Graph cuts for asynchronous event-based vision sensors

Nicolai Waniek Johannes von Stetten
Jörg Conradt
Neuroscientific System Theory, Technische Universität München
Arcisstraße 21, 80333 München
nicolai.waniek@tum.de

Abstract

Many computer vision results are based on the max-flow min-cut theorem. It is used to denoise images, solve stereo-vision problems, or perform image segmentation, for instance. Consequently, its application to dense data is well understood. In contrast, novel event-based vision sensors provide a sparse and asynchronous data representation that is fundamentally different to image frames. So far it was unclear how to define a suitable graph and its edge weights for this data in the context of graph cuts. Here we present a method to construct this graph for the task of denoising event-based data of a natural scene. The graph and its edge weights are computed without the necessity for manual prior selection. We evaluate the approach using a manually labeled dataset to demonstrate proof-of-principle.

1. Introduction and motivation

Computer vision applications typically employ standard cameras to acquire input data. These sensors are well understood and usually generate dense frames that can be post-processed with suitable hardware, i.e. graphics cards, at a certain frequency. Furthermore, modern cameras provide resolutions of up to several megapixels.

Event-based sensors, or silicon retinas, work quite differently and mimic the response of receptors on biological retinas [26, 22, 24, 27]. They are built of independently and asynchronously operating pixels which detect changes in luminance and emit *events* as soon as the perceived change exceeds or falls below a configurable threshold. Therefore only significant changes in the observed scene are detected and transmitted, leading to an almost continuous stream of sparse data.

These novel sensors display several potential benefits and advantageous characteristics. They achieve high dynamic ranges of up to $120\,\mathrm{dB}$ at high speeds of about $15\,\mu\mathrm{s}$ inter-event intervals per pixel while exhibiting only very low power consumption. Hence they are highly suitable for small and embedded systems which are limited in space or the number and amount of available resources, such as miniature flying robots.

The sparse data and high speed pose a twofold challenge. On the one hand they yield tremendous event rates already for sensors with only low resolutions. For instance, one specific obtainable event-based sensor is the Dynamic Vision Sensor (DVS) [22]. It has a resolution of only 128×128 pixels but can generate up to several 100k events per second. The amount of data therefore requires algorithms with high throughput or that can ideally be realized in dedicated parallel hardware [10].

On the other hand it is difficult to apply algorithms that can be considered *classical* or *fundamental* in the computer vision literature directly to event streams. The sparse data may infringe algorithmic constraints or make their application simply impractical. For instance, tracking algorithms like SIFT or SURF rely on information extracted from the surrounding of a detected corner to locate the same corner in subsequent frames [25, 1]. It is non-trivial to define such a descriptor for event streams due to the lack of dense local information. Consequently it appears necessary to first understand how to perform basic computations on event streams, such as noise reduction, line tracking, or corner detection, that many people would argue are solved problems for frame based data.

In fact, this is what happened recently. For instance, progress has been reported to estimate time-to-contact [11], detect and track corners [12], to compute optical flow [3, 30, 9], or to estimate stereo-information and disparity [20, 15] from event-based data. Remarkably, some of these algorithms received strong inspiration from or exposed a striking similarity to biological neural networks in the visual cortex. Other applications of event-based sensors include anomaly detection [33] or simultaneous localization and mapping on small robotic platforms [35, 34, 32].

One of the aforementioned *fundamental* algorithms in computer vision is graph cuts. It is for instance applied to denoise images, label regions, or even stereo-vision. We

asked if it is possible to apply graph cuts to event data in the seemingly simple task of denoising an event stream. Data produced by silicon retinas unfortunately contains a non-neglectable amount of noise. Many algorithms would benefit if they had to process only events that were caused by real objects, though. In fact, some algorithms use hand-crafted heuristics to address the issue. Here we derive a method that reliably extracts almost exclusively events generated by real stimuli.

First a short introduction to graph cuts is given for readers unacquainted with the method. Then we will outline the challenges posed by event data and suggest one way to solve them. We evaluate our technique using a manually labeled dataset to demonstrate proof-of-principle. Finally we will discuss our findings and conclude with remarks about future work.

2. A short but gentle primer on graph cuts

To make this paper self-contained we will quickly introduce graph cuts and then proceed to necessary definitions. To promote uniformity, we tried to keep notations as close as possible to the referenced literature. Readers familiar with the matter may directly jump to the next section.

Many early vision tasks require labeling of individual pixels according to a set of labels. For instance, image segmentation assigns labels to pixels according to which segment they belong. Introducing a cost to assign the labels reduces the process to an energy minimization problem. The issue can be rephrased in form of graph theroetical notation, which thus allows to apply the max-flow min-cut theorem[28]. Despite their comparatively long history – a method to compute the maximum flow through a network is already known for many decades [16] – interest in efficient methods to compute graph cuts continues. Only recently, advances to distribute the computation were reported [36] and tighter theoretical bounds discovered [18].

After introduction to the computer vision community in [17], graph cuts have since seen widespread adoption. Their application to image data has undergone thorough and rigorous analysis. For in-depth analyses and overviews we would like to refer the reader to the work presented in [7, 5, 8, 4, 21] and especially [6]. The latter includes a comprehensive study of graph cuts and their application to computer vision tasks. Especially interactive labeling and segmentation [31] or extensions to N-dimensional spaces [23] are of current concern.

