during occlusions) Eq. 1 will have multiple distinct minima. An example is shown in Fig. 2: on the top - a typical occlusion of the background (shown as large, flat cluster of points) by a smaller foreground object (shown as a 'tube' cutting through the background motion plane).

To *distinguish* between a corner corresponding to a point trajectory (shown in red in Fig. 2) and a corner caused by the occlusion (shown in green), it is sufficient to analyze the local distribution of optical flow vectors. On the bottom left - the red point has multiple possible flow vectors, and its trajectory $\frac{d(tj)}{dt} = \infty$. These special points can be found, and corresponding corner regions labeled as *occlusion boundaries*.

It is also possible to extract at the occlusion, information about which of the two object is in the foreground, and which is in the background. Since due to occlusion the texture on theoccluded object is not visible, the foreground edge surface will be *hollow* inside and the background surface will be intersected by the foreground one. On the occlusion boundary this will always result in a T-shaped corner, shown in Fig. 2, bottom right.

4. The Architecture

4.1. Motion Segmentation in 3D

State-of-the-art 3D point cloud segmentation networks such as PointNet++ [30], EdgeConv [35], and 3DCNN [14] have been designed to extract static 3D feature descriptors in a uniform 3D metric space. The event cloud differs in that it has a temporal axis; the motion of the object itself controls the shape of the cloud and hence static (x, y, t) features cannot be learnt as descriptors. The metric space of depth data also allows for efficient downsampling and mesh simplifications, and modern depth sensors have significantly less noise than event cameras. Currently, only a handful of methods is capable of handling millions of points produced by event cameras. In this work, we use PointNet++ as a baseline and we develop a network using a Graph Convolutional Network, to perform the task of background-foreground motion segmentation.

4.2. Network Design

Our network architecture is shown in Figure 3 - it consists of five consecutive Graph Convolutional (GConv) layers and three fully connected hidden layers, which share the same weights across all points and perform global feature aggregation. The input to the network is an unstructured graph which consists of events in (x, y, t) space as nodes, the per-point surface normals (n_x, n_y, n_t), and the graph edges, which are computed as described in Sec. 4.3.

In each GConv layer, every point feature is aggregated from its neighbours, making points with similar vertices clustered together. When training multiple GConv layers, this is equivalent to a multi-scale clustering of the point features, which also preserves local geometric structures [23]. Each GConv layer has 64 input channels, and maps the features to 64 output channels at different scales across the entire graph [23]. Then, the 5 sets of 64 multi-scale features extracted by the five GConv layers are concatenated and fed to a Multi Layer Perceptron (MLP) classifier. The MLP starts with 256 initial channels and reduces the channels by a rate of 4 into 16 channels in the last hidden layer. Then, the outputs of the MLP are connected to a fully connected layer to produce a single point-wise score.

We supervise our training as a regression problem instead of a classification problem to decrease the possibility of overfitting to object contours. The raw response values are compared with the point-wise ground truth labels $\in [0, 1]$ by Binary Cross Entropy with Logits Loss.

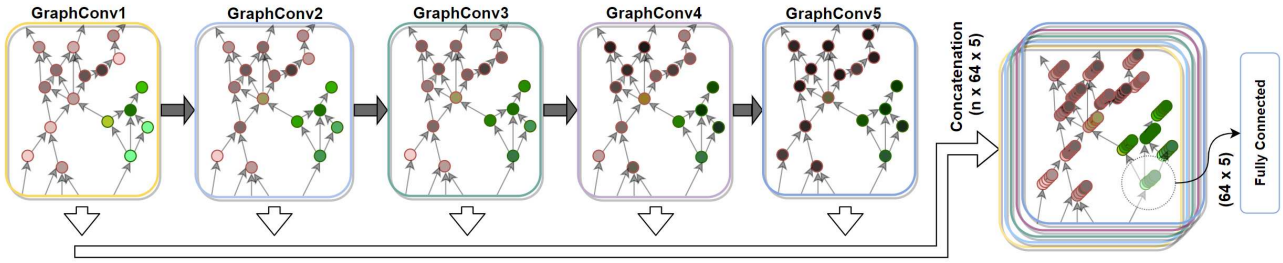


Figure 3. *An illustration of the architecture. 5 Graph Convolutional layers aggregate multiscale features around each point; the features are concatenated and fed into a fully connected layer to predict the point class.*

