Contour Detection and Characterization for Asynchronous Event Sensors

Francisco Barranco* [1,2], Ching L. Teo* [1], Cornelia Fermüller[1], Yiannis Aloimonos[1]
[1] Computer Vision Lab, University of Maryland (USA) [2] CITIC, University of Granada (Spain)
{barranco, cteo, fer, yiannis}@umiacs.umd.edu

Abstract

The bio-inspired, asynchronous event-based dynamic vision sensor records temporal changes in the luminance of the scene at high temporal resolution. Since events are only triggered at significant luminance changes, most events occur at the boundary of objects and their parts. The detection of these contours is an essential step for further interpretation of the scene. This paper presents an approach to learn the location of contours and their border ownership using Structured Random Forests on event-based features that encode motion, timing, texture, and spatial orientations. The classifier integrates elegantly information over time by utilizing the classification results previously computed. Finally, the contour detection and boundary assignment are demonstrated in a layer-segmentation of the scene. Experimental results demonstrate good performance in boundary detection and segmentation.

1. Introduction

Event-based computation has been gaining increasing attention in Machine Vision. The Dynamic Vision Sensor (DVS) provides asynchronous responses at pixels where luminance changes, along with very precise timing information in the order of a few microseconds. In other words, it records at high temporal resolution where and when changes in the image occur. This sensor can become a new tool for processing dynamic scenes, especially when there is demand for real-time performance, low computational resources or low latency. Such capabilities currently cannot be exploited by Computer Vision methods using frame-based sensors.

Unlike conventional cameras, the great advantage of the DVS is that it provides very high temporal resolution data at object boundaries. These are the most challenging locations for frame-based image motion estimation, and most of the computations in image motion algorithms are spent there. In conventional Computer Vision, estimation of image motion and the detection of object boundaries are considered two problems coupled in a chicken-and-egg situation. Finding object contours in early processing stages, will greatly facilitate further processes, such as accurate and dense image motion estimation, segmentation, or recognition.

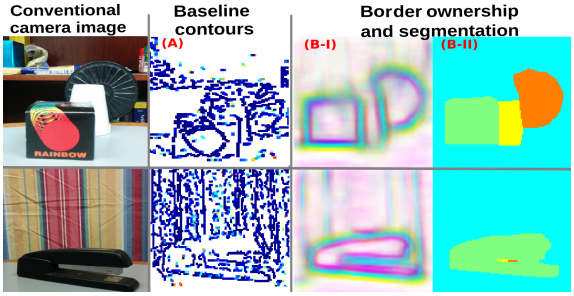


Figure 1. Our approach improves over the simple baseline contours (A), obtained by accumulating DVS events over short time intervals, to produce contours that are not only more accurate but also includes border ownership information (B-I) useful for generating a segmentation of the scene (B-II).

Recent studies have shown the advantages of event-based data for the estimation of local features such as sparse image motion [2, 3, 4], moving corners [8], and tracking [9]. However, the problem of detecting extended contours has not yet been addressed. The asynchronous frame-free representation of the DVS consists of sparse event responses with accurate timing, but they are not grouped into meaningful entities: e.g. object boundaries or segments. It may appear at first sight that this data, because of its sparseness and the fact that there is no image intensity, is not suited well for conventional vision techniques. In this work we introduce an approach, using methods from Computer Vision, but adapted to the event based data, to obtain contours and ordinal depth as shown in Fig. 1.

Neurophysiological experiments suggest that human vision has processes dedicated to linking contours in early visual areas V1 and V2 (see [11]). Further evidence shows that biological vision also has mechanisms for assigning ordinal depth information to contours, via the so-called border *ownership* assignment [31, 33]. Such processes determine which of the two sides next to the boundary belongs to the

*– indicates equal contribution

foreground object and which to the *background*. Inspired by these findings, we propose an approach that learns to detect object border pixels and their border ownership from event data as shown for some objects in Fig. 1 (B-I).