The energy minimization problem can be formalized with the help of the following notations.

Definition 1. *A dataset $\mathcal{P}$ consists of N enumerable elements $p \in \mathcal{P}$ for which a neighborhood $\mathcal{N}$ is defined. More formally, $\mathcal{N} = \{\{p, q\} : p, q \in \mathcal{P}, p \neq q\}$.*

For example $\mathcal{P}$ represents an input image, $p \in \mathcal{P}$ is an individual pixel, and the set $\mathcal{N}$ covers all unique pairs of pixels. It is hence possible to express neighborhood relations with the help of additional functions, e.g. the Euclidean distance between two pixels in the image coordinate space. Note that a relationship function d is not necessarily symmetric, i.e. $d(p, q) = d(q, p)$ may not hold.

Definition 2. *A labeling L of $\mathcal{P}$ assigns each $p \in \mathcal{P}$ one element of the set of labels $\mathcal{L}$, i.e. it is a vector*

$$L = (l_0, \ldots, l_p, \ldots, l_{N-1})$$

where $l_p \in \mathcal{L}$ is the label of p.

Obviously there are $|\mathcal{L}|^N$ different possible labelings. The goal is to find one labeling which is optimal to certain criteria.

According to [5], it is possible to find an optimal labeling $\hat{L}$ by defining and minimizing the basic energy function

$$E(L) = \lambda \cdot R(L) + B(L) \tag{1}$$

where λ is a tune-able, application dependent parameter. $R(L) = \sum_{p \in \mathcal{P}} R(l_p)$ is an accumulator for the cost to assign individual labels using a cost function $R(l_p)$ and $B(L) = \sum_{\{p,q\} \in \mathcal{N}} B(l_p, l_q)$ determines the cost to assign labels to neighboring elements. Usually, $B(l_p, l_q) = 0$ for $l_p = l_q$ which serves to penalize discontinuities across pixels and is called boundary term. Nevertheless, $B(l_p, l_q)$ needs to preserve discontinuities in general as it would equalize the whole labeling otherwise [21].

The definitions above, especially the neighborhood relations, lend themselves to description in graph theoretical terms. Let the elements $p \in \mathcal{P}$ be vertices in a graph and their neighborhood relationships edges. After introducing two additional nodes, called source (s) and sink (t) to which all other vertices are connected, the minimization can be expressed using the following definition.

Definition 3. *Let a graph $G = (V, E)$ consists of vertices $v \in V$, edges $a \in E$, and non-negative edge-weights $w_a \in \mathbb{R}^+$. An s-t cut $C \subset E$ is a set of edges which separates G into two disjunct subsets S and T. The cost $c(C) = \sum_{a \in C} w_a$ is the sum of all weights of edges on the cut.*

According to the max-flow min-cut theorem, it is possible to minimize Equation 1 by computing the minimum cut $\hat{C}$. Examples of the construction of a graph and cuts are shown in Figure 1.

It is straightforward to define cost functions and edge weights for many applications. For instance, the gray scale value of images can be used as regional term and the inter-pixel distance as boundary term. Unfortunately, this is not the case for event-based data. We will address the reason and present a possible technique in the following sections.

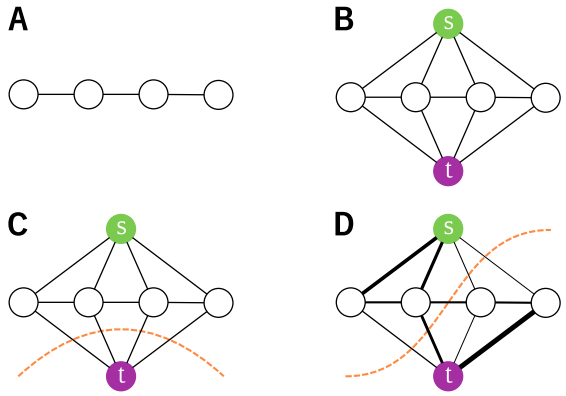

Figure 1: Construction of a graph from data and cut examples. **A** One dimensional input in which consecutive vertices are considered neighbors. **B** Original vertices connected to source (s) and sink (t) terminals. **C** A trivial cut (dashed line) completely separating one terminal node. **D** An optimal cut in a graph with different edge weights, indicated by line thickness.

3. A short story on naive graph cuts on event data

The sensors under consideration yield streams of events that are induced by changes of luminance. In natural scenes these changes are mostly due to motion. Formally, an event e is a tuple $e = (t, p, \mathbf{x})$ with time-stamp t, polarity $p \in \{-1, 1\}$, and location $\mathbf{x} \in \mathbb{N}_0^{X \times Y}$ where X and Y correspond to the sensor resolution. The polarity informs about decrease or increase in luminance, respectively. Visualizations of event data are usually confined to 2D representations, where events of a certain time window are accumulated and rendered as frames such as in Figure 2A. However, this representation disguises the fact that events are actually elements of a very sparse voxel-space, such as depicted in Figure 2B. Nevertheless some algorithms for event data internally operate on some form of accumulation frames, e.g. [29, 13].