4.3. Edge Computation

A central component of Graph Convolutional neural networks is the edge computation. If using the raw data, the high density of points in the event cloud would give rise to a very large number of edges, making large edge radii prohibitively expensive to compute and use. To get around this, while preserving connections with neighbouring points along possible motion trajectories, we filter the edges in a sphere radius r so that they are parallel to event-surface. Given a point p_0 and its normal n_0, we keep only the edges orthogonal to n_0, with a certain filtering threshold α, as described in Eq. 2-(a) (and illustrated in Fig. 4-(b))

$$\begin{aligned}(a) &: \{p_i | \forall p_i \in B_r(p_0) \quad \& \quad (p_i - p_0) \cdot n_0 < \alpha\} \\ (b) &: \{p_i | \forall p_i \in B_r(p_0) \quad \& \quad p_i^t > p_0^t\} \end{aligned} \tag{2}$$

Since most of an event's temporal motion information is contained within a plane parallel to time-surface, this filtering strategy is a good trade-off between the richness of surface features and the computational performance. Yet, our experiments have shown that most surface patches are rather isolated in the absence of strong texture or extremely fast motion, and in practice the filtering is not required.

As a second constraint (Eq. 2-(b)), we impose the points to lie in the upper (along temporal axis) hemisphere of a point - this halves the number of edges with little to no decrease in network performance. Using both constraints, we obtain sparse edges which will maximally *align with possible local optical flow values* (see Sec. 3). In our experiments we use $r = 10$ pixels (the scaling along the temporal axis is described in Sec. 5.1.1), and we perform an ablation study on the radius size (Fig. 7(b)).

4.4. Temporal Augmentation

Following the intuitions discussed in Sec. 3, we use a *temporal augmentation* to artificially introduce variations in object speed (and hence - cloud shape) and improve the generalization to varying motion. Given a point in an input cloud $p = (x, y, t)$ with a normal $n = (n_x, n_y, n_t)$ the augmented point p' and its normal n' are computed as:

$$p' = (x, y, \alpha * t) \quad n' = (\alpha * n_x, \alpha * n_y, n_t), \tag{3}$$

where α is a random scaling parameter (in our work, $0.8 - 1.2$) and n' is normalized to unit length.

The motivation for this method is that the essential cue characterizing separate objects does not depend on speed, but rather on spatio-temporal *anomalies* - such as T-corners, points with multiple flow values or the continuity of the event cloud. The temporal augmentation forces the neural network to learn such features (which remain unaffected by augmentation), and reduces overfitting to the local shape of the cloud. Our ablation studies (in Sec. 5.3) show that the model generalizes better using this augmentation.

5. Experiments

5.1. Dataset

EV-IMO is the only publicly available event-based segmentation dataset. It includes pixelwise masks at 200Hz and roughly 30 minutes of recording, although the dataset was collected in a single room and only includes three objects. Therefore segmentation might be prone to overfitting. Since our pipeline is not trained on depth, overfitting to room structure is unlikely. The 3D shape of the objects varies a lot, depending on object motion, but overfitting on contours is still possible. We perform an ablation study in the supplementary material, where we train a NN using only sequences with 2 of the objects and evaluate on sequences with the third object.

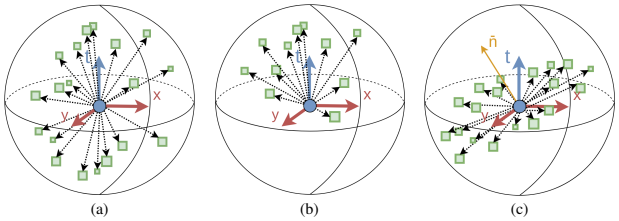


Figure 4. *Illustration of edge configurations: (a) a typical configuration used in 3D processing: all points are used to create edges; (b) and (c) our configuration: using only points in the upper hemisphere (b); (c) using edges parallel to the time-surface (c).*