A few works in the event-based literature have grouped local edges. Delbruck [9] locates short edges by selecting events fired in spatial proximity within a small time interval. In [20], the rate of events per time interval for a specific location is used for grouping, with a threshold to select the locations that are most likely part of the same contour. However, none of the existing approaches explicitly locates contours. Thus, our baseline method for comparison groups these events, creating contours if their orientation is similar, and making them thinner with non-maximum suppression as in the Canny edge operator (see Fig. 1 (A)).

The proposed method is the first to detect boundaries directly from events. A learning approach is used to associate patterns of events in spatial neighborhoods and over multiple time intervals with object contours and assign border ownership. In other words, the method localizes object boundaries and assigns which side of the boundary belongs to the foreground or background. The usefulness of the approach is demonstrated in a segmentation application. However, the long contours detected with our method can serve as input to many other processes, such as flow estimation, 3D motion and structure estimation, and recognition.

2. Related work

In this section we describe first the prior works on event-based algorithms and second, conventional Computer Vision approaches on motion-based segmentation and occlusion detection for border ownership assignment.

2.1. Event-based computing

The DVS ([18]) asynchronously records address-events corresponding to scene reflectance changes, with a maximum temporal resolution of 15 μs, and 128×128 spatial resolution. This sensor fires an event every time the log intensity changes by a fixed amount at a position (x, y). An event $ev(x, y, t, p)$ encodes position and time information, along with the polarity p of the event, that is, +1 or -1 depending on whether the log of the intensity increases or decreases by a global threshold.

The first computational approaches on asynchronous event data exploited the high temporal resolution of the sensor to track simple moving objects [9]. Then, different approaches for estimating image motion, such as [4, 2], were presented. Assuming a strong correlation between events fired relatively close in time and at neighboring positions, these methods reconstruct gradients from the number of events at a position to compute the velocity, and/or their exact timestamps. There are also a few approaches on object recognition and detection. The authors in [13] propose a hardware system that detects hundreds of objects per second using orientation information from edges. The method works well for simple plain shapes with sharp edges. Finally, in [24] an approach for object recognition is presented, using a neural network trained on conventional frames that is mapped to an event-driven representation.

2.2. Classic Computer Vision approaches

Related works fall into two areas: 1) motion-based segmentation and 2) occlusion boundary detection.

1) *Motion-based segmentation* uses mainly image motion to consistently segment regions in the input data (usually a video) based on geometric constraints or some motion priors. Earlier works in the so-called structure from motion framework, developed geometric constraints to segment, for an observer undergoing rigid motion, other moving objects in the scene [5, 21]. The classical work of Wang and Adelson [32] partitions an image frame into motion layers that exhibits similar optical flow vectors. The clustering is done via a parametric motion model. Along similar lines, [6] applied the method of level sets on optical flow fields to produce segments exhibiting different motion patterns. The recent work of Papazoglou and Ferrari [23] introduced a fast video segmentation technique that combines an initial foreground segmentation from flow followed by an energy based labeling refinement where temporal smoothness constraints on labels over the entire video are used to segment moving objects.

2) *Boundaries* in an image frame are defined as edge regions where two objects meet. *Occlusion* boundaries are induced by static image cues or motion discontinuities. An important output of occlusion boundary detection is the assignment of ownership to an edge: which side next to the edge belongs to the foreground (closer to the camera) and which to the background. Such ownership assignments are very useful in computational vision approaches, as they provide some geometric information useful for further processing such as for motion segmentation (above), or higher-level processes, such as recognition, spatial reasoning and scene understanding. Numerous works in computer vision have addressed this problem using *static* image cues. Recent works such as [14, 26, 17] have explored the use of multiple local image cues: color, image gradients, junctions, convexity etc. together with global reasoning via Conditional Random Fields (CRFs) to enforce spatial smoothness in the final ownership assignment. Stein and Herbert [28] combined static cues with motion cues from video frames for improving the detection of occlusion boundaries. The approach begins with an oversegmentation into superpixels, and the goal is to determine for each superpixel, whether it is an occlusion edge or not, and if so, which side (foreground or background) it belongs to.