It may be tempting to apply graph cuts to accumulated events of a certain time window and treating the accumulated events as image. Here, gray-scale values are typically computed as temporal difference between the last occurrence of an event on the respective pixel and the time the image is produced, where bright pixels indicate a small difference. The issue is thus reduced to compute a graph cut on a gray scale image. Given the typically small resolution of event sensors this can be computed quickly. However, there are significant drawbacks in this approach.

Either the accumulated frame is computed only a few times, e.g. after a fixed delta-t, or the cut needs to be evaluated for each event. The first method partially neglects the temporal information and high temporal resolution of event-based sensors, one of their main advantages compared to other sensors. The latter technique leads to significant performance issues, as both the accumulation frame needs to be updated and the graph cut computed for each event. This seems to be ill-advised, given event rates of up to several 100k events per second in many scenes.

The major issue with this naive approach is the flawed interpretation of temporal information, though. Assume two independent stimuli that are moving at two different speeds, one very fast and one very slow. Furthermore, consider that accumulation frames are read out at a steady frequency. As mentioned before, the gray scale values of their pixels are determined as temporal differences. It can be expected that the fast moving stimulus generated events that are recent enough with respect to the read-out time of the image to produce a high gray scale value. However, this is not necessarily the case for the slowly moving stimulus. The gray-scale value therefore depends on the relative time between read-out and event. A quick solution, namely completely ignoring the temporal difference and simply setting all pixels on which an event occurred to a high gray scale value won't solve the issue either. Due to the small resolution of many event-based sensors, some stimuli cover only very few pixels. Small regions however are prone to get cut off as background noise in graph cuts.

Clearly a solution is required that operates on events only, treats all events equally but separately, and uses a meaningful interpretation of the temporal information that is independent of read-out times.

4. Temporal trace graph cuts for event-based data

We seek to apply graph cuts to denoise a given stream $s = (e_0, \ldots, e_T)$ of events $e_t = (t, p, \mathbf{x})$ at times $0 \leq t \leq T$. Furthermore, manual prior selection which is often necessary in classical computer vision applications should be avoided. For this, we will need to extract additional information from the stream itself. Note that we constrain additional information for an event e_t to origin only from events e_k prior to e_t, i.e. $k < t$. Although the graph and the cut are computed for the whole stream s in the work presented here, this constraint will help to develop on-line versions of the algorithm in future work. To further simplify the technique, only events of one polarity are considered if not stated otherwise, namely on-events.

4.1. The trivial homogeneous labeling pitfall

We will now argue why additional information is required and present its computation in the subsequent section. Although events carry polarity information, this by itself not necessarily relates directly to real brightness of input stimuli. For instance, a pen moving in front of a checkerboard pattern may produce both on- and off events in a counter-intuitive manner at every edge transition. Hence without any

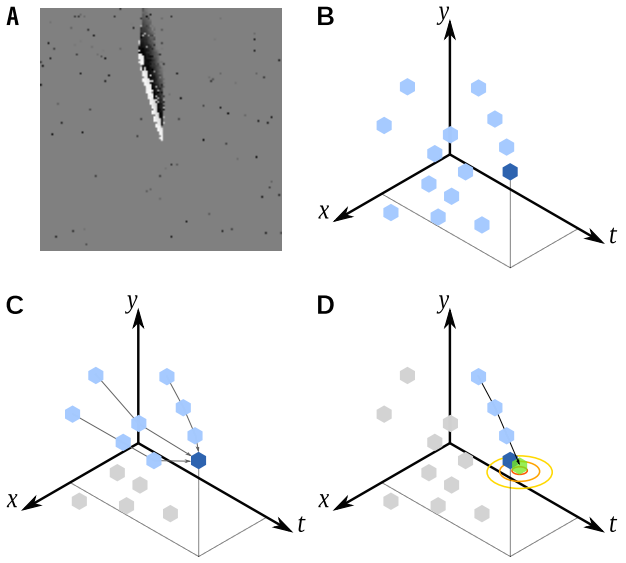

Figure 2: Visualization of event-based data and trace computation. **A** On- and off- events (white and black, respectively) accumulated for few milliseconds of a pen falling in front of a dynamic vision sensor. **B** A single event (dark blue) is a voxel in a sparse $x - y - t$ volume of events (light blue) that happened before. **C** Feasible traces of previous events (light blue voxels linked by gray lines) that end at the novel event. Unsuitable events (gray voxels) are ignored. **D** The most likely trace (light blue voxels) predicts the time at which the novel event is expected (green voxel, circles indicate certainty).

additional knowledge, events should be considered primarily to carry only unary information, namely that some change occurred at their locations. Unfortunately, this leads to the following vexing situation.

Without loss of generality, consider a one dimensional sensor which consists of solely one singular pixel that emits events when luminance changes are detected. Given some events, a graph can be constructed in the following way that conforms to Definition 3 which comprises regional and boundary terms as required in Equation 1. The boundary term consists of neighborhood relations between individual events, for instance subsequent events can be considered immediate neighbors. An example is given in Figure 1A. In addition, s and t nodes are introduced into the graph and connected to each event to form the regional term. The final graph is shown Figure 1B.

If a graph cut is applied to this graph, the result will always be a homogeneous labeling in which all events will be labeled either object or noise. Due to lack of sufficient information, all edges have to be treated equally weighted, hence the graph cut reduces to minimizing the number of edges that need to be cut. A homogeneous cut example is shown in Figure 1C. It is trivial to proof that this always yields either of the two possible homogeneous labelings. The proof can easily be extended to two dimensional sensors.