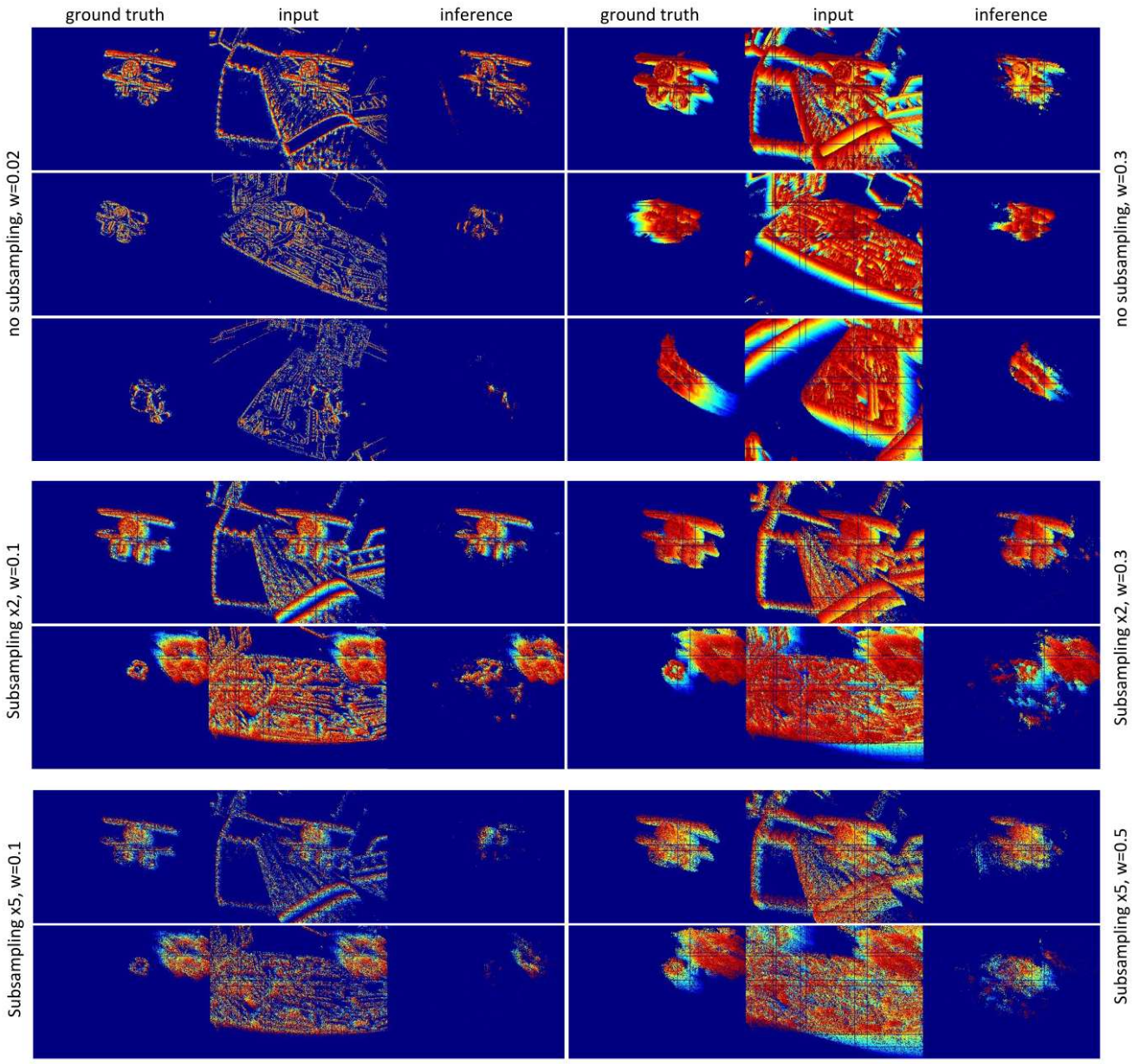


Figure 5. *Qualitative results - the event cloud was projected on a plane, color represents the normalized timestamp. The subsampling rate increases from top to bottom; small time slices are shown on the left, and large on the right. Experiments with* 0.5 *second slices could be achieved only with a downsampling factor of* $u = 5$*, but the dramatic decrease in feature quality yields poor results compared to the* 0.3 *second slice with no subsampling. Note how the results improve for larger slice width, especially in the presence of texture.*

