This paper has been accepted for publication at the
IEEE International Conference on Computer Vision (ICCV), Seoul, 2019. ©IEEE

# Event-Based Motion Segmentation by Motion Compensation

Timo Stoffregen[1,2], Guillermo Gallego[3], Tom Drummond[1,2], Lindsay Kleeman[1], Davide Scaramuzza[3]

[1]Dept. Electrical and Computer Systems Engineering, Monash University, Australia.
[2]Australian Centre of Excellence for Robotic Vision, Australia.
[3]Dept. Informatics (Univ. Zurich) and Dept. Neuroinformatics (Univ. Zurich & ETH Zurich), Switzerland.

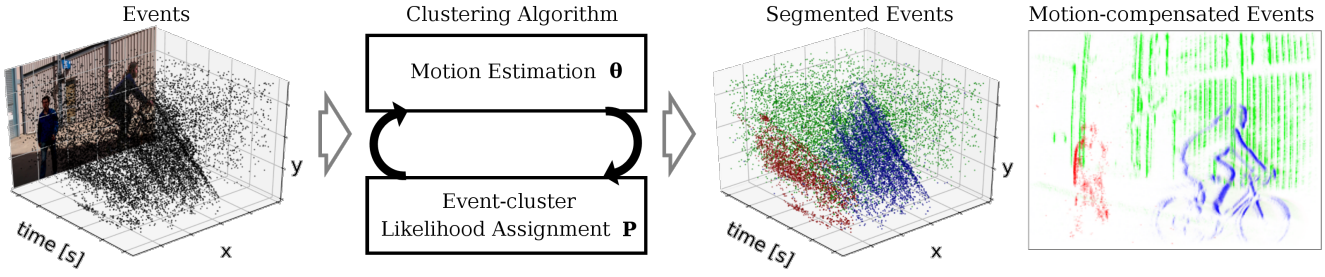



Figure 1: Our method segments a set of events produced by an event-based camera (Left, with color image of the scene for illustration) into the different moving objects causing them (Right: pedestrian, cyclist and camera's ego-motion, in color). We propose an iterative clustering algorithm (Middle block) that jointly estimates the motion parameters $\boldsymbol{\theta}$ and event-cluster membership probabilities $\mathbf{P}$ to best explain the scene, yielding motion-compensated event images on all clusters (Right).

## Abstract

*In contrast to traditional cameras, whose pixels have a common exposure time, event-based cameras are novel bio-inspired sensors whose pixels work independently and* asynchronously *output intensity changes (called "events"), with microsecond resolution. Since events are caused by the apparent motion of objects, event-based cameras sample visual information based on the scene dynamics and are, therefore, a more natural fit than traditional cameras to acquire motion, especially at high speeds, where traditional cameras suffer from motion blur. However, distinguishing between events caused by different moving objects and by the camera's ego-motion is a challenging task. We present the first per-event segmentation method for splitting a scene into independently moving objects. Our method jointly estimates the event-object associations (i.e., segmentation) and the motion parameters of the objects (or the background) by maximization of an objective function, which builds upon recent results on event-based motion-compensation. We provide a thorough evaluation of our method on a public dataset, outperforming the state-of-the-art by as much as* $10\,\%$*. We also show the first quantitative evaluation of a segmentation algorithm for event cameras, yielding around* $90\,\%$ *accuracy at* $4$ pixels *relative displacement.*

## Supplementary Material

Accompanying video: https://youtu.be/0q6ap_OSBAk. We encourage the reader to view the added experiments and theory in the supplement.

## 1. Introduction

Event-based cameras, such as the Dynamic Vision Sensor (DVS) [1, 2], are novel, bio-inspired visual sensors. In contrast to conventional cameras that produce images at a fixed rate, the pixels in an event-based camera operate independently and asynchronously, responding to intensity changes by producing *events*. Events are represented by the $x, y$ pixel location and timestamp $t$ (in microseconds) of an intensity change as well as its polarity (i.e., whether the pixel became darker or brighter). Since event-based cameras essentially sample the scene at the same rate as the scene dynamics, they offer several advantages over conventional cameras: very high temporal resolution, low latency, very high dynamic range (HDR, $140\,\mathrm{dB}$) and low power and bandwidth requirements - traits which make them well suited to capturing motion. Hence, event-based cameras open the door to tackle challenging scenarios that are inaccessible to traditional cameras, such as high-speed and/or HDR tracking [3–6], control [7–9] and Simultaneous Localization and Mapping (SLAM) [10–13]. Due to their principle of operation and unconventional output, these cameras

represent a paradigm shift in computer vision, and so, new algorithms are needed to unlock their capabilities. A survey paper on event-based cameras, algorithms, and applications, has been recently published in [2].

We consider the problem of segmenting a scene viewed by an event-based camera into independently-moving objects. In the context of traditional cameras, this problem is known as *motion segmentation* [14], and it is an essential pre-processing step for several applications in computer vision, such as surveillance, tracking, and recognition [15]. Its solution consists of analyzing two or more consecutive images from a video camera to infer the motion of objects and their occlusions. In spite of progress in this field, conventional cameras are not ideally suited to acquiring and analyzing motion; since exposure time is globally applied to all pixels, they suffer from motion blur in fast moving scenes. Event-based cameras are a better choice since they sample at exactly the rate of scene dynamics, but conventional techniques cannot be applied to the event data.

Motion segmentation in the case of a static event-based camera is simple, because in this scenario events are solely due to moving objects (assuming there are no changes in illumination) [16–18]. The challenges arise in the case of a moving camera, since in this scenario events are triggered everywhere on the image plane, produced by both the moving objects as well as the apparent motion of the static scene induced by the camera's ego-motion. Hence, event-based motion segmentation consists of classifying each event into a different object, including the background. However, each event carries very little information, and therefore it is challenging to perform the mentioned per-event classification.

We propose a method to tackle the event-based motion segmentation problem in its most general case, with a possibly moving camera. Inspired by classical layered models [19], our method classifies the events of a space-time window into separate clusters (i.e., "layers"), where each cluster represents a coherent moving object (or background) (see Fig. 1). The method jointly estimates the motion parameters of the clusters and the event-cluster associations (i.e., likelihood of an event belonging to a cluster) in an iterative, alternating fashion, using an objective function based on motion compensation [20, 21] (basically, the better the estimated unknowns, the sharper the motion-compensated event image of each cluster). Our method is flexible, allowing for different types of parametric motions of the objects and the scene (translation, rotation, zooming, etc.).

**Contributions.** In summary, our contributions are:

- A novel, iterative method for segmenting multiple objects based on their apparent motion on the image plane, producing a per-event classification into space-time clusters described by parametric motion models.
- The detection of independently moving objects without having to compute optical flow explicitly. Thus, we circumvent this difficult and error-prone step toward reaching the goal of motion-based segmentation.
- A thorough evaluation in challenging, real-world scenarios, such as high-speed and difficult illumination conditions, which are inaccessible to traditional cameras (due to severe motion blur and HDR), outperforming the state-of-the-art by as much as $10\,\%$, and showing that accuracy in resolving small motion differences between objects is a central property of our method.

As a by-product, our method produces sharp, motion-compensated images of warped events, which represent the appearance (i.e., shape or edge-map) of the segmented objects (or background) in the scene (Fig. 1, Right).

The rest of the paper is organized as follows: Section 2 reviews related work on event-based motion segmentation, Section 3 describes the proposed solution, which is then evaluated in Section 4. Conclusions are drawn in Section 5.

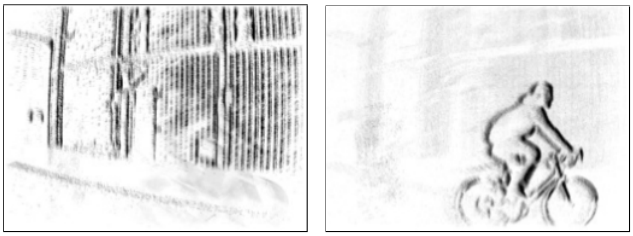
## 2. Related Work

Event-based motion segmentation in its non-trivial form (i.e., in the presence of event-clutter caused by camera ego-motion, or a scene with many independently moving, overlapping objects) has been addressed before [22–26].

In [22], a method is presented for detection and tracking of a circle in the presence of event clutter. It is based on the Hough transform using optical flow information extracted from temporal windows of events. Segmentation of a moving object in clutter was also addressed in [23]. It considered more generic object types than [22] by using event corners as primitives, and it adopted a learning technique to separate events caused by camera motion from those due to the object. However, the method required the additional knowledge of the robot joints controlling the camera.

Segmentation has been recently addressed by [24,25] using the idea of motion-compensated event images [20, 21, 27–30]. For example, [24] first fitted a motion compensation model to the dominant events, then removed these and fitted another motion model to the remaining events, greedily. Similarly, [25] detected moving objects in clutter by fitting a motion-compensation model to the dominant events (i.e., the background) and detecting inconsistencies with respect to that motion (i.e., the objects). The objects were then "segmented" via morphological operations on the warped image, and were used for tracking. The method could handle occlusions during tracking, but not during detection.

Our method differs from [22] in that we demonstrate segmentation on objects with arbitrary shapes, and from [23] in that we do not require additional inputs (e.g., robot joints). Our work is most related to [24, 25]; however, it has the following novelties: (*i*) it actually performs *per-event* segmentation, rather than just providing bounding boxes for detected object regions, (*ii*) it allows for general parametric motion models (as those in [20]) to describe each clus-



(a) Cluster 1 (b) Cluster 2

(c) Cluster 3 (d) All clusters (merged IWEs)

Figure 2: Our method splits the events into clusters (one per moving object), producing motion-compensated images of warped events (IWEs), as shown in (a)-(c) for the three objects in the scene of Fig. 1. The likelihood of each event is represented by the darkness of the pixel. Since the likelihoods are nonzero, "ghosts" can be seen in the individual clusters. IWEs in (a)-(c) are merged into a single image (d), using a different color for each cluster. Segmented events in upcoming experiments are shown using this colored motion-compensated image representation.

ter, (*iii*) it performs optimization based on a single objective function (as opposed to two sequential optimization criteria in [25]), (*iv*) it is able to handle occlusions between objects at any point in time. The proposed method, is, to our knowledge, the first one that *jointly* estimates the apparent motion parameters of all objects in the scene and the event-object associations (i.e., segmentation). It does so by leveraging the idea of motion-compensated event images within an iterative, alternating optimization approach, in an expectation-maximization (EM) fashion.

# 3. Methodology

Our method is inspired by the combination of classical layered models [19] and event-based motion compensation [20]. In the following, we review the working principle of event-based cameras, describe the motion segmentation problem addressed and present our proposed solution.

**Event-Based Camera Working Principle.** Event-based cameras, such as the DVS [1], have independent pixels that output "events" in response to intensity changes. Specifically, if $L(\mathbf{x},t) \doteq \log I(\mathbf{x},t)$ is the logarithmic intensity at pixel $\mathbf{x} \doteq (x,y)^\top$ on the image plane, the DVS generates an event $e_k \doteq (\mathbf{x}_k, t_k, s_k)$ if the change in intensity at pixel $\mathbf{x}_k$ reaches a threshold $C$ (e.g., 10-15% relative change):

$$\Delta L(\mathbf{x}_k,t_k) \doteq L(\mathbf{x}_k,t_k) - L(\mathbf{x}_k,t_k-\Delta t_k) = s_k\, C, \quad (1)$$

where $t_k$ is the timestamp of the event, $\Delta t_k$ is the time since the previous event at the same pixel $\mathbf{x}_k$ and $s_k \in \{+1,-1\}$ is the polarity of the event (the sign of the intensity change).