Hence suitable edge weights are required that are significantly different from each other to allow non-trivial cuts. One likely candidate is temporal information between events. However, using time-stamps themselves introduces issues as soon as multiple stimuli with different velocities are perceived by a sensor, similar to the issue in the naive approach. Consequently we will now present prerequisites for and the derivation of a method that yields such a measure.

4.2. Temporal traces and the linearity conjecture

Consider an ideal sensor of resolution $X \times Y$ with arbitrary small pixels that discretize the XY space and process which yields unique time-stamps t. In natural scenes, stimulus typically covers multiple pixels at the same time. For now however, focus on a smallest possible moving stimulus S triggering only *singular events* without any aliasing effects. S can therefore be understood as a particle moving in front of the sensor, generating events $e_q, \ldots, e_s, e_r$ as soon as it transits across pixels. By implication we can thus assume that there exists a predecessor event e_s with $s \leq t - 1 < t$ in the immediate pixel-neighborhood of any e_t which is due to the same stimulus. The same argument holds recursively for e_s. Hence, e_t is the most recent entry in an ordered sequence of events, called *trace*. Clearly, each *trace* can be decomposed into chunks of events which we call *trace segments*.

Definition 4. *Given an arbitrarily small stimulus S, the* trace $\mathcal{T}_t$ *of an event e_t is the ordered tuple of events triggered by S since stimulus onset time q. More formally, $\mathcal{T}_t = (e_q, \ldots, e_s, e_t)$ for $q < \cdots < s < t$.*

Definition 5. *Each trace $\mathcal{T}_t$ can be decomposed into an arbitrary number of unique* segments τ *such that $\mathcal{T}_t = (e_q, \ldots, e_s, e_t) = (\tau_0, \ldots, \tau_N)$ holds.*

Regarding their time-stamps, individual segments are related by $\min \tau_r \leq \max \tau_{r-1}$ and $\max \tau_{r-1} < \max \tau_r$. Consider the example $\mathcal{T}_t = (e_0, e_1, e_2, e_3)$. A suitable decomposition of $\mathcal{T}_t$ is (τ_0, τ_1) with $\tau_0 = (e_0, e_1)$ and $\tau_1 = (e_1, e_2, e_3)$.

Two consecutive events e_s and e_t of a trace are denoted $e_s \rightarrow e_t$, where "$\rightarrow$" reads as *happens before*. With the assumption of an ideal sensor, we can thus formally identify the origin of a predecessor event.

Axiom 1. *Given two events $e_s \rightarrow e_t$ on a two-dimensional ideal sensor. The origin of e_s is one of the coordinates in the immediate 8-neighborhood of e_t.*

The neighborhood is visualized in Figure 3A. Note that $s = t - 1$ may not hold: as soon as multiple stimuli and

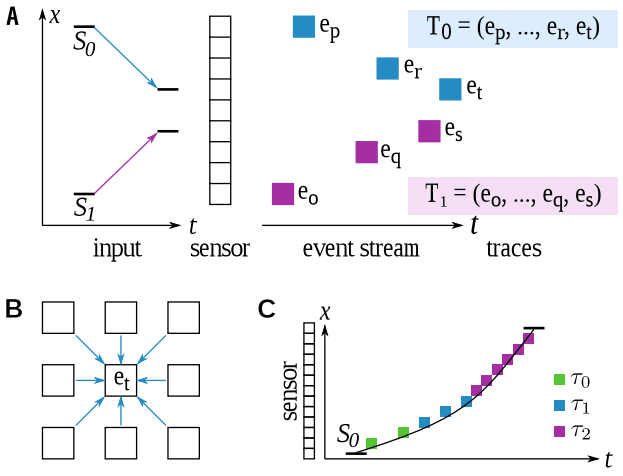

Figure 3: Traces, neighborhood, and linearity conjecture. **A** Two bar stimuli S_0, S_1 moving in front of a one dimensional sensor in opposite directions, generating the traces $\mathcal{T}_0, \mathcal{T}_1$. Each event gets a unique time-stamp, potentially leading to interleaving traces. **B** For a two dimensional sensor, the location of a preceding event e_s of an event e_t is within the 8-neighborhood. **C** The high temporal sampling rate of event-based sensors decomposes a trace $\mathcal{T}$ of stimulus S_0 into piecewise linear segments τ_0, τ_1, τ_2.

therefore traces exist, event times of different traces could interleave (Figure 3B). This is due to hardware constraints, for more details see [22].

Given a stimulus S, its acceleration is coded in the temporal difference between consecutive events of its trace $\mathcal{T}_t$. Recall that pixels on event-based sensors operate asynchronously. Furthermore it can be expected that the extremely high temporal sampling rate of individual pixels far exceeds the rate of change of motion in most natural scenes. A short segment $\tau_r = (\dots, e_{r-1}, e_r, e_{r+1}, \dots)$ of $\mathcal{T}_t$ with $|\tau_r| \ll |\mathcal{T}_t|$ therefore captures an approximately linear part of the motion of S. We can thus state the following postulate which is visualized in Figure 3C.