5.1.1 Data Preprocessing

We preprocess the raw data of EV-IMO. We upscale the temporal axis of the event cloud by a factor of 200 to keep the density of events more uniform across the x,y,t axes. The normals are precomputed using PCA, with $r = 5$ (after temporal upscaling). We also apply a radius outlier filter with $r = 3$ and $k = 30$ - which removes all points having less than 30 points in a radius of 3 pixels; this results in approximately 10% to 15% points removed. To further reduce the number of points, we use random subsampling with a factor u, which keeps 1 random point out of u (in practice, we use $u = 1$ for no subsampling or $u = 2$, to remove half the points). We precompute edges within a radius of 10 pixels for all points, and (optionally) keep up to 30 edges connected to nearest points.

The ground truth provided in EV-IMO is image-based, sampled at $200Hz$. For every event, we approximate its class by locating the nearest class mask according to mask and event timestamps and downprojecting the event coordi-

nate onto the image. This produces good results in practice, even for fast motion; we show a sample of the event cloud with class annotations in the supplementary material.

5.2. Implementation Details

Our network architecture consists of five consecutive Graph Convolutional(GC) layers and three fully connected hidden layers. As input features we use spatio-temporal event locations (x, y, t), time-surface normals (n_x, n_y, n_t), and a per-event binary polarity value $(p \in \begin{bmatrix} 0, & 1 \end{bmatrix})$ which signifies whether the brightness has increased or decreased at pixel (x, y), at time t.

In our current implementation the network (Sec. 4.2) has $117k$ trainable parameters. We train the network with a batch size of 3 using three Nvidia GTX 1080Ti GPUs. Larger batch sizes are difficult to achieve in practice because of the large amount of data. We use the Adam optimizer with a learning rate of $7e-4$ and cosine annealing scheduling strategy. We train our models for 200 epochs on a portion of the $EV-IMO$ [25] dataset. The model takes 6-minutes to train for each epoch with a slice width of 0.3 seconds and no subsampling.

As baseline we use the state-of-the-art *PointNet++* [30] on event clouds. We take the segmentation implementation of *PointNet++* presented in [30], which in turn was inspired by [14] and [29]. The implementation consists of two hierarchical sampling and grouping modules, followed by one k-nearest-neighbours interpolation and three forward passing modules. K-nearest-neighbours are searched with a ratio and a radius of $0.2/0.2$ in the first sampling layer and $0.25/0.4$ in the second sampling layer. The responses from the last forward passing modules are passed to three concatenated layers of a fully connected network to produce as out a label for each point.

5.3. Ablation Studies

5.3.1 Effects of the Slice Width

We conduct experiments to investigate the effects of slice width and cloud sub-sampling on the performance of the network. Fig. 6 shows results on the *boxes* validation set for the first $12k$ iterations of training. The experiments on the full resolution clouds (shown in solid lines) perform notably better, as to be expected, but the training speed is 30% lower (for results on training speed see Table 2).

We observe that the largest effect was for larger slice widths - the $u2, w0.1$ and $u1, w0.1$ perform similarly, while $u1, w0.3$ is 10% better than $u2, w0.3$. Our intuition is that this due to the lack of temporal features in smaller slices - a similar effect is observed when varying the edge radius (Fig. 7(b)). The edge radius, together with the network depth, essentially controls the maximum size of global features.

5.3.2 Temporal Augmentation

Temporal augmentation is designed to reduce overfitting of the network to the local 3D feature descriptors by randomly scaling the event cloud along time axis. The *EV-IMO* dataset has high variation in object velocities, including across training and validation sets; the velocity of the object defines the shape of the cloud (as explained in Sec. 3.1), which can cause significant drop in scores when transferring to unseen sequences in validation set. We show the comparison for learning with and without augmentation in Fig. 7-(a) by training on the *boxes* dataset and evaluation on *fast* dataset.