## 3.1. Problem Statement

Since each event carries little information and we do not assume prior knowledge of the scene, we process events in packets (or groups) to aggregate sufficient information for estimation. Specifically, given a packet of events $\mathcal{E} \doteq \{e_k\}_{k=1}^{N_e}$ in a space-time volume of the image plane $V \doteq \Omega \times T$, we address the problem of classifying them into $N_\ell$ clusters (also called "layers"), with each cluster representing a coherent motion, of parameters $\boldsymbol{\theta}_j$. We assume that $T$ is small enough so that the motion parameters of the clusters $\boldsymbol{\theta} \doteq \{\boldsymbol{\theta}_j\}_{j=1}^{N_\ell}$ are constant.

The images on both sides of the algorithm block in Fig. 1 illustrate the above-mentioned problem and its solution, respectively. Notice that, (*i*) since events have space-time coordinates, clusters are three-dimensional, contained in $V$, and (*ii*) since corresponding events (caused by the same point of a moving edge) describe point trajectories in $V$, optimal clusters should contain them, therefore, clusters have a "tubular" shape (Fig. 1, segmented events). Implicit in motion segmentation, if two objects share the same motion, they are segmented together, regardless of their location.

## 3.2. Summary of Proposed Solution

Leveraging the idea of motion compensation [20], we seek to separate the events $\mathcal{E}$ into clusters by maximizing *event alignment*, i.e., maximizing the sharpness of motion-compensated images (one per cluster) of warped events.

More specifically, the idea of motion compensation [20] is that, as an edge moves on the image plane, it triggers events on the pixels it traverses. The motion of the edge can be estimated by warping the events to a reference time and maximizing their alignment, producing a sharp Image of Warped Events (IWE) [20]. In the case of multiple objects with different motions, maximal event alignment cannot be achieved using a single warp, and so, several warps (i.e., motion models or "clusters") are required, as well as identifying which events belong to which object (i.e., "event-cluster associations"). This is the essence of our approach, which is illustrated in Figs. 1 and 2. Fig. 1 shows the events produced by three objects in a scene: a pedestrian, a cyclist and a the building facade (camera motion). Each object has a different motion and triggers events on the image plane as it moves. When events are warped to a reference time (e.g., $t_{\text{ref}} = 0$) according to a candidate motion model, they produce an IWE. If the candidate motion coincides with the true motion of the object causing the events, the warped events align, producing a sharp motion-compensated IWE, as shown in Fig. 2 using three different motion models

(one per object). Otherwise, they do not align, producing a blurred IWE. We use the sharpness of such IWE as the main cue to segment the events. Our method jointly identifies the events corresponding to each independently moving object as well as the object's motion parameters.

### 3.3. Mathematical Formulation

In contrast to previous methods [24, 25], we explicitly model event-cluster associations in the motion-compensation framework, i.e., $p_{kj} = P(e_k \in \ell_j)$ is the probability of the $k$-th event belonging to the $j$-th cluster. Let $\mathbf{P} \equiv (p_{kj})$ be an $N_e \times N_\ell$ matrix with all event-cluster associations. The entries of $\mathbf{P}$ must be non-negative, and each row must add up to one. Using these associations, we define the *weighted* IWE of the $j$-th cluster as

$$I_j(\mathbf{x}) \doteq \sum_{k=1}^{N_e} p_{kj}\, \delta(\mathbf{x} - \mathbf{x}'_{kj}), \tag{2}$$

with $\mathbf{x}'_{kj} = \mathbf{W}(\mathbf{x}_k, t_k; \boldsymbol{\theta}_j)$ the warped event location, and $\delta$ the Dirac delta. Equation (2) states that events are warped,

$$e_k \doteq (\mathbf{x}_k, t_k, s_k) \mapsto e'_k \doteq (\mathbf{x}'_k, t_{\text{ref}}, s_k), \tag{3}$$

and the values $p_{kj} \geq 0$ (i.e., weights) are accumulated at the warped locations $\mathbf{x}'_k$. Event alignment within the $j$-th cluster is measured using image contrast [31], which is defined by a sharpness/dispersion metric, such as the variance [20]:

$$\text{Var}(I_j) \doteq \frac{1}{|\Omega|} \int_\Omega (I_j(\mathbf{x}) - \mu_{I_j})^2 d\mathbf{x}, \tag{4}$$

where $\mu_{I_j}$ is the mean of the IWE over the image plane $\Omega$.

We propose to find the associations $\mathbf{P}$ and cluster parameters $\boldsymbol{\theta}$ that maximize the sum of contrasts of all clusters:

$$(\boldsymbol{\theta}^*, \mathbf{P}^*) = \arg\max_{(\boldsymbol{\theta}, \mathbf{P})} \sum_{j=1}^{N_\ell} \text{Var}(I_j). \tag{5}$$

Since the problem addressed does not admit a closed-form solution, we devise an iterative, alternating optimization approach, which we describe in the next section.

The pseudo-code of our method is given in Algorithm 1. From the output of Algorithm 1, it is easy to compute motion-compensated images of events corresponding to each cluster, i.e., the weighted IWEs (2) shown in Fig. 2. Each IWE shows the sharp, recovered edge patterns (i.e, and appearance model) of the objects causing the events.

### 3.4. Alternating Optimization

Each iteration of Algorithm 1 has two steps (lines 6 and 7), as in a coordinate ascent algorithm. If the associations $\mathbf{P}$ are fixed, we may update the motion parameters

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mu \nabla_{\boldsymbol{\theta}} \left( \sum_{j=1}^{N_\ell} \text{Var}(I_j) \right) \tag{6}$$

by taking a step ($\mu \geq 0$) in an ascent direction of the objective function (5) with respect to the motion parameters.

**Algorithm 1** Event-based Motion Segmentation

1: **Input**: events $\mathcal{E} = \{e_k\}_{k=1}^{N_e}$ in a space-time volume $V$ of the image plane, and number of clusters $N_\ell$.
2: **Output**: cluster parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_j\}_{j=1}^{N_\ell}$ and event-cluster assignments $\mathbf{P} \equiv p_{kj} \doteq P(e_k \in \ell_j)$.
3: **Procedure:**
4: Initialize the unknowns $(\boldsymbol{\theta}, \mathbf{P})$ (see Section 3.5).
5: **Iterate** until convergence:
6: • Compute the event-cluster assignments $p_{kj}$ using (7).
7: • Update the motion parameters of all clusters (6).

Motion-compensation methods [20, 25] typically use gradient ascent or line search to solve for the motion parameters that maximize some objective function of the IWE. In our case, because the IWE (2) depends on both $\boldsymbol{\theta}$ and $\mathbf{P}$ and we seek to jointly estimate them, we do not wastefully search for the best $\boldsymbol{\theta}$ given the current estimate of $\mathbf{P}$. Instead, we update $\boldsymbol{\theta}$ using (6), proceed to refine $\mathbf{P}$ (see (7)), and iterate.

Fixing the motion parameters $\boldsymbol{\theta}$, we may refine the associations $\mathbf{P}$ using a closed-form probability partitioning law:

$$p_{kj} = \frac{c_j(\mathbf{x}'_k(\boldsymbol{\theta}_j))}{\sum_{i=1}^{N_\ell} c_i(\mathbf{x}'_k(\boldsymbol{\theta}_i))}, \tag{7}$$

where $c_j(\mathbf{x}) \neq 0$ is the local contrast (i.e., sharpness) of the $j$-th cluster at pixel $\mathbf{x}$, and it is given by the value of the weighted IWE, $c_j(\mathbf{x}) \doteq I_j(\mathbf{x})$. Thus, each event is softly assigned to each cluster based on how it contributes to the sharpness of all $N_\ell$ IWEs. The alternating optimization approach in Algorithm 1 resembles the EM algorithm, with the E-step given by (7) and the M-step given by (6).

### 3.5. Initialization

The proposed alternating method converges locally (i.e., there is no guarantee of convergence to a global solution), and it requires initialization of $\boldsymbol{\theta}, \mathbf{P}$ to start the iteration.

Several initialization schemes are possible, depending on the motion models. For example, if the warps of all clusters are of optical flow type, one could first extract optical flow from the events (e.g., using [32, 33]) and then cluster the optical flow vectors (e.g., using the k-means algorithm). The resulting cluster centers in velocity space would provide an initialization for the motion parameters of the clusters $\boldsymbol{\theta}$.

We follow a greedy approach, similar to that in [24], that works well in practice, providing cluster parameters close to the desired ones. It is valid regardless of the motion models used. We initialize events to have equal association probabilities, and then maximize the contrast of the first cluster with respect to its motion parameters. We then find the gradient of the local contrast for each event with respect to the motion parameters. Those events that belong to the cluster under consideration become less "in focus" when we move away from the optimized parameters, so those events that

have a negative gradient are given a high association probability for that cluster and a low one for clusters subsequent. The process is repeated for the remaining clusters until all motion parameters $\boldsymbol{\theta}$ and associations $\mathbf{P}$ have been filled.

### 3.6. Discussion of the Segmentation Approach

The proposed approach is versatile, since it allows us to consider diverse parametric motion/warping models, such as linear motion (optic flow) [4, 20, 24], rotational motion [21], 4-DOF (degrees-of-freedom) motion [25], 8-DOF homographic motion [20], etc. Moreover, each cluster may have a different motion model, $\{\mathbf{W}_j\}_{j=1}^{N_\ell}$, as opposed to having a single model for all events, and therefore, all clusters. This characteristic is unique of our method.

It is also worth noting that the proposed method classifies events according to motion without having to explicitly compute optical flow, which is a widespread motion descriptor. Thus, our method is not simply optical flow clustering. Instead, our method encapsulates motion information in the warps of each cluster, thus by-passing the error-prone step of optical flow estimation in favor of achieving the desired goal of motion segmentation of the events.

The edge-like motion-compensated IWEs corresponding to each cluster are, upon convergence, a description of the intensity patterns (entangled with the motion) that caused the events. Thus our method recovers fine details of the appearance (e.g., shape) of the objects causing the events without having to estimate a (computationally expensive) 3D scene representation. In [25] fine details were only available for the dominant motion cluster.

Finally, the number of clusters $N_\ell$ is a hyper-parameter that may be tuned by a meta-algorithm (in the experiments, we set $N_\ell$ manually). This is a well-known topic in clustering [34]. While automatically determining the optimal $N_\ell$ depending on the scene is outside the scope of this paper, it should be noted that as we show in Section 4.3, our method is not sensitive to excess clusters $N_\ell$.

### 3.7. Sequence Processing

The above method segments the events $\mathcal{E}$ from a short time interval $T$. To process an entire stream of events, a sliding window approach is used, splitting the stream into packets of events $\{\mathcal{E}_n\}_{n=1}^{N_g}$. We process the $n$-th packet and then slide the window, thus selecting more recent events. The motions estimated by clustering $\mathcal{E}_n$ can be propagated in time to predict an initialization for the clusters of the next packet, $\mathcal{E}_{n+1}$. We use a fixed number of events $N_e$ per window, and slide by half of it, $N_e/2$.