Conjecture 1. *Consider an ideal sensor and a moving stimulus S. The acceleration of S can be recovered from the corresponding trace $\mathcal{T}_t$ by composition of the piecewise linear segments $\tau_r \subset \mathcal{T}_t$.*

Conclusively there exists a τ_t for any event e_t which is linear. The temporal differences between all consecutive events of τ_t are constant. In other words, τ_t has a fixed mean with zero variance.

4.3. Using maximum likelihood traces to compute edge weights

Let us now return to the problem of finding suitable edge weights w_a for a graph to yield non-homogeneous cuts in the task of data denoising. Recall that, according to Equation 1 and Definition 3, we need to define weights for the regional term as well as the boundary term for elements of a dataset $\mathcal{P}$. Here, $\mathcal{P}$ stands for the entirety of events recorded with an event-based sensor. Hence it is necessary to determine the likelihood of an event to either belong to an object or to be noise.

Consider an event $e_t \in \mathcal{P}$ that is not triggered at the stimulus on-set time q, i.e. $q < t$. Examples of e_t and $\mathcal{P}$ are shown in Figure 2B. Definition 4 tells that there exists exactly one trace $\mathcal{T}_t$ which leads to e_t. Due to noisy measurements, real sensor data of a natural scene is likely to contain many potential candidate traces $\mathcal{T}_t^{(j)}$, for $j \geq 1$, though. The first step is thus to reduce $\mathcal{P}$ to the set of potential traces which, according to Axiom 1, live in the (recursive) 8-neighborhood. An example of some traces is depicted in Figure 2C.

Conjecture 1 allows to further narrow down the search space. Clearly, the number of possible traces $\mathcal{T}_t^{(j)}$ grows exponentially with the length of the longest trace. The linearity conjecture reduces this to query only the trace segments $\tau_t^{(j)}$. In fact, the results reported below suggest that a segment length of 5 is already sufficient. Still, this is a problem for real-time scenarios that we will address in Section 5.

The most likely segment can be computed due to the linearity conjecture. Let $\mu(\tau_t)$ and $\sigma(\tau_t)$ be functions that return the mean and variance, respectively, of temporal differences between consecutive events $e_r \to e_s$ of a trace segment. The most likely segment $\hat{\tau}_t$ is given by

$$\hat{\tau}_t = \arg\min_{\tau_t^{(j)}} \sigma(\tau_t^{(j)}) \tag{2}$$

where $\tau_t^{(j)}$ are trace segments with $|\tau_t^{(j)} \geq n > 1$.

It turns out that $\mu_t := \mu(\hat{\tau}_t)$ and $\sigma_t := \sigma(\hat{\tau}_t)$ lead to a characterization of e_t. Let μ_{t-1} and σ_{t-1} be the mean and variance, respectively, of the most likely segment $\hat{\tau}_{t-1}$ without e_t, i.e. $\hat{\tau}_t = (\hat{\tau}_{t-1}, e_t)$. The Kullback–Leibler divergence $D_{KL}(\cdot \parallel \cdot)$, for instance, is one possible measure to finally reason about an event using

$$c(e_t) = D_{KL}(\mathcal{N}(\mu_{t-1}, \sigma_{t-1}) \parallel \mathcal{N}(\mu_t, \sigma_t)) \tag{3}$$

where $\mathcal{N}(\mu, \sigma)$ denotes a Gaussian distribution with mean μ and variance σ. The resulting value indicates how strongly an event changes the most likely trace segment with respect to the linearization conjecture. Another interpretation is that the best matching trace segment $\hat{\tau}_t$ is used to predict the occurrence of the next event and quantify how good the real event matches the prediction. This interpretation is visualized in Figure 2D.

Note that other measures than the Kullback-Leibler divergence are possible. Therefore we termed $c(e_t)$ the *characteristic value* to distinguish it from the function used in its computation.

Finally we can construct remaining edges and compute their weights. Recall that two different types of weights are required. First consider weights for the regional term in which an event e is connected to the source and sink nodes s and t, respectively (see Figure 1 for clarification). With a suitable characteristic value $c(e)$ such as given by Equation 3 this can be defined as

$$w_e^s = c(e) \text{ and } w_e^t = 1 - c(e)\,. \tag{4}$$

The principle is that according to Equation 3 a small characteristic value indicates an event that matches the prediction. Hence, it is more likely to origin from a stimulus than it should be considered noise.

Secondly, and lastly, focus on the boundary term. Most natural scenes will not trigger singular events but whole waves of events, for instance due to an edge. The linearity conjecture leads to the assumption that events of an event wave that are triggered by the same stimulus should have the same or similar characteristic values. Conclusively we connect each event e to all (existing) events f in its 8-neighborhood that happened before. The edge weights are computed according to

$$w_e^f = \frac{1}{\sqrt{|dx|^2 + |dy|^2 + |\kappa \cdot dc|^2}}\,, \tag{5}$$

where dx and dy are spatial offsets in x and y direction. dc is the difference in characteristic value. We introduced the tunable parameter $\kappa \geq 1$ to adapt to sensor characteristics and to avoid numerical issues when dc becomes too small.

One of the design goals of the algorithm was that the construction of the graph can be performed iteratively. Therefore, each characteristic value is only allowed to depend on events that happened before. This constraint will allow to extend the algorithm to an on-line variant in future work.