3. Approach

The structure of the event pattern in the regions close to the boundaries, where occlusions occur, can be used for localizing the boundary and determining the relative depth ordering of the regions next to the boundary. We first describe in this section the features derived from DVS in §3.1, that are used to train a Structured Random Forest (SRF) classifier, a variant of the standard Random Forest [12] that predicts *structured* outputs. In this work, we use the SRF to predict reliable boundary locations and ownership assignment from DVS data (§3.2), unlike [30] which considers only RGB images. Finally, we present a sequential extension of the SRF in §3.3 that uses temporal history on a continuous DVS sequence to further improve predictions.

3.1. Features

Several works in the computer vision literature use features such as gradients, contrast, or texture [29, 14, 28, 19] for boundary detection and border ownership assignment in images. However, due to the nature of the DVS, such visual-based features cannot be extracted. For this reason, we have selected here event-based features on the basis of how their evolution in time could help us detecting occlusion boundaries. The features selected for the training are separated into four groups:

Event-based Orientation. Recent works such as [14, 28, 7] explore the use of orientation for contour detection and border ownership. As done in [15], we use the convexity of edges, which is encoded by the edge orientation. The intuition is that concave edges often belong to foreground objects [22]. To estimate the orientation we consider spatio-temporal neighborhoods of 11×11 pixels and 20ms. We only consider eight orientations (from 0 to π). For every new event, first its timestamp is compared to the average timestamp of the events in the neighborhood. If the difference exceeds 10 ms, the event is considered an outlier. Then, a winner-takes-all strategy is used to obtain the most likely orientation for the new event. Next, the orientation is updated, using the previous stored orientation for the location. As another feature, a normalized Histogram of Orientations with 8 bins is computed for patches of 15×15 pixels. Every event-based orientation is extracted and stored with its timestamp.

Event temporal information. Intuitively, the timestamps provide information for tracking contours [9]. Specifically, they define a surface that encodes locally the direction and speed of image motion; and the changes of this time-surface also encode information about occlusions boundaries. In the same way, some authors use the orientation from temporal surfaces extracted from frame sequences as a cue to assign border ownership [28]. Guided by this intuition we collect the following features: the number of events accumulated for different time intervals, the first and last timestamps of the events at every pixel, and the series of timestamps for all the events per location. This last feature captures the surface variability within local patches.

Event-based Motion Estimation. Image motion encodes relative depth information useful for assigning border ownership. The idea is formulated in the so-called *motion parallax constraint*, used in previous works [28, 27, 26], i.e. objects that are close to the camera move faster in the image than objects farther away. To compute image motion we used the method in [4], where a function $\mathcal{T}_e$ that assigns to every position the timestamp of its last event is defined. This function locally defines a surface (in our case of size 5×5). The spatial derivatives of this surface provide the speed and direction of the local motion. With every new event, a local plane that is fitted to the surface is updated within a time interval of 7.5 ms. The plane is updated with a new event if it lies reasonably close (< 0.2 pixels) to the current fitted plane. We repeat this process of iterative plane fitting two times.

Along with the motion estimate, we store for every pixel, the time of the *last* update of the motion estimate. In particular, this timing characterizes the evolution of the motion in time which encodes the type of occlusion/disocclusion, that is encountered for example when the object is moving on top of a changing background.

Event-based time texture. Many works have used spatial texture patterns to detect contours and assign border ownership [19, 26, 14]. The aim in this case is to separate occlusion edges from texture edges. First, texture edges are surrounded by nearby edges with comparable contrasts making them not very salient [26, 25]. Second, long smooth boundaries with strong texture gradients are more likely to occur at occlusions [14]. Instead of intensity texture gradients as used on images, we use a map of the timestamps of the last event triggered at every pixel. This map defines a *time-texture* surface, that is similar to the concept of surface orientation used in [14]. As features we use time-texture maps that are updated continuously with every new event, so there is no need to interpolate information as in the frame-based representations. To this we apply a bank of Gabor filters, using 6 orientations and 3 scales. As features, we use at every scale the maximum response over different orientations at every location.

All these features are estimated using short time intervals at 20 ms, 40 ms, 60 ms, 80 ms, and 100 ms. Examples of these features can be found in the Supplementary Material.