## 4. Experiments

**Overview.** In this section we first provide a quantitative evaluation of our method on a publicly available, real-world dataset [25], showing that we significantly outperform two baseline methods [24, 25]. We provide further quantitative results on the accuracy of our method with regard to *relative motion differences* and we demonstrate the efficacy of our method on additional, challenging real-world data. Throughout the experiments, we demonstrate several features of our method, namely that (*i*) it allows arbitrary motion models for different clusters, (*ii*) it allows us to perform motion segmentation on difficult scenes (high speed and/or HDR), where conventional cameras would fail, (*iii*) it is robust to the number of clusters used ($N_\ell$), and (*iv*) that it is able to perform motion segmentation on non-rigid scenes. The sequences considered cover a broad range of motion speeds, from $12\,\text{pixel/s}$ to several hundred $\text{pixel/s}$.

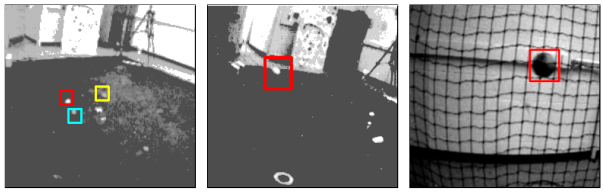
We strongly recommend looking at the accompanying video and supplementary material, where we present further experiments, including a comparison to "naive" k-means clustering, mixture density models and fuzzy-k-means.

The following experiments were carried out with data from a DAVIS240C camera [35], which provide both events and grayscale frames. The frames are not used in the experiments; they serve only an illustrative purpose.

### 4.1. Quantitative Evaluation

**Results on Dataset from [25].** We ran our segmentation method on the Extreme Event Dataset (EED) from [25] and compared against the results from [25] and [24]. The sequences in the EED dataset showcase a range of scenes (Fig. 3 and Table 1) which are very challenging for conventional cameras. In particular, they comprise fast moving objects (around $600\,\text{pixel/s}$) in `Fast Moving Drone` and `Multiple Objects`, which are almost indiscernible on the frames provided by the DAVIS camera, as well as scenes with extreme lighting variations, such as `Lightning variation` (in which a drone is tracked despite a strobe light pointing at the camera), and object occlusions. Having object segmentation rather per-event segmentation in mind, the EED dataset provides timestamped bounding boxes around the moving objects in the scene and proposes to measure *object segmentation success* whenever the estimated bounding box overlaps at least $50\,\%$ with the hand-labeled one *and* it has more area within the hand-labeled bounding box than outside. To compare against [25], we perform motion segmentation on the events that occur around the timestamp of the bounding-box and count success if for a given cluster the above criterion is true for the segmented events. For a fair comparison, we used the same type of motion models (4-DOF warps) as in [25].

Table 1 reports the results of the comparison of our method against [24] and [25]. Our method outperforms [24] in all sequences by a large margin (from $7.41\,\%$ to $84.52\,\%$), and improves over [25] in all but one sequence, where it has comparable performance. In four out of five sequences we achieve accuracy above $92\,\%$, and in one of them, a



(a) (b) (c)

Figure 3: Several scenes from the Extreme Event Dataset (EED) [25]: (a) `Multiple Objects`, (b) `Occluded Sequence` and (c) `What is Background?` Moving objects (drones, balls, etc.) are within hand-labeled bounding boxes. Images have been brightened for visualization.

| EED Sequence Name | SOFAS [24] | Mitrokhin [25] | Ours (Alg. 1) |
|---|---|---|---|
| `Fast moving drone` | 88.89 | 92.78 | **96.30** |
| `Multiple objects` | 46.15 | 87.32 | **96.77** |
| `Lighting variation` | 0.00 | **84.52** | 80.51 |
| `What is Background?` | 22.08 | 89.21 | **100.00** |
| `Occluded sequence` | 80.00 | 90.83 | **92.31** |

Table 1: Comparison with state-of-the-art using the success rate proposed by [25] of detection of moving objects (in %).

perfect score, 100 %. Some results of the segmentation are displayed on the first columns of Fig. 4. In the `What is Background?` scene (1st column of Fig. 4), a ball is thrown from right to left behind a net while the camera is panning, following the ball. Our method clearly segments the scene into two clusters: the ball and the net, correctly handling occlusions. In the `Lightning variation` (2nd column of Fig. 4), a quadrotor flies through a poorly lit room with strobe lightning in the background, and our method is able to segment the events due to the camera motion (green and blue) and due to the quadrotor (purple).

**Accuracy vs Displacement.** While the dataset from [25] provides a benchmark for comparison against the state-of-the-art, it does not allow us to assess the *per-event* accuracy of our method. Here we measure segmentation success directly as a percentage of correctly classified events, thus much more fine-grained than with bounding boxes. Since real data contains a significant proportion of noise events, we perform the quantitative analysis on event data from a state-of-the-art photorealistic simulator [36]. Knowing which objects generated which events, allows us to finely resolve the accuracy of our method.

However, segmentation accuracy is closely coupled with the observation window over which events are collected. Intuitively, this makes sense; observing two objects with a relative velocity of 1 pixel/s for only 1 s means that the objects have moved only 0.1 pixels relative to each other, a difference that is difficult to measure. Observing these two objects for 10 s means a relative displacement of 10 pixels, which is easier to distinguish.

Fig 5 evaluates the above effect on a sequence consisting of textured pebbles moving with different relative velocities (Fig. 5a, with events colored in red and blue, according to polarity). The plot on Fig 5b shows that as the relative displacement increases, the proportion of correctly classified events, and therefore, the segmentation accuracy, increases. Our method requires that roughly 4 pixels of relative displacement have occurred in order to achieve 90 % accuracy. This holds true for any relative velocity.

**Computational Performance.** The complexity of Algorithm 1 is linear in the number of clusters, events, pixels of the IWE and the number of optimization iterations, in total, $O((N_e + N_p)N_\ell N_{\text{it}})$. Our method generally converges in less than ten iterations of the algorithm, although this clearly depends on several parameters, such as the data processed. Further details are given in the supplementary material. Here, we provide a ballpark figure for the processing speed. We ran our method on a single core, 2.4 GHz CPU where we got a throughput of 240 000 events/s for optical-flow–type warps (Table 2). Almost 99 % of the time was spent in warping events, which is parallelizable. Using a GeForce 1080 GPU, we achieved a 10× speed-up factor, as reported in Table 2. The bottleneck is not in computation but rather in memory transfer to and from the GPU.

Throughput decreases as $N_\ell$ increases, since all of the events need to be warped for every extra cluster in order to generate motion compensated images. Further, extra clusters add to the dimensionality of the optimization problem that is solved during the motion-compensation step.

| $N_\ell$ | CPU [kevents/s] | GPU [kevents/s] |
|---|---|---|
| 2 | 239.86 | 3963.20 |
| 5 | 178.19 | 1434.66 |
| 10 | 80.93 | 645.02 |
| 20 | 32.43 | 331.50 |
| 50 | 12.62 | 113.78 |

Table 2: Throughput in kilo-events per second (optical-flow type [24]) of Algorithm 1 running on a single CPU core vs GPU for varying $N_\ell$ (using the test sequence in Fig. 7).

Regardless of throughput, our method allows exploiting key features of event cameras, such as its very high temporal resolution and its HDR, as shown in experiments on the EED dataset (Table 1) and on some sequences in Fig. 4 (Vehicle facing the sun, Fan and coin).

### 4.2. Further Real-World Sequences

We test our method on additional sequences in a variety of real-world scenes, as shown in Fig. 4. The third column shows the results of segmenting a traffic scene, with the camera placed parallel to a street and tilting upwards while vehicles pass by in front of it. The algorithm segmented the

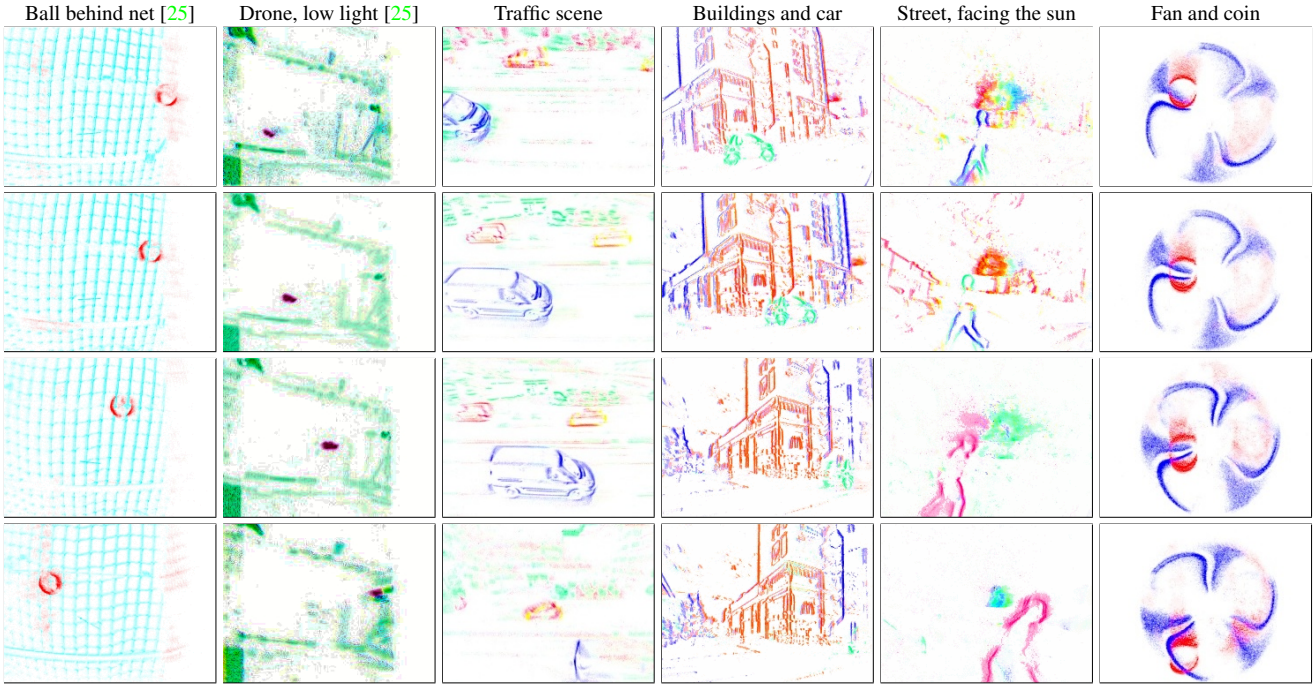
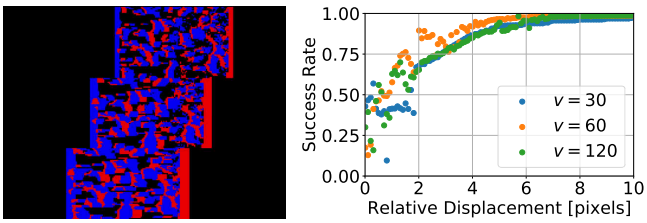



Figure 4: From top to bottom: snapshots (motion-compensated images, as in Fig.2) of events segmented into clusters on multiple sequences (one per column). Events colored by cluster membership. The images in the first row are GIFs that are animated if the document is opened with Adobe Acrobat Reader. Best viewed in the accompanying video.