5. Experimental results and discussion

In the following section we present experimental results of our approach. We implemented the gist of our algorithm in C++ and used Lemon Library 1.3.1 to compute the graph cut [14]. We discarded trace segments with less than 3 elements and already stopped at a maximum segment length of 5. Albeit not shown here, our results suggest that longer segments not necessarily improve the results.

The dataset is a short event stream of a pen falling in front a white background. A two dimensional accumulation frame of the pen falling in front of the sensor is shown in Figure 2A. The sensor was fixed on a static mounting to avoid events due to shaking and only natural light was used.

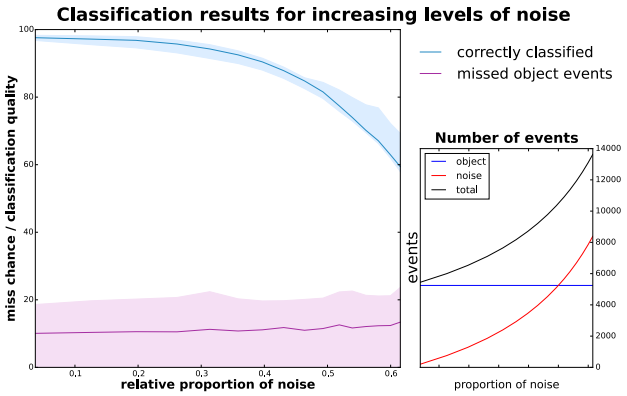

Figure 4: Combined classification results for several values of κ.

The dataset was subsequently manually labeled. Events that are clearly far away from the majority of other events were labeled as noise. All other events that visually formed a coherent structure were marked as object. Note that hereby some events were marked object which could be considered to be artifacts of trailing noise. There are no known clear indicators for trailing noise, though. We will discuss this form of noise further below. Finally, artificial random noise was systematically added to the original dataset.

Recall that we are interested in extracting events that are almost certainly due to a real stimulus. Most algorithms that we are aware of can handle fewer numbers of events easily. However they show issues with distractor events. Thus it is preferable to discard some events that were in fact generated by an object than to include too many events that are due to noise.

Results for different levels of noise and settings of κ are depicted in Figure 4. The classification quality was computed as follows. Each event reported to be an object was checked against the ground truth data. The relative number of correctly identified events with respect to the total number of events reported to be an object yields the classification quality. The number of events missed during the process is displayed as well. The algorithm yields a high classification quality despite the increase in noise. Quite interestingly, the number of missed events stays approximately constant although the number of noise increases steadily.

Our technique successfully selects almost only genuine object events in the original dataset. Three dimensional example representations of the original and noisy inputs and outputs are shown in Figure 5A and B, respectively. The figure displays all events as blue circles in a three dimensional space. Already the original dataset contains perceivable noise. However, most of the noise is correctly identified. However, some events that belong to the object were marked as noise as well.