5.4. Results

Both $GConv$ and $PointNet++$ were trained on all *EV-IMO* train sequences containing different objects, which

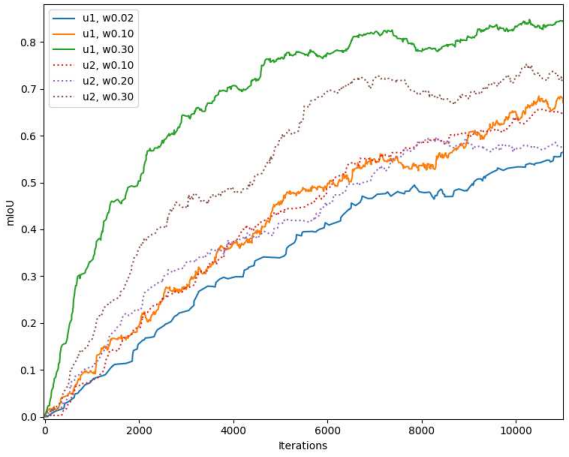


Figure 6. *Inference performance, in $mIoU$, for the first $12k$ iterations of training for slice widths $w = 0.02, 0.1, 0.3$ sec. Dashed line corresponds to a subsampling factor $u = 2$. The solid line corresponds to experiments with no subsampling ($u = 1$).*

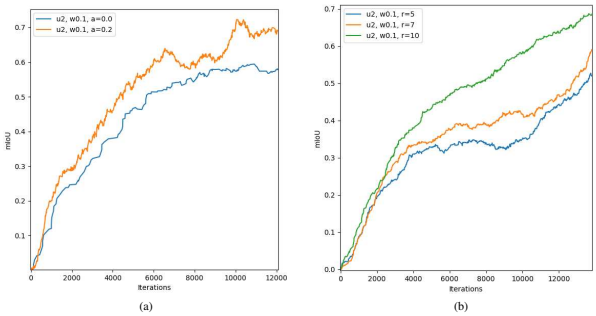


Figure 7. *$mIoU$ scores for the box validation for varying edge radius (without temporal augmentation). (a) Plots are shown for $\alpha = 0.0$ (no augmentation) and $\alpha = 0.2$ for the first $12k$ iterations of training. (b) Reducing the maximum edge radius from 10 to 7 and 5, results in a significant performance drop.*

	boxes			floor			wall			table			fast		
	0.3	0.1	0.02	0.3	0.1	0.02	0.3	0.1	0.02	0.3	0.1	0.02	0.3	0.1	0.02
GConv	84±9	81±8	60±18	80±9	79±7	55±19	**85±8**	83±4	51±16	**87±7**	80±7	57±14	**77±10**	74±17	39±19
$GConv_{u2}$	**85±5**	70±11	54±10	**81±12**	69±8	52±14	83±4	71±9	61±17	85±11	77±19	59±19	74±19	69±24	37±11
$PointNet++$ [30]	69±17	71±22	80±15	66±19	68±18	76±10	71±17	75±19	74±20	59±22	62±28	68±23	21±12	24±10	20±6
$EV-IMO$ [25]		70±5			59±9			78±5			79±6			67±3	

Table 1. Segmentation results on EV-IMO event-based dataset. Metric is $mIoU(\%)$ on points; the 3D methods were evaluated on 3 different time-slice widths: 0.3, 0.1 and 0.02 seconds. The best results are shown in bold for every separate validation set type.

are called *boxes*, *floor*, *wall* and *table*. They were evaluated separately on *boxes*, *floor*, *wall*, *table* and the *fast* validation set with a 0.02 sec. temporal step between slices, $\alpha = 0.2$ for temporal augmentation (disabled for validation), subsampling rates of $u = 1$ and $u = 2$ and slice widths $w = 0.02$, 0.1 and 0.3 sec. The quantitative results are presented in Table 5.2.

For the EV-IMO method, which outputs dense masks, we set the slice width to $0.025sec$. We project the inferenced masks on the event cloud (similar to how we handle ground truth, Sec. 5.1.1) and compute IoU scores on the labeled event clouds.