(a) Events (red and blue). (b) Success Rate.

Figure 5: *Per-event Success Evaluation.* Segmentation accuracy vs relative object displacement (5b) on `pebbles` sequence (5a) at various relative velocities of objects $v_{\text{rel}} = \{30, 60, 120\}\text{pixel/s}$).

scene into four clusters: three corresponding to the vehicles (blue, red, yellow) and another one corresponding to the background buildings (ego-motion, in green). Even the cars going in the same direction are separated (red and yellow), since they travel at different speeds.

The fourth column of Fig. 4 shows the results of segmenting a scene with a car and some buildings while the camera is panning. We ran the algorithm to segment the scene into three clusters using optical flow warps. One cluster segments the car, and the other two clusters are assigned to the buildings. Note that the algorithm detects the differences in optical flow due to the perspective projection of the panning camera: it separates the higher speed in the periphery (blue) from the slower speed in the image center (red).

An HDR scene is shown on the fifth column of Fig. 4. The camera is mounted on a moving vehicle facing the sun (central in field of view) while a pedestrian and a skateboarder cross in front of it. The camera's forward motion causes fewer events from the background than in previous (panning) examples. We run the segmentation algorithm with six clusters, allowing the method to adapt to the scene. Segmented events are colored according to cluster membership. The algorithm correctly segments the pedestrian and the skateboarder, producing motion-compensated images of their silhouettes despite being non-rigidly moving objects.

Finally, the last column of Fig. 4 shows the versatility of our method to accommodate different motion models for each cluster. To this end, we recorded a coin dropping in front of the blades of a ventilator spinning at $1800\,\text{rpm}$. In this case the fan is represented by a rotational motion model and the coin by a linear velocity motion model. Our method converges to the expected, optimal solution, as can be seen in the motion compensated images, and it can handle the occlusions on the blades caused by the coin.

Fig. 6 shows that our method also works with a higher resolution ($640 \times 480$ pixels) event-based camera [37]. More experiments are provided in the Appendix.

### 4.3. Sensitivity to the Number of Clusters

The following experiment shows that our method is not sensitive to the number of clusters chosen $N_\ell$. We found that $N_\ell$ is not a particularly important parameter; if it cho-

(a) Scene (b) Segmentation into clusters

Figure 6: A 640×480 pixel DVS [37] panning over a slanted plane. Segmentation with 10 optical flow clusters (colored).

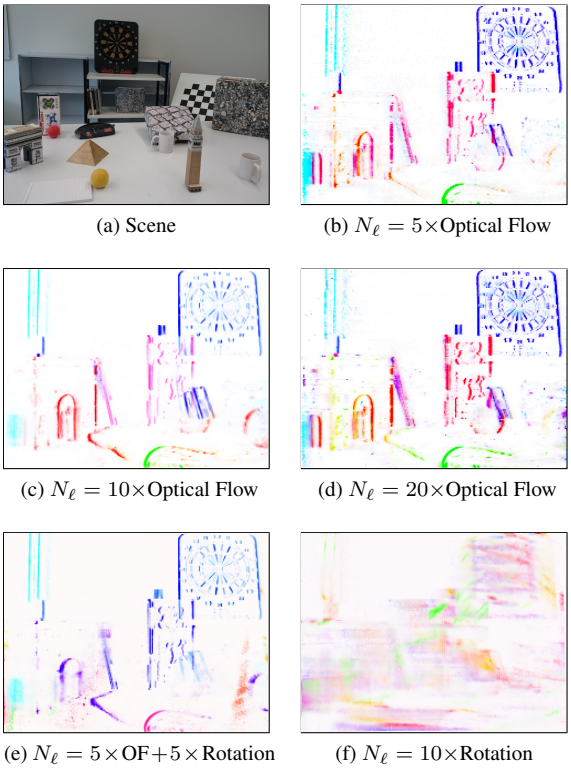
sen to be too large, the unnecessary clusters end up not having any events allocated to them and thus "die". This is a nice feature, since it means that in practice $N_\ell$ can simply be chosen to be large and then not be worried about. We demonstrate this on the `slider_depth` sequence from [38]; where there are multiple objects at different depths (depth continuum), with the camera sliding past them. Because of parallax, this results in a continuum of image plane velocities and thus infinitely many clusters would in theory be needed to segment the scene with an optical flow motion-model. Thus the sequence can only be resolved by adding many clusters which discretize the continuum of velocities.

Fig. 7 demonstrates that our method is robust with regard to the number of clusters chosen (in Figs. 7b–7d); too few clusters and the method will simply discretize the event cluster continuum, too many clusters and some clusters will "collapse", i.e., no events will be assigned to them. By segmenting with enough clusters and preventing cluster collapse, our method can be used to detect depth variations; nevertheless, tailored methods for depth estimation [39] are more suitable for such a task. The experiment also shows that our method deals with object occlusions.

Similarly, Fig. 7 shows that our method is not sensitive to the mixture of motion models either. Fig. 7e shows the result with five clusters of optical flow type and five clusters of rotation type. As can be seen, our method essentially allocates no event likelihoods to these rotation models clusters, which clearly do not suit any of the events in this sequence. Fig. 7f shows the result of using only rotation motion models, resulting in failure, as expected. As future work, a meta-algorithm could be used to select which motion models are most relevant depending on the scene.

## 5. Conclusion

In this work we presented the first method for per-event segmentation of a scene into multiple objects based on their apparent motion on the image plane. We jointly segmented a given set of events and recovered the motion parameters of the different objects (clusters) causing them. Additionally, as a by-product, our method produced motion-compensated images with the sharp edge-like appearance of the objects in the scene, which may be used for further analysis (e.g., recognition). We showed that our method outperforms two recent methods on a publicly available dataset (with as much as 10 % improvement over the state-of-the-art [25]), and showed it can resolve small relative motion differences between clusters. Our method achieves this using a versatile cluster model and avoiding explicit estimation of optical flow for motion segmentation, which is error prone. All this allowed us to perform motion segmentation on challenging scenes, such as high speed and/or HDR, unlocking the outstanding properties of event-based cameras.



(a) Scene (b) $N_\ell = 5\times$Optical Flow

(c) $N_\ell = 10\times$Optical Flow (d) $N_\ell = 20\times$Optical Flow

(e) $N_\ell = 5\times\text{OF}+5\times\text{Rotation}$ (f) $N_\ell = 10\times$Rotation

Figure 7: *Experiment on* `slider_depth` *sequence* of [38]. Motion-compensated images in 7b to 7f show events colored by cluster. Using optical flow (OF) warps (in 7b–7d), event clusters correspond to depth planes with respect to the camera. Using as few as five clusters, the events are discretized and approximately spread over the depth continuum. Using 5, 10 or 20 clusters in 7b, 7c, 7d gives very similar results, showing that our method is fairly insensitive to the value of $N_\ell$ chosen. Adding clusters with motion models that do not suit the motion, as in 7e, where five clusters are pure rotational warps, does not much disturb the output either.

**Acknowledgments.** This work was supported by the Swiss National Center of Competence in Research Robotics, through the Swiss National Science Foundation and the SNSF-ERC Starting Grant as well as the ARC Centre of Excellence for Robot Vision, project #CE140100016.

# Event-Based Motion Segmentation by Motion Compensation —Supplementary Material—

Timo Stoffregen[1,2], Guillermo Gallego[3], Tom Drummond[1,2], Lindsay Kleeman[1], Davide Scaramuzza[3]

[1]Dept. Electrical and Computer Systems Engineering, Monash University, Australia.
[2]Australian Centre of Excellence for Robotic Vision, Australia.
[3]Dept. Informatics (Univ. Zurich) and Dept. Neuroinformatics (Univ. Zurich & ETH Zurich), Switzerland.

## A. Multimedia Material

The video accompanying this work is available at https://youtu.be/0q6ap_OSBAk

## B. Two Additional Motion-Compensation Segmentation Methods

In this section, we describe how two classical clustering methods (mixture densities and fuzzy k-means) can be modified to tackle the task of event-based motion segmentation, by leveraging the idea of motion-compensation [20, 21] (Sections B.1 and B.2). Examples comparing the three per-event segmentation models developed (Algorithms 1 to 3) are given in Section B.3; they are called *proposed* (or *Layered*), *Mixture Densities* and *Fuzzy k-Means*, respectively.

### B.1. Mixture Densities

The mixture models framework [40, 41] can be adapted to solve the segmentation problem addressed. The idea is to fit a mixture density to the events $\mathcal{E}$, with each mode representing a cluster of events with a coherent motion.

**Problem Formulation.** Specifically, following the notation in [40, Ch.10], we identify the elements of the problem: the data points are the events $\mathcal{E}$ without taking into account polarity; thus, feature space is the volume $V$, and, consequently, the clusters are comprised of events (i.e., they are not clusters of optic flow vectors in velocity space).

The mixture model states that events $e_k \in V$ are distributed according to a sum of several distributions ("clusters"), with mixing weights ("cluster probabilities") $\boldsymbol{\pi} \doteq \{P(\omega_j)\}_{j=1}^{N_\ell}$:

$$p(e_k|\boldsymbol{\theta}) = \sum_{j=1}^{N_\ell} p(e_k|\omega_j, \boldsymbol{\theta}_j) P(\omega_j), \tag{8}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\theta}_j\}_{j=1}^{N_\ell}$ are the parameters of the distributions of each component of the mixture model and we assumed that the parameters of each cluster are independent of each other: $p(e_k|\omega_j, \boldsymbol{\theta}) = p(e_k|\omega_j, \boldsymbol{\theta}_j)$. The function $p(z|\boldsymbol{\theta})$ in (8), with $z \in V$, is a scalar field in $V$ representing the density of events in $V$ as a sum of several densities, each of them corresponding to a different cluster, and each cluster describing a coherent motion.

To measure how well the $j$-th cluster explains an event (8), we propose to use the unweighted IWE (10):

$$p(z \,|\, \omega_j, \boldsymbol{\theta}_j) \;\propto\; H_j\left(\mathbf{x}'(z; \boldsymbol{\theta}_j)\right) \tag{9}$$

$$H_j\left(\mathbf{x}\right) \;\doteq\; \sum_{m=1}^{N_e} \delta\left(\mathbf{x} - \mathbf{x}'_{mj}\right) \tag{10}$$

with warped event location $\mathbf{x}'_{mj} = \mathbf{W}(\mathbf{x}_m, t_m; \boldsymbol{\theta}_j)$. The image point $\mathbf{x}'(z; \boldsymbol{\theta}_j)$ corresponds to the warped location of point $z \in V$ using the motion parameters of the $j$-th cluster.

Hence, the goodness of fit between a point $z \in V$ and the $j$-th cluster is measured by the amount of event alignment (i.e., "sharpness"): the larger the IWE of the cluster at the warped point location, the larger the probability that $z$ belongs to the cluster.

Notice that the choice (9) makes the distribution of each component in the mixture $p(z \,|\, \omega_j, \boldsymbol{\theta}_j)$ be constant along the point trajectories defined by the warping model of the cluster, which agrees with the "tubular" shape mentioned in the problem statement (Section 3). The mixture model (8) may not be constant along point trajectories since it is a weighted sum of several distributions, each with its own point trajectories.