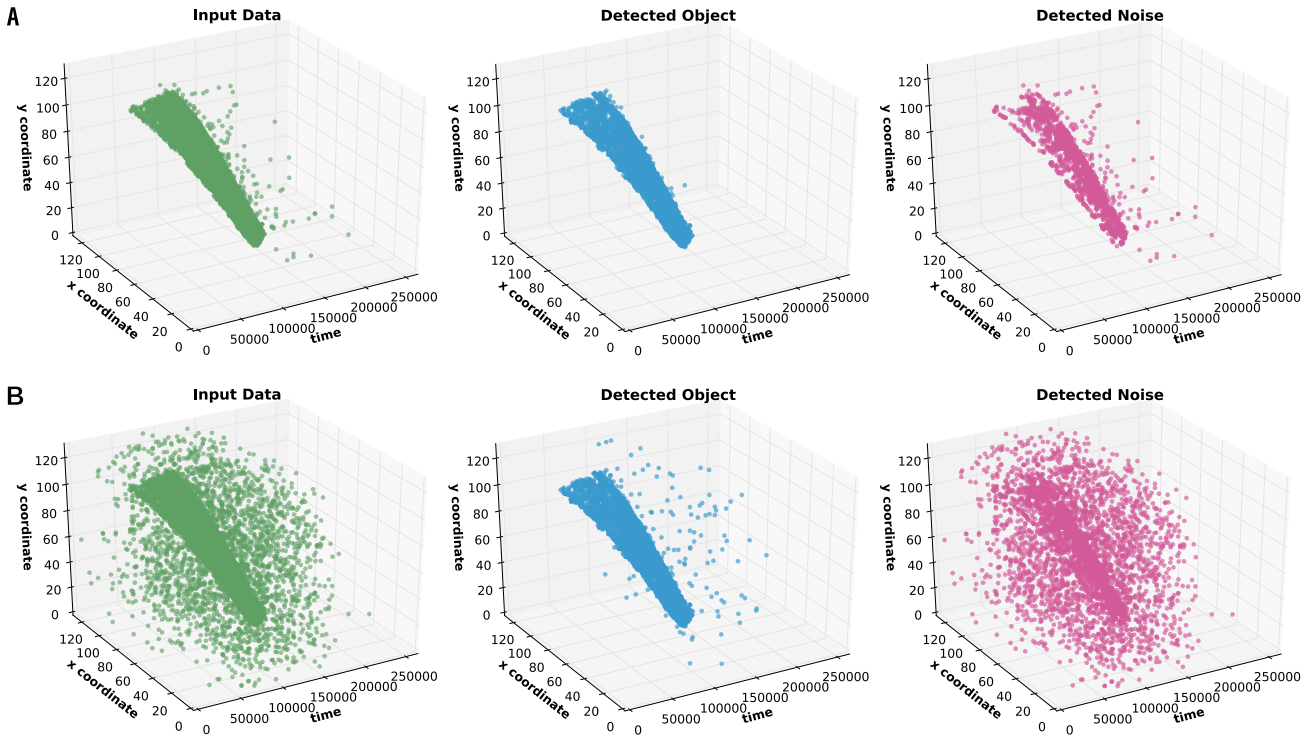

Figure 5: Result examples. **A** Application of our technique to the original dataset. **B** The input data is augmented with artificial noise, here a proportional amount of 20%.

The algorithm cuts off a layer on the outer shell of events. This behavior is depicted in the third column of Figure 5A. Close visual inspection of the data shows that the events that belong to the object according to ground truth but are marked as noise seem to fall into two categories. The first class is visible in the figure as a seemingly thin line of events right before the remaining depicted events. These are events that mark the onset of the stimulus, as they are the temporally first to appear in this area. Either the trace segment was not long enough at this location, or the temporal information was not yet coherent. The latter may be due to the characteristics of the used sensor and the generated events. Quite often a sensor pixel generates two events that follow each other quickly instead of a single event that would be expected.

The second class of events detected to be noise but belonging to the object may be due to trailing noise. It is visible in Figure 5A as well and forms a coherent area of events right after the previously mentioned line of events. The analog circuits of some of the sensors express a form of ringing noise after a stimulus left a sensor area. It is not yet clear what exactly causes this trailing noise. In some recordings this form of noise is clearly visible for a human observer (data not shown), in others it is difficult to identify if an event is genuine or not.

The characteristic value yields informative knowledge about an event. In fact, it seems as if the value itself may be enlightening enough to discard an event. Early experiments with this idea required manually tuning the classifier to reach comparable results, though. An approach we deem questionable, definitely not satisfactory, as a fully automated process would find wider application. Hence we left an evaluation using only the characteristic value for future work.

In addition, future work will require to evaluate the approach on more datasets. This is necessary to confirm that the principle works on datasets from sensors with higher resolutions and other internal characteristics. Only recently a novel set of ground truth datasets was released [19] and is likely to be used in future evaluations.

6. Conclusion & future work

In this paper we demonstrated a method to construct graphs for event-based data that can be used in graph cut applications. First we showed why it is necessary to extract additional information from a stream of events. Then we argued why a naive approach to extract such information is based on a potentially flawed interpretation of the temporal data stored in the stream. Using a linearity conjecture, we then derived a way to compute a characteristic value for each

event using Kullback–Leibler divergence. This characteristic value in turn can be used to set up weights required in the graph. We successfully applied our technique on a small manually labeled dataset and show very good classification rates even for a huge number of noise events.

Our results suggest that the computation of a characteristic value for events is indeed a suitable approach to automatically compute edge weights for graph cuts. However, the computation of these weights have a significant impact on the runtime of the algorithm. Currently, exponentially many traces are evaluated. Given the linearity conjecture on which we based the characteristic value, it is arguable if all theoretically possible traces need evaluation or if the traces can be pruned to a very small subset. For instance, it may be sufficient to examine only traces that lead to the event on a rectilinear path. In fact, early experiments suggest that this is indeed the case. Another approach is to generally limit the temporal depth. In other words, discarding events that are older than a certain time could reduce the accrued traces. Moreover, it would then be possible to define dedicated filters that can be evaluated in parallel for the whole temporal domain in which the remaining events live. Interestingly this moves our technique close towards Gabor filter banks, an approach that was successfully employed to compute optic flow from event data just recently [9]. Hence the results provoke plenty of further research and development. The computation of the characteristic values is embarrassingly parallel, as each trace segment can be evaluated independently of each other. Therefore pure hardware realizations or improved software implementations that prune the necessary computations are imaginable. A real-time application would be the targeted result of such an attempt.

Further theoretical investigations are likely as well. At the moment, the characteristic value is computed for each event only by examining its own trace. However real-world stimuli usually cover more than just one pixel, hence evoking multiple related traces. We think that changing the way the characteristic value is computed to collect immediate lateral information from neighboring events as well as additional feedback from a larger pool of events will improve our algorithm. The idea for a multi-resolution analysis is based on the lateral connectivity and pooling principles found in the processing of the visual cortex and was already applied to the related task of computing optic flow, however for regular images [2].

Finally, trace segments are not simply a by-product but can be used for other purposes. Inherent in all traces is the temporal information and direction of the stimulus. Hence, it is possible to not only predict where a stimulus is moving. Grouping events by trace similarity and performing segmentation without further graph cuts are just two possible areas of interest.