3.2. Border ownership assignment via SRF

We detail in this section how we train an SRF for predicting border ownership given input DVS features. Similar to other works using SRFs ([16, 10]), we assume that the input features $x_f \in \mathcal{X}_f$ are non-structured, while only the output is structured. Each $\mathcal{X}_f \in \mathbb{R}^{N \times N}$ is derived from

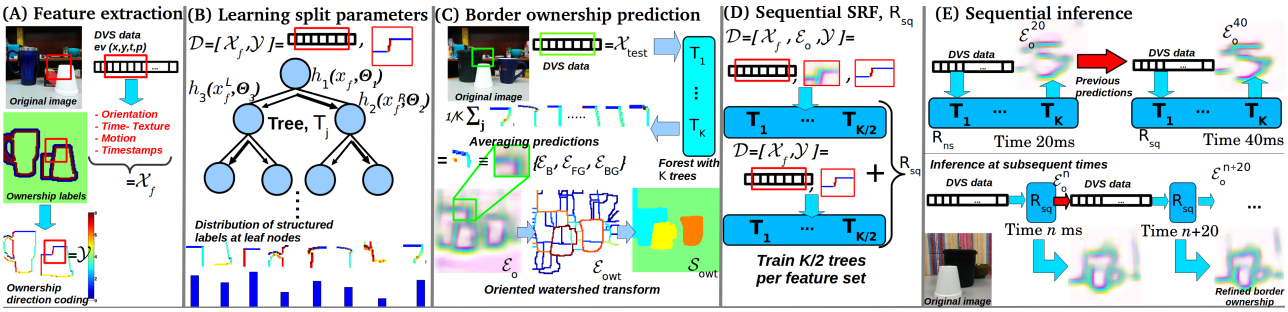


Figure 2. Training an SRF for border ownership assignment. (A) Features $\mathcal{X}_f$ are extracted from the DVS data. (B) During training, we pair features $x_f \in \mathcal{X}_f$ with its ownership annotations, $\mathcal{Y}$ to learn the optimal split thresholds θ for each $h(x_f, \theta)$. (C) During inference, we average the prediction over all K trees to obtain the final ownership prediction $\mathcal{E}_o = \{E_B, E_{FG}, E_{BG}\}$: boundary (blue), foreground (red) and background (yellow). We then obtain E_{owt} by applying watershed transformation over E_B to recover an initial segmentation S_{owt} given a scale threshold. (D) Training a sequential SRF, R_{sq}. We first run R_{ns} (non-sequential SRF) over the training data to provide initial predictions $\mathcal{E}_o$ (left) which is then used as an augmented training feature set $\mathcal{U}_f$ for learning weights in R_{sq} (right). (E) Inference from sequential data. (Above) For the first DVS data at 20ms, we use R_{ns} to predict $\mathcal{E}_o^{20}$. Using the augmented input features, the sequential R_{sq} is then used to produce $\mathcal{E}_o^{40}$ at 40ms. (Below) The process is repeated for all subsequent DVS data using R_{sq}.

a $N \times N$ ($N = 16$) feature patch. The corresponding target output structure, obtained from the same spatial location is a structured label $\mathcal{Y} \in \mathbb{Z}^{N \times N}$ that contains the *orientation* coded annotation of the border ownership which is used as groundtruth during training (Fig. 2 (A)). Using a 8 way local neighborhood system, this amounts to 8 possible directions of border ownership that each decision tree, $T_j \in (T_1, \cdots, T_K)$ will predict. For each tree, the goal is to determine at each split node i, the optimum parameters, θ_i that will send features x_f either to the left or right child nodes (Fig. 2 (B)) via a binary split function $h(x_f, \theta_i) \in \{0, 1\}$. If $h(\cdot) = 1$ we send x_f to the left child and to the right child otherwise. $h(x_f, \theta_i)$ is an indicator function with $\theta_i = (k, \rho)$ and $h(x_f, \theta_i) = \mathbf{1}$ if $[x_f(k) < \rho]$, where k is the feature dimension corresponding to one of the features described in §3.1. ρ is the learned decision threshold that splits the data $\mathcal{D}_i \subset \mathcal{X}_f \times \mathcal{Y}$ at node i into $\mathcal{D}_i^L$ and $\mathcal{D}_i^R$ for the left and right child nodes respectively. ρ is based on maximizing a standard information gain criterion N_i:

$$N_i = H(\mathcal{D}_i) - \sum_{p \in \{L,R\}} \frac{|\mathcal{D}_i^p|}{|\mathcal{D}_i|} H(\mathcal{D}_i^p). \qquad (1)$$

We use the Gini impurity measure $H(\mathcal{D}_i) = \sum_y c_y(1 - c_y)$ with c_y denoting the proportion of features in $\mathcal{D}_i$ with ownership label $y \in \mathcal{Y}$. Since we have structured output labels, we use an intermediate mapping $\Pi : \mathcal{Y} \mapsto \mathcal{L}$ of discrete labels $\mathcal{L}$ following [10] to compute Eq. (1). The process is repeated with the remaining data $\mathcal{D}^p, p \in \{L, R\}$ at both child nodes until a maximum tree depth of $d_t = 64$ is reached or N_i is sufficiently small. During inference, we extract test feature patches and classify them using all K trees. The final ownership label at each pixel is determined by averaging the predicted ownership labels over all trees, producing a direction code that we convert into an *oriented boundary*, $\mathcal{E}_o = \{E_B, E_{FG}, E_{BG}\}$, that respectively encodes the boundary, foreground and background (Fig. 2 (C)).

Given the predicted boundaries with ownership information, we adapt the hierarchical oriented watershed transform (owt) segmentation technique of Arbelaez et al. [1] that converts $\mathcal{E}_o$ into a closed segment. This is done by weighing each of the watershed arcs derived from E_B with the predicted boundary strength to derive the owt transform E_{owt} of E_B. Using a fixed scale threshold (learned from a separate training data), we obtain an initial segmentation S_{owt}.

The runtime performance of the trained SRF is extremely fast. After feature extraction, it takes ≈ 0.05s to process the DVS data for a 128×128 resolution. It takes around 0.01s to estimate S_{owt} from E_B. Training a SRF with $K = 8$ decision trees in parallel takes around 5 to 10 minutes (depending on dataset size) over a cluster with dual Intel-Xeon 2.9GHz CPUs and 128GB of RAM using Matlab.

3.3. A sequential extension to the SRF

The SRF described in §3.2 above and used in the experimental evaluation (§4.2) predicts boundaries and border ownership for input DVS data for each time interval *independently*. For the evaluation, since the DVS data does not have temporal correlation across time, this is sufficient. However, as we will be dealing with DVS data from a *continuous* sequence in §5, it would be desirable that the SRF's prediction uses in addition to the features $\mathcal{X}_f$ some form of temporal history, resulting in a smoother and cleaner prediction over time. We do this by augmenting the existing DVS features used with the output predictions of the *previous* time's DVS data as shown in Fig. 2 (D-E). Specifically, denoting $n = 20$ as the first DVS data at 20ms, we use the existing non-sequential SRF (R_{ns} in Fig. 2 (E)) to produce a prediction of the data, $\mathcal{E}_o^{20}$. For subsequent DVS times, $n+20$, we augment the input DVS features $\mathcal{X}_f^{n+20}$ with the

Sequence	Description	# Objects/ # Layers/ # Motion/{# Train | # Test}
Rotation	Mainly rotational motion	1/1/1/{20 | 20}
Translation	Mainly translational motion	1/1/1/{18 | 18}
Zoom	Mainly zoom motion	1/1/1/{18 | 18}
Complex	Up to 3 objects and clutter, different backgrounds	3/3/3/{73 | 86}
NewObj-NewBG	Only for testing: new objects and backgrounds	3/3/3/{- | 47}
Cars	Moving and static cars in real world	2/2/3/{43 | 36}
Complex-C	Mainly translation + rotation	3/3/2/{-|1}