5.4.1 Qualitative Results

Fig. 5 shows qualitative results (for visualization events are projected the along time axis). The color encodes the timestamp (blue is 0, red is slice width). The top section of the figure compares results with no subsampling for a 0.02 slice and a 0.3 slice. Note how the increase in the size of the slice improves the segmentation quality even in textured regions. The subsampling factor $u = 2$, which removes half the points from the event cloud, also produces high quality results, but is more prone to false positives. For the bottom section of the figure, 80% of the points were removed. The quality of input suffers significantly, with many spatial and temporal features lost, and the network performs poorly in high-textured regions.

	GConv		*PointNet++*		
params	117.5k		1.4M		
u	1	2	2	5	$\#pts$
20ms	0.016	0.012	0.219	0.134	23073
0.1s	0.109	0.048	4.260	0.663	170503
0.3s	0.275	0.140	22.93	6.792	417315

Table 2. Forward time (seconds) for different uniform downsampling and temporal slice widths.

5.4.2 Performance Considerations

We present the execution times for our GConv and for PointNet++ for different time slice widths and event cloud subsampling factors in Table 2, measured on a single NVIDIA GTX1080Ti. In all experiments, PointNet++ is notably slower than GConv. However, starting from the 0.1 second slice, GConv is not real time anymore without subsampling. This problem can be addressed practically by constructing a lookup table [31] and avoid recomputing features when sequential time slices overlap.

6. Limitations and Future Work

Our approach operates on large slices in time - this allows the neural network to observe a history of scene motion, and potentially make better decisions based on the global temporal features - all without relying on LSTM-like approaches (which essentially memorize scene content). On the other hand, modern event-based cameras are capable of generating up to 10^7 events per second, and with increased resolution, the amount of local edge connections between graph nodes can grow exponentially. In this work, we were able to achieve slice widths of up to 0.3 seconds (and 0.5 seconds with severe downsampling) due to the large amount of memory consumed by 3D points. A dedicated GPU-optimized module could alleviate many of the shortcomings we have witnessed. We also observe that edges are useful mostly for extracting local features, and hence a dual neural network - one to extract local features on thin temporal slices and another to extract temporal features on large slices will be among our future efforts.

7. Acknowledgement

This work was funded by a fellowship to A.M. from Prophesee SA, the National Science Foundation under grant BCS 1824198 and ONR under grant N00014-17-1-2622.

8. Conclusion

We have described spatial and temporal features of event clouds, which provide cues for motion tracking and segmentation. Our novel segmentation pipeline inherently captures these features and is able to perform well with rapid, 6 dof motion of the camera and objects on the scene. The graph-based approach allows for the first time to use wide slices as a single input, with inference speeds of up to 0.02 seconds. We believe that our contribution is a step forward towards understanding the geometric properties of event clouds and making a more complete use of the information provided by event cameras.

References

[1] Edward H Adelson and James R Bergen. Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Am. A*, 2(2), 1985. 1

[2] Francisco Barranco, Cornelia Fermüller, and Yiannis Aloimonos. Contour motion estimation for asynchronous event-driven cameras. *Proceedings of the IEEE*, 102(10):1537–1556, 2014. 2

[3] Francisco Barranco, Cornelia Fermüller, and Yiannis Aloimonos. Bio-inspired motion estimation with event-driven sensors. In *International Work-Conference on Artificial Neural Networks*, pages 309–321. Springer, 2015. 2

[4] Francisco Barranco, Ching L Teo, Cornelia Fermüller, and Yiannis Aloimonos. Contour detection and characterization for asynchronous event sensors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 486–494, 2015. 3

[5] R. Benosman, C. Clercq, X. Lagorce, Sio-Hoi Ieng, and C. Bartolozzi. Event-based visual flow. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(2):407–417, 2014. 2, 3

[6] Robert C. Bolles, H. Harlyn Baker, and David H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1(1):7–55, Mar 1987. 2

[7] C. Brandli, R. Berner, M. Yang, S. Liu, and T. Delbruck. A 240 180 130 db 3 s latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, Oct 2014. 1

[8] Tobias Brosch, Stephan Tschechne, and Heiko Neumann. On event-based optical flow detection. *Frontiers in neuroscience*, 9:137, 2015. 2