**Iterative Solver: EM Algorithm.** With the above definitions, we may apply the EM algorithm in [40, Ch.10] to compute the parameters of the mixture model, by maximizing the (log-)likelihood of the mixture density:

$$(\boldsymbol{\theta}^*, \boldsymbol{\pi}^*) = \arg\max_{(\boldsymbol{\theta}, \boldsymbol{\pi})} \sum_{k=1}^{N_e} \log\, p(e_k|\boldsymbol{\theta}) \tag{11}$$

**Algorithm 2** Event-based Motion Segmentation using Mixture Density Model

**Input**: events $\mathcal{E} = \{e_k\}_{k=1}^{N_e}$ in a space-time volume $V$ of the image plane, and number of clusters $N_\ell$.
**Output**: cluster parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_j\}_{j=1}^{N_\ell}$ and mixing weights $\boldsymbol{\pi} \doteq \{P(\omega_j)\}_{j=1}^{N_\ell}$.
**Procedure:**
Initialize $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$.
**Iterate** until convergence:
• Update the mixing weights (12), using the current motion parameters $\boldsymbol{\theta}$ and the mixing weights from the previous iteration in (13).
• Update motion parameters $\boldsymbol{\theta}$ by ascending on (11).

In the E-step, the mixing weights $\boldsymbol{\pi}$ are updated using

$$P(\omega_j) = \frac{1}{N_e} \sum_{k=1}^{N_e} p(\omega_j | e_k, \boldsymbol{\theta}_j) \tag{12}$$

with membership probabilities given by the Bayes formula

$$p(\omega_j | e_k, \boldsymbol{\theta}_j) = \frac{p(e_k | \omega_j, \boldsymbol{\theta}_j) P(\omega_j)}{\sum_{i=1}^{N_\ell} p(e_k | \omega_i, \boldsymbol{\theta}_i) P(\omega_i)}. \tag{13}$$

In the M-step, gradient ascent or conjugate gradient [42] of the log-likelihood (11) with respect to the warp parameters $\boldsymbol{\theta}$ is used to update $\boldsymbol{\theta}$, in preparation for the next iteration.

The pseudo-code of this mixture model method is given in Algorithm 2. From the mixing weights and the motion parameters, it is straightforward to compute the event-cluster assignment probabilities using (13). To initialize the iteration, we use the procedure described in Section 3.5.

Notice that, during the EM iterations, the above method not only estimates the cluster parameters $\boldsymbol{\theta}$ and the mixing weights $\boldsymbol{\pi}$ but also the distributions $p(z|\omega_i, \boldsymbol{\theta}_i)$ themselves, i.e., the "shape" of the components of the mixture model. These distributions get sharper (more peaky or "in focus") around the segmented objects as iterations proceed, and blurred around the non-segmented objects corresponding to that cluster. An example is given in Section B.3.

## B.2. Fuzzy k-Means

Event-based motion segmentation can also be achieved by designing an objective function similar to the one used in the fuzzy k-means algorithm [40, Ch.10].

**Problem Formulation.** This approach seeks to maximize

$$(\boldsymbol{\theta}^*, \mathbf{P}^*) = \arg\max_{\boldsymbol{\theta}, \mathbf{P}} \sum_{j=1}^{N_\ell} \sum_{k=1}^{N_e} p_{kj}^b d_{kj}, \tag{14}$$

where $b > 1$ (e.g., $b = 2$) adjusts the blending of the different clusters, and the goodness of fit between an event $e_k$ and a cluster $j$ in $V$ is given in terms of event alignment (i.e., "sharpness"):

$$d_{kj} \doteq \log H_j(\mathbf{x}'_{kj}), \tag{15}$$

the value of the unweighted IWE (10) at the warped event location using the motion parameters of the cluster. We use the logarithm of the IWE, as in (11), to decrease the influence of large values of the IWE, since these are counted multiple times if the events are warped to the same pixel location. Notice that (14) differs from (2)-(5): the responsibilities $p_{kj}$ appear multiplying the IWE (i.e., they are not included in a weighted IWE), and the sum is over the events (as opposed to over the pixels (4)).

Notice also that this proposal is different from clustering in optical flow space (Fig. 16). As mentioned in Section B.1, here the feature space is the space-time volume $V \in \mathbb{R}^3$ (i.e., event location), rather than the optical flow space ($\mathbb{R}^2$) (i.e., event velocity).

**Algorithm 3** Event-based Motion Segmentation using the Fuzzy k-Means Method

**Input**: events $\mathcal{E} = \{e_k\}_{k=1}^{N_e}$ in a space-time volume $V$ of the image plane, and number of clusters $N_\ell$.
**Output**: cluster parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_j\}_{j=1}^{N_\ell}$ and event-cluster assignments $\mathbf{P} \equiv p_{kj} \doteq P(e_k \in \ell_j)$.
**Procedure:**
Initialization (as in Section 3.5).
**Iterate** until convergence:
• Update the event-cluster assignments $p_{kj}$ using (16).
• Update motion parameters $\boldsymbol{\theta}$ by ascending on (14).

**Iterative Solver: EM Algorithm.** The EM algorithm may also be used to solve (14). In the E-step (fixed warp parameters $\boldsymbol{\theta}$) the responsibilities are updated using the closed-form partitioning formula

$$p_{kj} = d_{kj}^{\frac{1}{b-1}} \Big/ \sum_{i=1}^{N_\ell} d_{ki}^{\frac{1}{b-1}}. \tag{16}$$

In the M-step (fixed responsibilities) the warp parameters of the clusters $\boldsymbol{\theta}$ are updated using gradient ascent or conjugate gradient. The pseudo-code of the event-based fuzzy k-means segmentation method is given in Algorithm 3.

## B.3. Comparison of Three Motion-Compensation Segmentation Methods

We compare our method with the two above-mentioned methods (Sections B.1 and B.2) that we also designed to leverage motion compensation.

Fig. 8 shows the comparison of the three methods on a toy example with three objects (a filled pentagon, a star

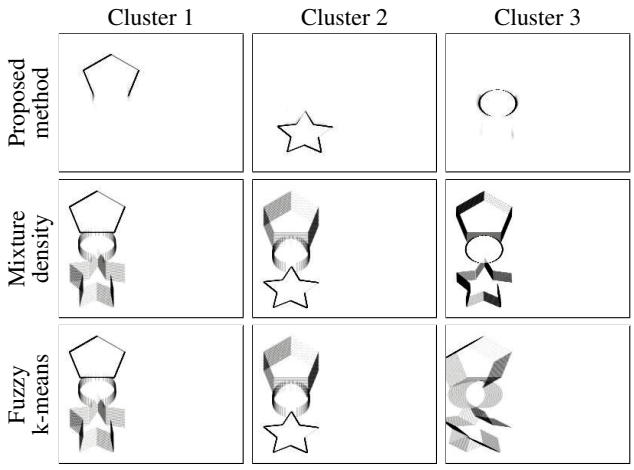



Figure 8: Comparison of three methods for event-based Motion Segmentation: Algorithms 1 to 3 (one per row).

and a circle) moving in different directions on the image plane. In the mixture density and fuzzy k-means methods, the motion-compensated IWEs do not include the event-cluster associations $\mathbf{P}$, and so, all objects appear in all IWEs, sharper in one IWE than in the others. In contrast, in the proposed method (Algorithm 1), the associations are included in the motion-compensated image of the cluster (weighted IWE), as per equation (2), and so, the objects are better split into the clusters (with minor "ghost" effects, as illustrated in Fig. 2), thus yielding the best results.

It is worth mentioning that the three per-event segmentation methods are novel: they have not been previously proposed in the literature. We decided to focus on Algorithm 1 in the main part of the paper and thus leave the adaptation of classical methods (mixture model and fuzzy k-means) for the supplementary material.

### B.4. Computational Complexity

Next, we analyze the complexity of the three segmentation methods considered (Algorithms 1 to 3), defined by objective functions (5), (11) and (14), respectively. The core of the segmentation methods is the computation of the images of warped events (IWEs (2) or (10); one per cluster), which has complexity $O(N_e N_\ell)$.

**Proposed (Layered) Model.** The complexity of updating the event assignments using (7) is essentially that of computing the (weighted) IWEs of all clusters, i.e., $O(N_e N_\ell)$. The complexity of computing the contrast (4) of a generic image is linear in the number of pixels, $O(N_p)$, and so, the complexity of computing the contrast of one IWE is $O(N_e + N_p)$. The computation of the contrast is negligible compared to the effort required by the warp. Computing the contrast of $N_\ell$ clusters (corresponding to a set of candidate parameters) has complexity $O\big((N_e + N_p)N_\ell\big)$. Since multiple iterations $N_{\text{it}}$ may be required to find the optimal parameters, the total complexity of the iterative algorithm used is $O\big((N_e + N_p)N_\ell N_{\text{it}}\big)$.

**Mixture Density Model.** The complexity of updating the mixture weights is that of computing the posterior probabilities $p(\omega_j|e_k, \boldsymbol{\theta}_j)$, which require to compute the IWEs of all clusters, i.e., complexity $O(N_e N_\ell)$. The complexity of updating the motion parameters is also that of computing the contrasts of the IWEs of all clusters, through multiple ascent iterations. In total, the complexity is $O\big((N_e + N_p)N_\ell N_{\text{it}}\big)$.

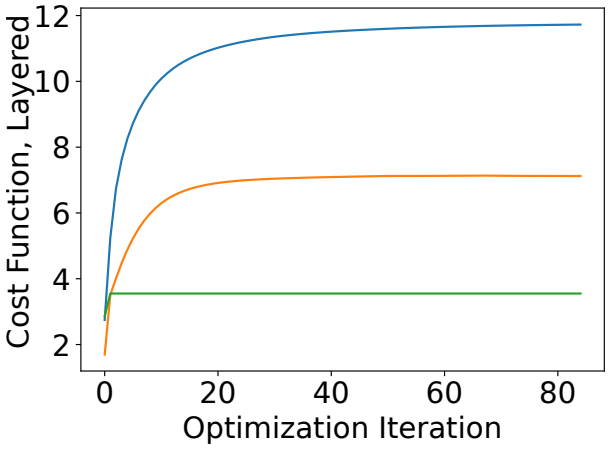
**Fuzzy k-means Model.** The complexity of computing the responsibilities (16) is that of computing $N_\ell$ IWEs (values $d_{kj}$), i.e., $O(N_e N_\ell)$. The complexity of updating the motion parameters is that of computing the objective function (14), $O(N_e N_\ell)$, through multiple iterations. In total, the complexity is $O(N_e N_\ell N_{\text{it}})$.

**Plots of Computational Effort and Convergence.** Fig. 9 shows the convergence of the three above methods on real data from a traffic sequence that is segmented into four clusters (Fig. 10 and third column of Fig. 4): three cars and the background due to ego-motion. The top plot, Fig. 9a, shows the evolution of the sum-of-contrasts objective function (5) vs the iterations. All methods stagnate after $\approx 20$ iterations, and, as expected, the proposed method provides the highest score among all three methods (since it is designed to maximize this objective function). The Mixture model and Fuzzy k-means methods do not provide such a large score mostly due to the event-cluster associations, since they are not as confident to belonging to one cluster as in the proposed method. Fig. 9b displays the number of warps (i.e., number of IWEs) that each method computes as the optimization iterations proceed; as it can be shown, the relationship is approximately linear, with the proposed method performing the least warps for a considerable number of iterations, before stagnation (Fig. 9a).