References

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008. 1

[2] C. Beck, P. Bayerl, and H. Neumann. *Optic Flow Integration at Multiple Spatial Frequencies – Neural Mechanism and Algorithm*, pages 741–750. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. 8

[3] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan. Asynchronous frameless event-based optical flow. *Neural Networks*, 27:32 – 37, 2012. 1

[4] Y. Boykov and G. Funka-Lea. Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006. 2

[5] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, pages 105–112, 2001. 2

[6] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004. 2

[7] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *1998 Conference on Computer Vision and Pattern Recognition (CVPR '98), June 23-25, 1998, Santa Barbara, CA, USA*, pages 648–655, 1998. 2

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001. 2

[9] T. Brosch, S. Tschechne, and H. Neumann. On event-based optical flow detection. *Frontiers in Neuroscience*, 9(137), 2015. 1, 8

[10] L. A. Camuñas-Mesa, T. Serrano-Gotarredona, and B. Linares-Barranco. Live demonstration: Event-driven sensing and processing for high-speed robotic vision. In *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*, pages 183–183, Oct 2014. 1

[11] X. Clady, C. Clercq, S.-H. Ieng, F. Houseini, M. Randazzo, L. Natale, C. Bartolozzi, and R. Benosman. Asynchronous visual event-based time-to-contact. *Front Neurosci*, 8:9, Feb 2014. 24570652[pmid]. 1

[12] X. Clady, S.-H. Ieng, and R. Benosman. Asynchronous event-based corner detection and matching. *Neural Networks*, 66:91 – 106, 2015. 1

[13] J. Conradt. On-board real-time optic-flow for miniature event-based vision sensors. In *2015 IEEE International Conference on Robotics and Biomimetics, ROBIO 2015, Zhuhai, China, December 6-9, 2015*, pages 1858–1863, 2015. 3

[14] B. Dezs, A. Jüttner, and P. Kovács. Lemon - an open source c++ graph template library. *Electron. Notes Theor. Comput. Sci.*, 264(5):23–45, July 2011. 6

[15] M. Firouzi and J. Conradt. Asynchronous event-based cooperative stereo matching using neuromorphic silicon retinas. *Neural Processing Letters*, 43(2):311–326, 2016. 1

[16] D. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, USA, 1962. 2

[17] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal*

of the Royal Statistical Society. Series B (Methodological), 51(2):271–279, 1989. 2

[18] D. G. Harris and A. Srinivasan. Improved bounds and algorithms for graph cuts and network reliability, 2016. 2

[19] Y. Hu, H. Liu, M. Pfeiffer, and T. Delbruck. Dvs benchmark datasets for object tracking, action recognition, and object recognition. *Frontiers in Neuroscience*, 10:405, 2016. 7

[20] J. Kogler, M. Humenberger, and C. Sulzbachner. *Event-Based Stereo Matching Approaches for Frameless Address Event Stereo Data*, pages 674–685. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. 1

[21] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):147–159, 2004. 2

[22] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128 times; 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *Solid-State Circuits, IEEE Journal of*, 43(2):566–576, Feb 2008. 1, 5

[23] C. Liu, F. Li, Y. Zhang, and H. Gu. Interactive image segmentation based on hierarchical graph-cut optimization with generic shape prior. In *ICIAR 2009*, volume 5627 LNCS, pages 201–210, Halifax, NS, Canada, 2009. Springer Verlag. 2

[24] S.-C. Liu and T. Delbruck. Neuromorphic sensory systems. *Current Opinion in Neurobiology*, 20(3):288 – 295, 2010. Sensory systems. 1

[25] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. 1

[26] C. A. Mead and M. Mahowald. A silicon model of early visual processing. *Neural Networks*, 1(1):91 – 97, 1988. 1

[27] C. Posch, D. Matolin, and R. Wohlgenannt. High-dr frame-free pwm imaging with asynchronous aer intensity encoding and focal-plane temporal redundancy suppression. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 2430–2433, May 2010. 1

[28] A. Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2002. 2

[29] C. Sulzbachner, C. Zinner, and J. Kogler. An optimized silicon retina stereo matching algorithm using time-space correlation. In *CVPR 2011 WORKSHOPS*, pages 1–7, June 2011. 3

[30] S. Tschechne, R. Sailer, and H. Neumann. *Bio-Inspired Optic Flow from Event-Based Neuromorphic Sensor Input*, pages 171–182. Springer International Publishing, Cham, 2014. 1

[31] V. Vezhnevets and V. Konouchine. "Grow-Cut" - interactive Multi-Label N-D image segmentation. In *Graphicon*, pages 150–156, 2005. 2

[32] N. Waniek, J. Biedermann, and J. Conradt. Cooperative slam on small mobile robots. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1810–1815, Dec 2015. 1

[33] N. Waniek, S. Bremer, and J. Conradt. Real-time anomaly detection with a growing neural gas. In S. Wermter, C. Weber, W. Duch, T. Honkela, P. Koprinkova-Hristova, S. Magg, G. Palm, and A. Villa, editors, *Artificial Neural Networks and Machine Learning – ICANN 2014*, volume 8681 of *Lecture Notes in Computer Science*, pages 97–104. Springer International Publishing, 2014. 1

[34] D. Weikersdorfer, D. Adrian, D. Cremers, and J. Conradt. Event-based 3d slam with a depth-augmented dynamic vision sensor. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 359–364, May 2014. 1

[35] D. Weikersdorfer, R. Hoffmann, and J. Conradt. Simultaneous localization and mapping for event-based vision systems. In *Computer Vision Systems*, pages 133–142. Springer Berlin Heidelberg, 2013. 1

[36] M. Yu, S. Shen, and Z. Hu. Dynamic parallel and distributed graph cuts. 2015. 2