[9] Thusitha N Chandrapala and Bertram E Shi. Invariant feature extraction from event based stimuli. In *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 1–6. IEEE, 2016. 2

[10] Xavier Clady, Charles Clercq, Sio-Hoi Ieng, Fouzhan Houseini, Marco Randazzo, Lorenzo Natale, Chiara Bartolozzi, and Ryad B. Benosman. Asynchronous visual event-based time-to-contact. In *Front. Neurosci.*, 2014. 2, 3

[11] Cornelia Fermüller, David Shulman, and Yiannis Aloimonos. The statistics of optical flow. *Computer Vision and Image Understanding*, 82(1):1–32, 2001. 1

[12] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3867–3876, 2018. 2

[13] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, June 2012. 1

[14] A. Szlam J. Bruna, W. Zaremba and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv:1312.6203v3*, 2017. 4, 7

[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *The International Conference on Learning Representations 2017*, 2016. 3

[16] Jan J Koenderink. Optic flow. *Vision research*, 26(1):161–179, 1986. 1

[17] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E Shi, and Ryad B Benosman. Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1346–1359, 2016. 2

[18] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman. Hots: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1346–1359, July 2017. 1, 2

[19] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128 x 128 at 120db 15 micros latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008. 1

[20] Min Liu and Tobi Delbruck. Block-matching optical flow for dynamic vision sensors: Algorithm and fpga implementation. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017. 2

[21] Hugh Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981. 1

[22] Jacques Manderscheid, Amos Sironi, Nicolas Bourdis, Davide Migliore, and Vincent Lepetit. Speed invariant time surface for learning to detect corner points with event-based cameras. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 4

[23] Pierre Vandergheynst Michaël Defferrard, Xavier Bresson. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems 29 (2016)*, 2017. 4

[24] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking. *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018. 2

[25] Anton Mitrokhin, Chengxi Ye, Cornelia Fermüller, Yiannis Aloimonos, and Tobi Delbrück. EV-IMO: motion segmentation dataset and learning pipeline for event cameras. *CoRR*, abs/1903.07520, 2019. 2, 3, 7, 8

[26] Elias Mueggler, Christian Forster, Nathan Baumli, Guillermo Gallego, and Davide Scaramuzza. Lifetime estimation of events from dynamic vision sensors. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4874–4881. IEEE, 2015. 2

[27] Garrick Orchard, Ryad Benosman, Ralph Etienne-Cummings, and Nitish V Thakor. A spiking neural network architecture for visual motion estimation. In *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 298–301. IEEE, 2013. 2

[28] C. Posch, D. Matolin, and R. Wohlgenannt. A QVGA 143 db dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain cds. *IEEE Journal of Solid-State Circuits*, 46(1):259–275, 2011. 1, 3

[29] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3, 7

[30] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 5105–5114, USA, 2017. Curran Associates Inc. 2, 4, 7, 8

[31] Yusuke Sekikawa, Kosuke Hara, and Hideo Saito. Eventnet: Asynchronous recursive event processing. *CoRR*, abs/1812.07045, 2018. 3, 8

[32] B. Son, Y. Suh, S. Kim, H. Jung, J. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, Y. Roh, H. Lee, Y. Wang, I. Ovsiannikov, and H. Ryu. 4.1 a 640480 dynamic vision sensor with a 9m pixel and 300meps address-event representation. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 66–67, Feb 2017. 1

[33] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2432–2439. IEEE, 2010. 1

[34] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza. Feature detection and tracking with the dynamic and active-pixel vision sensor (davis). In *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–7, June 2016. 2

[35] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ArXiv*, abs/1801.07829, 2018. 4

[36] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019. 3

[37] Chengxi Ye, Anton Mitrokhin, Cornelia Fermüller, James A. Yorke, and Yiannis Aloimonos. Unsupervised learning of dense optical flow, depth and egomotion from sparse event data. *CoRR*, abs/1809.08625v2, 2018. 3

[38] Alex Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. 06 2018. 3

[39] A. Z. Zhu, N. Atanasov, and K. Daniilidis. Event-based feature tracking with probabilistic data association. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4465–4470, May 2017. 2

[40] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. *CoRR*, abs/1812.08156, 2018. 3