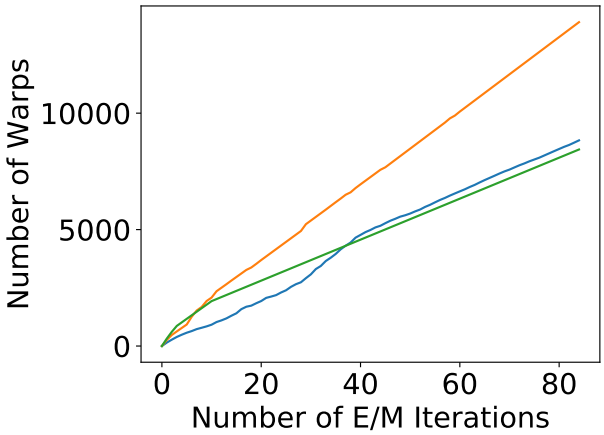
## C. Additional Experiments

### C.1. Non-rigid Moving Objects

In the following experiments we show how our method deals with non-rigid objects. Our algorithm warps events $\mathcal{E}$ according to point-trajectories described by parametric motion models whose parameters are assumed constant over the (small) time $\Delta t$ spanned by $\mathcal{E}$. Low-dimensional parametric warp models, such as the patch-based optic flow (2-DOF, linear trajectories), rotational motion in the plane (1-DOF) or in space (3-DOF) are simple and produce robust results by constraining the dimensionality of the solution space in the search for optimal point-trajectories. However,




(a) Value of the sum-of-contrasts objective function (5) for each of the three methods per iteration of optimization.




(b) Number of warps performed by each method per iteration of optimization.

Figure 9: *Comparison of three Methods.* We compare the convergence properties of three motion-compensated event-segmentation methods (Algorithms 1 to 3): proposed (layered) method (*blue*), Mixture Density Model (*orange*) and Fuzzy k-Means (*green*). Data used is from the Traffic Sequence (third column of Fig. 4), the warped events at each iteration are visualized in Fig. 10.

simple warp models (both in event-based vision or in traditional frame-to-frame vision) have limited expressiveness: they are good for representing rigidly moving objects, but do not have enough degrees of freedom to represent more complex motions, such as deformations (e.g., pedestrian, birds, jelly fish, etc.). One could consider using warps able to describe more complex motions, such as part-based warp models [43] or infinite-dimensional models [44]. But this would make the segmentation problem considerably harder, not only due to the increased dimensionality of the search space, but also because it would be possibly filled with multiple local minima.

**Pedestrian.** Fig. 11 shows a pedestrian walking past the camera while it is panning. In spite of using simple warp models, our method does a good job at segmentation: the background (due to camera motion), the torso of the person and the swinging arms are segmented in separate clusters. This is so, because during the short time span of $\mathcal{E}$ (in the order of milliseconds), the objects move approximately rigidly.

**Popping Balloon.** In order to test the limits of this assumption, we filmed the popping of a balloon with the event camera (see Fig. 12). While the segmentation struggles to give a clear result in the initial moments of puncturing (12b), it still manages to give reasonable results for the fast moving, contracting fragments of rubber flung away by the explosion (12c, 12d).

### C.2. Additional to Section 4.3 - Continuum Depth Variation

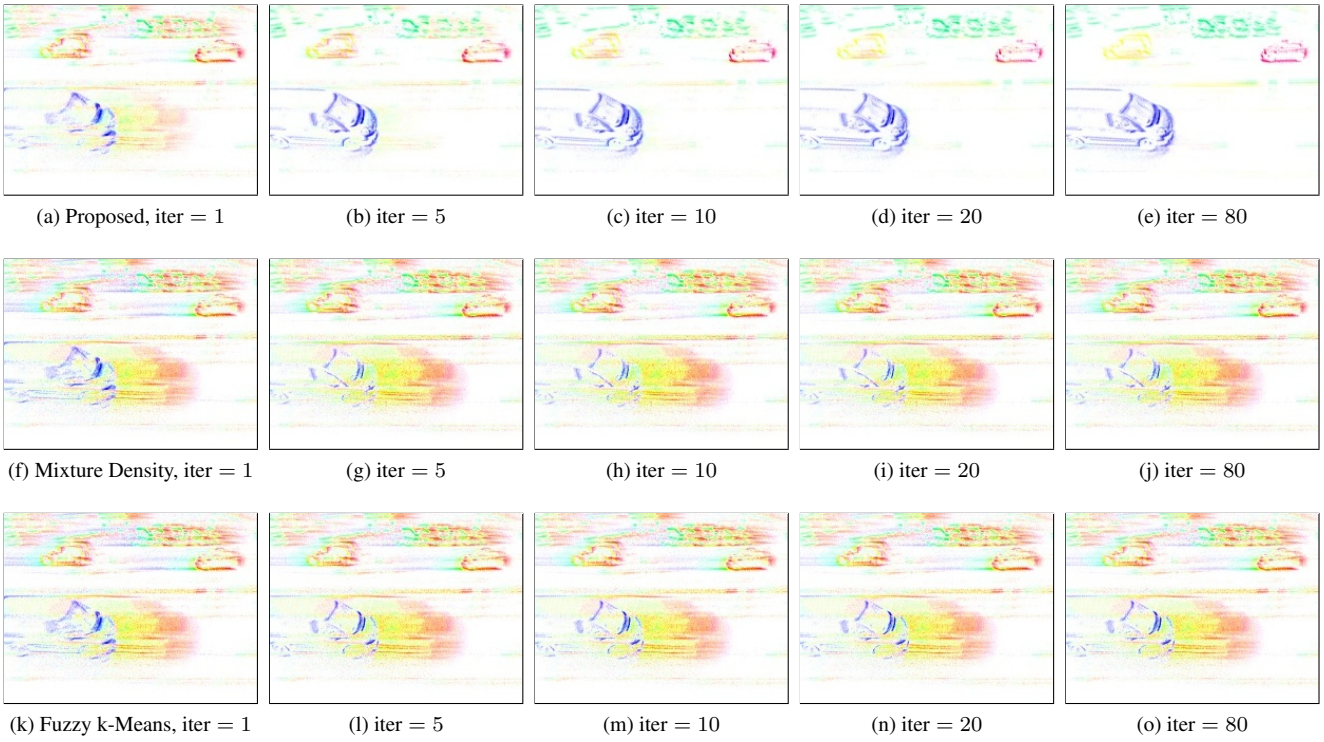
In this experiment we essentially show a scene similar to that in Fig. 7. The difference is that the scene in Fig. 13 shows a truly continuous depth variation. As can be seen in the results (using $N_\ell = 15$), our method discretizes the segmentation, although it is noteworthy that each "slice" of depth appears to fade toward foreground and background. This is because the method becomes less certain of the likelihood of events that sit between clusters, the darkness of a region reflecting the likelihood of a given event belonging to that cluster.

### C.3. Continuum Depth Variation with High-Resolution Event-based Camera

Due to the recent development of new, high resolution event-based cameras [37], we show the results of our method on the output of a Samsung DVS Gen3 sensor, with a spatial resolution of $640 \times 480$ pixels. In this experiment we show the segmentation of several scenes (a textured carpet, some leaves and a temple poster) as the camera moves. Due to ego-motion induced parallax, there is a continuous gradient in the motion in the scene, i.e., the scenes present a continuum of depths. As can be seen in Fig. 14, our method works the same on high-resolution data.

### C.4. Comparison to k-means Optic Flow Clustering

Finally, the following experiments shows the comparison of our method against k-means clustering of optic flow. We first illustrate the difference with a qualitative example and then quantitatively show the ability of our method to resolve small differences in velocities compared to k-means. To this end, we use an event-based camera mounted on a motorized linear slider, which provides accurate ground truth position of the camera. Since the camera moves at constant speed in a 1-D trajectory, the differences in optical flow values



(a) Proposed, iter = 1 (b) iter = 5 (c) iter = 10 (d) iter = 20 (e) iter = 80

(f) Mixture Density, iter = 1 (g) iter = 5 (h) iter = 10 (i) iter = 20 (j) iter = 80

(k) Fuzzy k-Means, iter = 1 (l) iter = 5 (m) iter = 10 (n) iter = 20 (o) iter = 80

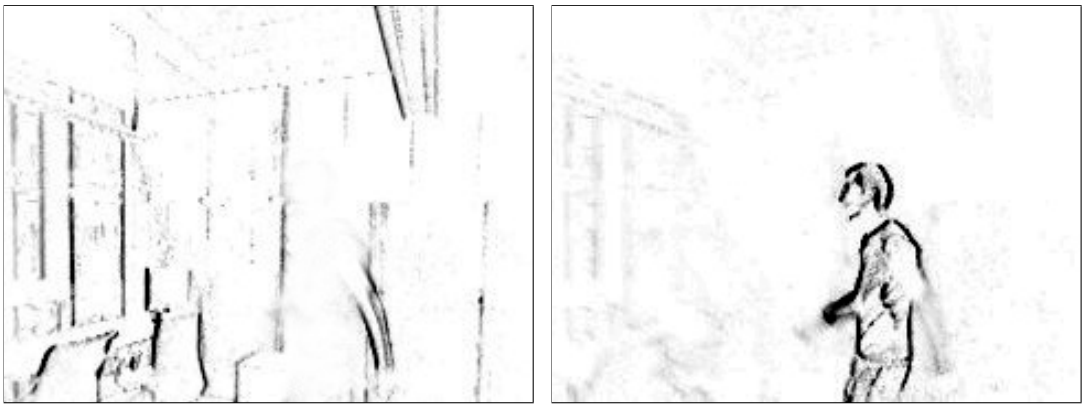
Figure 10: Images of the motion-corrected events for three segmentation methods. From left to right the images show the state after 1, 5, 10, 20 and 80 iterations respectively. *Top Row*: Algorithm 1, *Middle Row*: Algorithm 2, *Bottom Row*: Algorithm 3.

observed when viewing a static scene are due to parallax from the different depths of the objects causing the events.

**Numbers Sequence.** In this experiment, we placed six printed numbers at different, known depths with respect to the linear slider. The event-based camera moved back and forth on the slider at approximately constant speed. Due to parallax, the objects at different depths appear to be moving at different speeds; faster the closer the object is to the camera. Thus we expect the scene to be segmented into six clusters, each corresponding to a different apparent velocity.

Fig. 15 compares the results of k-means clustering optic flow and Algorithm 1. To compute optical flow we use conventional methods on reconstructed images at a high frame rate [45], with the optical flow method in [46] producing better results on such event-reconstructed images than state-of-the-art learning methods [47]. The results show that the velocities corresponding to the six numbers are too similar to be resolved correctly by the two-step approach (flow plus clustering), as evidenced by the bad segmentation of the scene (numbers 3, 4 and 5 are clustered together, whereas three clusters are used to represent the events of the fastest moving number–the zero, closest to the camera). In contrast, our method accurately clusters the events according to the motion of the objects causing them, in this case, according to velocities, since we used an optical flow warp (linear motion on the image plane). The higher accuracy of our method is easily seen in the sharpness of the motion-compensated images (cf. Fig. 15d and Fig. 15f).