Table 1. Descriptions of DVS sequences used. Note that "NewObj-NewBG" is a held out testing sequence and "Complex-C" is used only for testing the sequential SRF (§3.3).

previous data's prediction $\mathcal{E}_o^n$ to obtain a larger feature set $\mathcal{U}_f^{n+20} = \mathcal{E}_o^n \times \mathcal{X}_f^{n+20}$ which we then pass into a *sequential* SRF, R_{sq}. R_{sq} is a SRF that contains $K/2$ trees trained using DVS features $\mathcal{X}_f$ and $K/2$ trees trained using $\mathcal{U}_f$ (Fig. 2 (D)) which during inference produces two predictions: $\mathcal{E}_o$ from the DVS features and $\mathcal{E}_{osq}$ from the augmented features. $\mathcal{E}_o$ is exactly what R_{ns} predicts for the current DVS data (with half the number of decision trees) while $\mathcal{E}_{osq}$ is a prediction that takes into account the results from the previous time's DVS data. We use R_{ns} trained from the "Complex" sequence (see Table 1) and retrain R_{sq} using the same sequence in this paper. By choosing $w_f \in [0, 1]$, a weight factor that combines these two predictions, the final prediction is thus defined as $\mathcal{E}_o^{n+20} = w_f \mathcal{E}_{osq} + (1 - w_f)\mathcal{E}_o$.

4. Experiments

We first describe the DVS dataset used and the performance metrics for assessing boundary detection and border ownership assignment accuracies. Next, we detail the experiments that combine DVS features in a series of ablation studies to evaluate their contributions. We then show results of using the sequential variant of the SRF, R_{sq}, and compare it to the non-sequential version, R_{ns}.

4.1. Dataset, baseline and evaluation procedure

In order to investigate in a systematic manner the performance of the proposed approach, we have collected and annotated 7 DVS sequences of varying complexity, summarized in Table 1. The first three sequences depict single objects, and the camera undergoes a single predominant motion. Using these sequences we can investigate the effect of different motions on border detection and ownership. In the experiments we used 15 common objects of different shapes and sizes taken over varying backgrounds and distances to demonstrate the generalizability of the approach. The "Complex" sequence has general rigid motions (rotation + translation + zoom) with a maximum of 3 objects inducing 3 motion layers (excluding the background). In addition, we use for testing a challenge test set ("NewObj-NewBG") with random objects and complex backgrounds not encountered during training. We have also collected an outdoor "Cars" sequence that includes examples of moving and static cars with complex real-world dynamics and backgrounds. The "Complex-C" sequence contains temporally correlated DVS data and is recorded with the sensor moving with a predominantly translation motion in front of 3 objects partially occluding each other. For each sequence, we hand annotated each foreground object with a label indicating its ordinal depth, where foreground objects have smaller depth than the background. The ordinal depth was then converted into ownership labels, $\mathcal{Y}$, along boundaries.

We evaluate the performance of our approach by reporting the F-measure over Precision and Recall (P-R) for assessing ownership and boundary accuracy. For boundaries, we use the evaluation procedure from the Berkeley Segmentation Dataset [19] to generate P-R curves and report the maximal F-score (ODS). For ownership, we compute its F-score, F_{own}, by first matching ownership predictions that are no further than 0.4% of the image diagonal to the groundtruth (see [17]), and we consider a pixel to have the correct ownership when its orientation code is less than 90 degrees from the groundtruth. Finally, the average of the owner assignment and boundary accuracy is reported as a final *combined* measure of performance, denoted as F_c. Since this is the first approach that detects boundaries from DVS data, there are no other methods to compare with. However, we have created a baseline that groups events using their timestamps. This simple method connects edges to create long contours, if they appear in spatial proximity within a small time interval and their orientations match. Moreover, it applies non-maximum suppression as in the Canny edge operator, to make cleaner boundaries.

4.2. Feature ablation experiments

Table 2 summarizes the performance for both the boundary prediction and ownership assignment using the evaluation metrics described in §4.1. We also show in Fig. 4 the P-R curves comparing the performance of boundary prediction for several feature ablations over the DVS sequences.The same parameters noted in §3.2 are used to train the SRFs for all sequences and experiments.