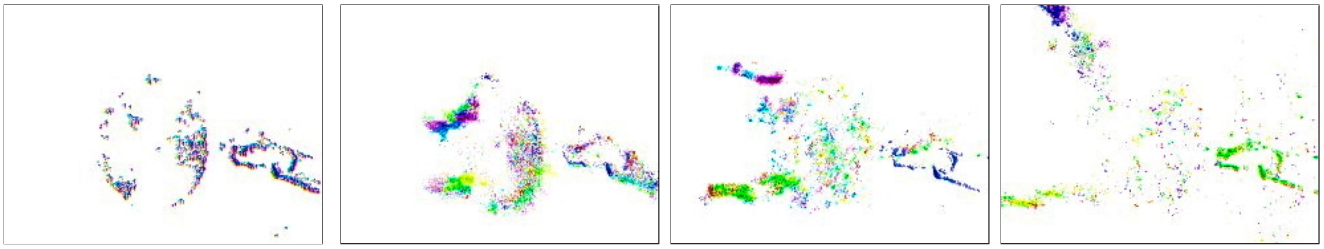
**Rocks at Different Speeds.** We also tested our algorithm on two real sequences with six objects of textured images of pebbles (qualitatively similar to Fig. 5), in which the relative velocities of the objects were $50\,\text{pixels/s}$ and $6\,\text{pixels/s}$, respectively. Fig. 16 shows the results. If the objects are moving with sufficiently distinct velocities (Fig. 16a), the clusters can be resolved by the two-step approach. However, once the objects move with similar velocities (Fig. 16a), k-means clustering of optical flow is unable to correctly resolve the different clusters. In contrast, our method works well in both cases; it is much more accurate: it can resolve differences of $6\,\text{pixel/s}$ for objects moving at $50\,\text{pixel/s}$ to $80\,\text{pixel/s}$, given the same slice of events.



(a) Cluster 1 (b) Cluster 2

(c) Cluster 3 (d) All clusters (merged IWEs)

Figure 11: *Non-Rigid Scene.* A person walks across a room, arms swinging. The room 11a, the body 11b and the arms 11c are segmented out, with greater uncertainty to the event associations in areas of deformation (such as elbows), visible in the fact that events are associated to both clusters (events colored by cluster in 11d).



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

Figure 12: *Non-Rigid Moving Objects.* From left to right: snapshots of segmentation of balloon popping. Run with $N_\ell = 4$ clusters, events colored by cluster membership.

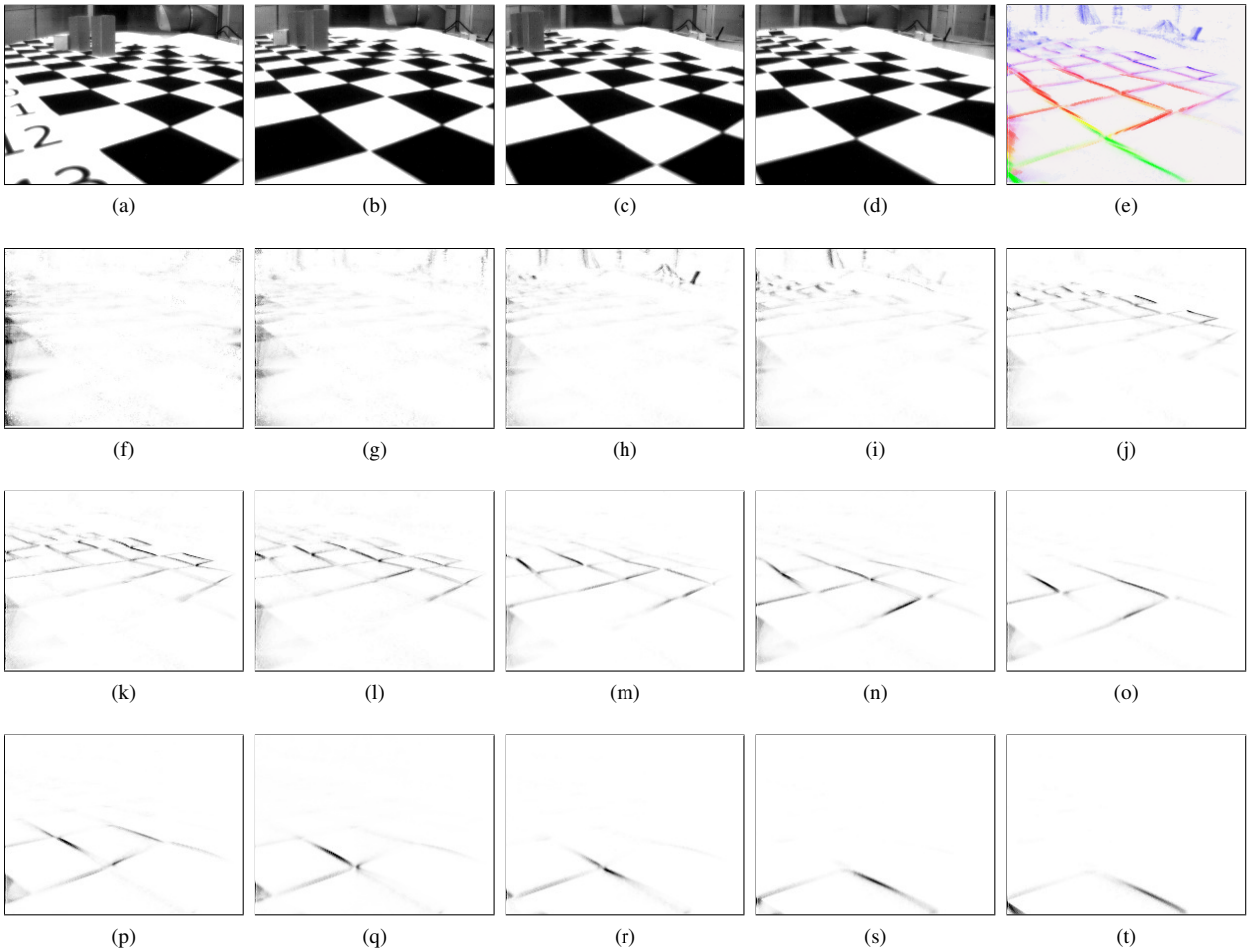



Figure 13: Sequence from a camera translating past a checkerboard (13a-13d). These grayscale frames, provided by the DAVIS [35] are not used by our method; they are just for visualization purposes. Each image in 13f-13t shows the IWE of each cluster (15 clusters, optical flow motion models). 13e shows the segmented output (combined IWEs) in the accustomed colored format.

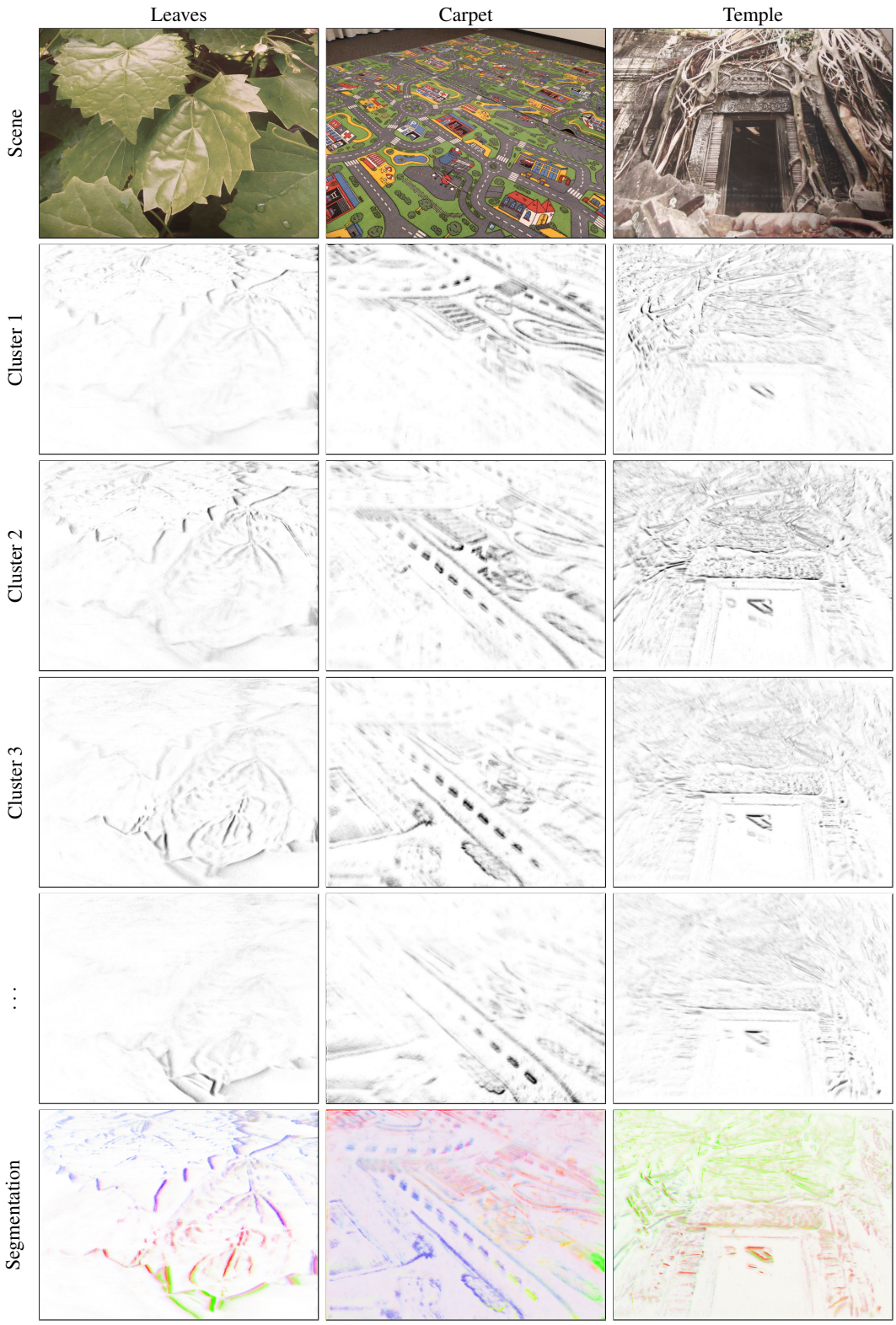
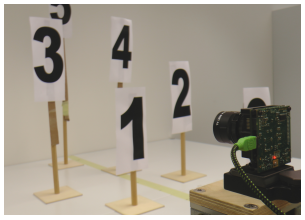
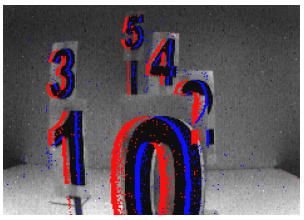


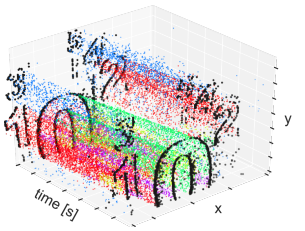

Figure 14: Scenes recorded with a Samsung DVS Gen3 event camera ($640 \times 480$ pixels); algorithm run with ten clusters ($N_\ell = 10$). From top to bottom: scene recorded, four of the clusters (motion-compensated IWEs, with darkness indicating event likelihoods), and the accustomed colored segmentation (as in Fig. 2). These examples illustrate that our method can be used to segment the scene according to depth from the camera, although it is not its main purpose.



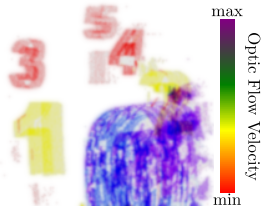
(a) DAVIS performs linear translation over a multi-object scene.



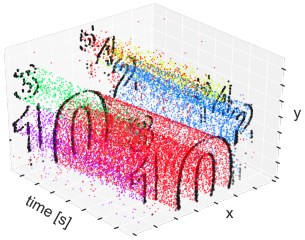
(b) Resulting image and events (red and blue, indicating polarity).



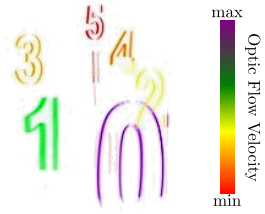

(c) Clustered volume of events (colored by cluster number).




(d) Motion-compensated image (colored by clustered optic flow).

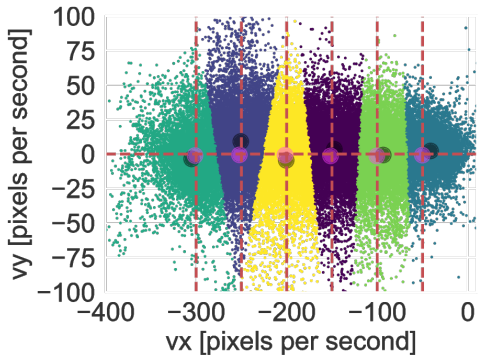



(e) Clustered volume of events (colored by cluster number).

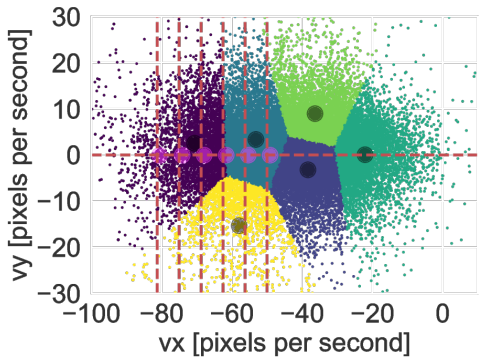



(f) Motion-compensated image (colored by recovered optic flow).

Figure 15: *Numbers Sequence*. Motion Segmentation by k-means clustering on estimated optic flow (center row) and by Algorithm 1 (bottom row).




(a) Minimum velocity between objects: 50 pixel/s




(b) Minimum velocity between objects: 6 pixel/s

Figure 16: *Rocks at Different Speeds*. Segmentation by k-means clustering of estimated optical flow ($k = 6$). The plots show the distribution of optical flow vectors and the six Voronoi diagrams resulting from k-means clustering on optic flow space. The crossings of red dashed lines indicate the ground truth optical flow velocity, the dark circles indicate the centroids of the k-means clusters. The pink circles indicate the cluster's optical flow estimated by our method (Algorithm 1).

## References


[1] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck, "A 128×128 120 dB 15 $\mu$s latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008. 1, 3

[2] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza, "Event-based vision: A survey," *arXiv:1904.08405*, 2019. 1, 2

[3] Elias Mueggler, Basil Huber, and Davide Scaramuzza, "Event-based, 6-DOF pose tracking for high-speed maneuvers," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pp. 2761–2768, 2014. 1

[4] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis, "Event-based feature tracking with probabilistic data association," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 4465–4470, 2017. 1, 5

[5] Guillermo Gallego, Jon E. A. Lund, Elias Mueggler, Henri Rebecq, Tobi Delbruck, and Davide Scaramuzza, "Event-based, 6-DOF camera tracking from photometric depth maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, pp. 2402–2412, Oct. 2018. 1

[6] Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza, "EKLT: Asynchronous photometric feature tracking using events and frames," *Int. J. Comput. Vis.*, 2019. 1

[7] Jörg Conradt, Matthew Cook, Raphael Berner, Patrick Lichtsteiner, Rodney J. Douglas, and Tobi Delbruck, "A pencil balancing robot using a pair of AER dynamic vision sensors," in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 781–784, 2009. 1

[8] Tobi Delbruck and Manuel Lang, "Robotic goalie with 3ms reaction time at 4% CPU load using event-based dynamic vision sensor," *Front. Neurosci.*, vol. 7, p. 223, 2013. 1

[9] Erich Mueller, Andrea Censi, and Emilio Frazzoli, "Low-latency heading feedback control with neuromorphic vision sensors using efficient approximated incremental inference," in *IEEE Conf. Decision Control (CDC)*, 2015. 1

[10] Hanme Kim, Stefan Leutenegger, and Andrew J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," in *Eur. Conf. Comput. Vis. (ECCV)*, pp. 349–364, 2016. 1

[11] Henri Rebecq, Timo Horstschäfer, Guillermo Gallego, and Davide Scaramuzza, "EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real-time," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 593–600, 2017. 1

[12] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis, "Event-based visual inertial odometry," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 5816–5824, 2017. 1

[13] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza, "Ultimate SLAM? combining events, images, and IMU for robust visual SLAM in HDR and high speed scenarios," *IEEE Robot. Autom. Lett.*, vol. 3, pp. 994–1001, Apr. 2018. 1

[14] Luca Zappella, Xavier Lladó, and Joaquim Salvi, "Motion segmentation: A review," in *Conf. Artificial Intell. Research and Development*, pp. 398–407, 2008. 2

[15] Björn Ommer, Theodor Mader, and Joachim M. Buhmann, "Seeing the objects behind the dots: Recognition in videos from a moving camera," *Int. J. Comput. Vis.*, vol. 83, pp. 57–71, Feb. 2009. 2

[16] Martin Litzenberger, Christoph Posch, D. Bauer, Ahmed Nabil Belbachir, P. Schön, B. Kohn, and H. Garn, "Embedded vision system for real-time object tracking using an asynchronous transient vision sensor," in *Digital Signal Processing Workshop*, pp. 173–178, 2006. 2

[17] Zhenjiang Ni, Sio-Hoï Ieng, Christoph Posch, Stéphane Régnier, and Ryad Benosman, "Visual tracking using neuromorphic asynchronous event-based cameras," *Neural Computation*, vol. 27, no. 4, pp. 925–953, 2015. 2

[18] Ewa Piatkowska, Ahmed Nabil Belbachir, Stephan Schraml, and Margrit Gelautz, "Spatiotemporal multiple persons tracking using dynamic vision sensor," in *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, pp. 35–40, 2012. 2

[19] John YA Wang and Edward H Adelson, "Layered representation for motion analysis," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 361–366, 1993. 2, 3

[20] 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 *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 3867–3876, 2018. 2, 3, 4, 5, 9

[21] Guillermo Gallego and Davide Scaramuzza, "Accurate angular velocity estimation with an event camera," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 632–639, 2017. 2, 5, 9

[22] Arren Glover and Chiara Bartolozzi, "Event-driven ball detection and gaze fixation in clutter," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pp. 2203–2208, 2016. 2

[23] Valentina Vasco, Arren Glover, Elias Mueggler, Davide Scaramuzza, Lorenzo Natale, and Chiara Bartolozzi, "Independent motion detection with event-driven cameras," in *IEEE Int. Conf. Adv. Robot. (ICAR)*, 2017. 2

[24] Timo Stoffregen and Lindsay Kleeman, "Simultaneous optical flow and segmentation (SOFAS) using Dynamic Vision Sensor," in *Australasian Conf. Robot. Autom. (ACRA)*, 2017. 2, 4, 5, 6

[25] Anton Mitrokhin, Cornelia Fermuller, Chethan Parameshwara, and Yiannis Aloimonos, "Event-based moving object detection and tracking," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018. 2, 3, 4, 5, 6, 7, 8

[26] Francisco Barranco, Ching L. Teo, Cornelia Fermuller, and Yiannis Aloimonos, "Contour detection and characterization for asynchronous event sensors," in *Int. Conf. Comput. Vis. (ICCV)*, 2015. 2

[27] Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza, "Focus is all you need: Loss functions for event-based vision," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 12280–12289, 2019. 2

[28] Timo Stoffregen and Lindsay Kleeman, "Event cameras, contrast maximization and reward functions: An analysis," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 12300–12308, 2019. 2

[29] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis, "Unsupervised event-based learning of optical flow, depth, and egomotion," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2019. 2

[30] Laurent Dardelet, Sio-Hoi Ieng, and Ryad Benosman, "Event-based features selection and tracking from intertwined estimation of velocity and generative contours," *arXiv:1811.07839*, 2018. 2

[31] Rafael C. Gonzalez and Richard Eugene Woods, *Digital Image Processing*. Pearson Education, 2009. 4

[32] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi, "Event-based visual flow," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 407–417, 2014. 4

[33] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis, "EV-FlowNet: Self-supervised optical flow estimation for event-based cameras," in *Robotics: Science and Systems (RSS)*, 2018. 4

[34] C. Fraley, "How many clusters? Which clustering method? Answers via model-based cluster analysis," *The Computer Journal*, vol. 41, pp. 578–588, Aug. 1998. 5

[35] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck, "A 240x180 130dB 3us latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014. 5, 15

[36] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza, "ESIM: an open event camera simulator," in *Conf. on Robotics Learning (CoRL)*, 2018. 6

[37] Bongki Son, Yunjae Suh, Sungho Kim, Heejae Jung, Jun-Seok Kim, Changwoo Shin, Keunju Park, Kyoobin Lee, Jinman Park, Jooyeon Woo, Yohan Roh, Hyunku Lee, Yibing Wang, Ilia Ovsiannikov, and Hyunsurk Ryu, "A 640x480 dynamic vision sensor with a 9um pixel and 300Meps address-event representation," in *IEEE Intl. Solid-State Circuits Conf. (ISSCC)*, 2017. 7, 8, 12

[38] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Robot. Research*, vol. 36, no. 2, pp. 142–149, 2017. 8

[39] Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza, "EMVS: Event-based multi-view stereo—3D reconstruction with an event camera in real-time," *Int. J. Comput. Vis.*, vol. 126, pp. 1394–1414, Dec. 2018. 8

[40] Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Wiley, 2000. 9, 10

[41] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006. 9

[42] Jorge Nocedal and S. Wright, *Numerical Optimization*. Springer-Verlag New York, 2006. 10

[43] M. Pawan Kumar, P. H. S. Torr, and Andrew Zisserman, "Learning layered motion segmentations of video," *Int. J. Comput. Vis.*, vol. 76, pp. 301–319, Mar. 2008. 12

[44] Anthony J. Yezzi and Stefano Soatto, "Deformotion: Deforming motion, shape average and the joint registration and approximation of structures in images," *Int. J. Comput. Vis.*, vol. 53, pp. 153–167, July 2003. 12

[45] Cedric Scheerlinck, Nick Barnes, and Robert Mahony, "Continuous-time intensity estimation using event cameras," in *Asian Conf. Comput. Vis. (ACCV)*, 2018. 13

[46] Gunnar Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian Conf. on Im. Analysis (SCIA)*, pp. 363–370, 2003. 13

[47] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 13