The various feature ablations are selected by training the SRF with a subset of the DVS features that capture certain properties that are sensitive to boundary prediction and ownership assignment. First, we train separate SRFs that used each feature subset (§3.1) separately: [Timestamp (TS) only], [Motion Only], [Orientation (Orient) Only] and [Time-Texture Only]. Next, we train a SRF that uses all features together [All features], with the exception of "Cars" where we used only [Timestamp + Motion]. We highlight key observations and results in the next paragraphs.

Boundary prediction accuracy, ODS. Our approach significantly outperforms the baseline predictions, producing much better boundaries that are closer to the groundtruth.

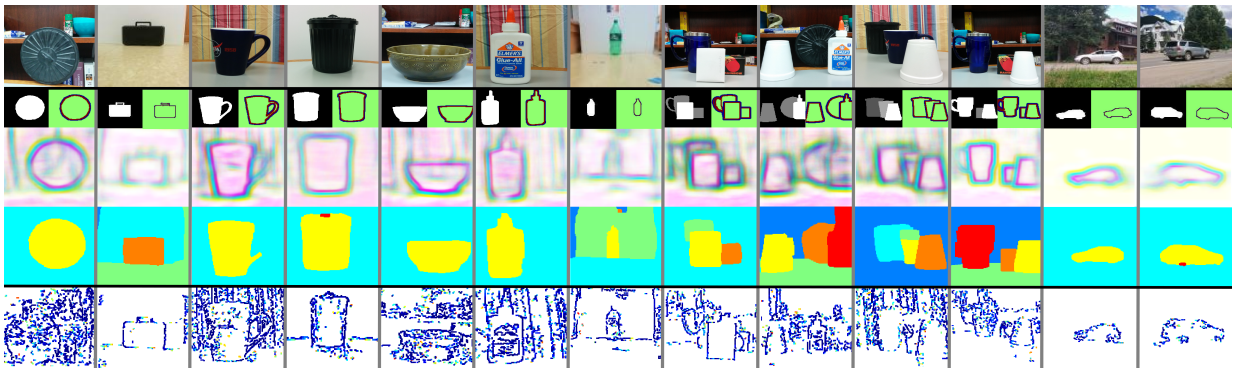


Figure 3. Example results (Top to bottom): Original scene configuration; Hand annotated segmentation and border ownership groundtruths; Predicted boundaries (blue) and ownership (red: foreground, yellow: background) from DVS data; Predicted segmentation from DVS data; Predicted baseline contours. More results are available in the Supplementary Material.

Feature ablations	Rotation	Translation	Zoom	Complex	NewObj-NewBG	Cars
Timestamp Only	**0.394**, 0.641, **0.517**	0.308, **0.591**, **0.449**	0.239, 0.498, 0.368	0.331, 0.569, 0.450	0.255, 0.473, 0.364	0.343, 0.517, 0.430
Motion Only	0.307, 0.558, 0.433	0.271, 0.492, 0.381	0.251, 0.475, 0.363	0.278, 0.522, 0.400	**0.217**, 0.429, 0.323	0.337, 0.510, 0.423
Orientation Only	0.321, 0.570, 0.445	**0.323**, 0.536, 0.429	0.243, 0.494, 0.368	0.311, 0.525, 0.418	0.232, 0.434, 0.333	0.286, 0.463, 0.375
Time-Texture Only	0.268, 0.552, 0.410	0.197, 0.512, 0.354	0.223, 0.492, 0.358	0.248, 0.472, 0.360	0.193, 0.409, 0.301	0.278, 0.426, 0.352
All features	0.373, **0.661**, **0.517**	0.313, 0.578, 0.445	**0.268**, **0.523**, **0.395**	**0.340**, **0.585**, **0.463**	0.255, **0.478**, **0.366**	†**0.344**, **0.519**, **0.431**
Baseline	–, 0.218, –	–, 0.237, –	–, 0.344, –	–, 0.273, –	–, 0.257, –	–, 0.240, –

Table 2. Performance evaluation of feature ablations over different DVS sequences. For every dataset and ablation, each cell reports the $\{F_{own}$, ODS, $F_c\}$ scores described in §4.1. †For "Cars", instead of All features, [TimeStamp + Motion] is reported. See text for details.

Moving on to the individual features, Timestamp (TS) is an extremely strong feature that predicts the spatial location of the object (motion) boundary, yielding the highest ODS scores in all sequences (except for "NewObj-NewBG"). This highlights the importance of further studies into the use of the event timestamps, which is a unique feature of the DVS camera, not present in conventional sensors. Next, we note that in "NewObj-NewBG", time textures yield the most accurate results which may indicate some form of invariance under challenging scenarios not captured by other features. Further experiments with more precise motions, however, are needed to confirm this. Finally, using all features together improves border ownership in all sequences except "Translation" (where TS remains the best). For the outdoor "Cars" dataset that contains mostly *moving* cars, the combination of [Timestamp + Motion] features produce the best results, as expected in a dynamic sequence.

Ownership assignment accuracy, F_{own}. We first note that the best results are obtained by different features for different sequences (motions). This shows that ownership assignment compared to boundary prediction is more complicated to capture from the features we investigated and no single feature accurately predicts ownership reliably across different motions (sequences). Interestingly, we note that even though the combination of all features do not yield the best accuracy, it consistently produces one of the top results which shows the advantage of using the SRF to determine the best feature combination. This also highlights another issue: the dependency of the motion pattern on the 3D motion. We believe that a possible approach in a practical application would be to selectively use features for border ownership characterization, according to the general predominant 3D motion, i.e. depending on the kind of motion (predominant parallel translation, zoom, or rotation) we can use specific SRF classifiers tuned for the predicted motion.

Overall performance, F_c. We note that in spite of the selectivity of features for boundary prediction and/or ownership, the best results (except for "Translation" and "Cars") are obtained when all features are used. This confirms that our choice of features is *balanced* in terms of these two performance criteria and the SRF is trained to make the optimal selection to this end. For "Cars", the combination of timestamp and motion features again result in the best overall performance due to the dynamic nature of the sequence.

We illustrate in Fig. 3 our prediction results using all features compared to the baseline for some DVS examples. We note that qualitatively, not only are our predictions much cleaner and smoother than the baseline, we are able to generate these predictions in *real-time* which is a key requirement for event-based approaches. However, due to the low spatial resolution of the DVS (128×128 pixels), small structures or distant objects (e.g. mug handles) do not provide sufficient information for reliable prediction. A sensor with higher resolution should ameliorate this issue.

4.3. Results of using the sequential SRF, R_{sq}

In this section, we turn our attention to the effects of using the sequential variant of the SRF, R_{sq}, over the

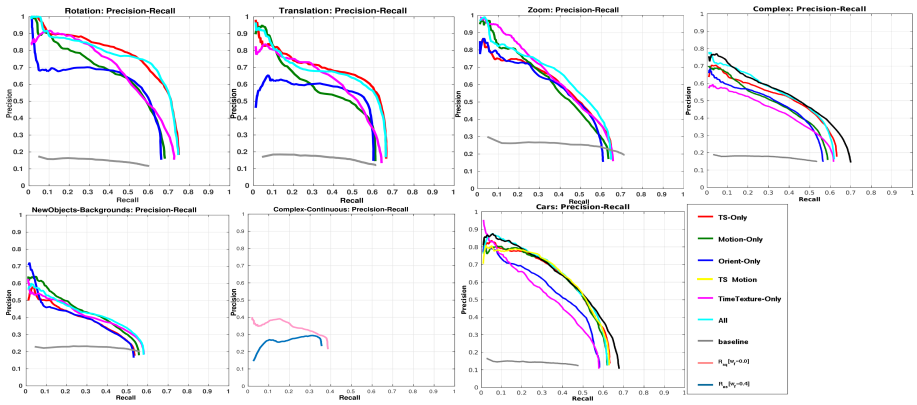
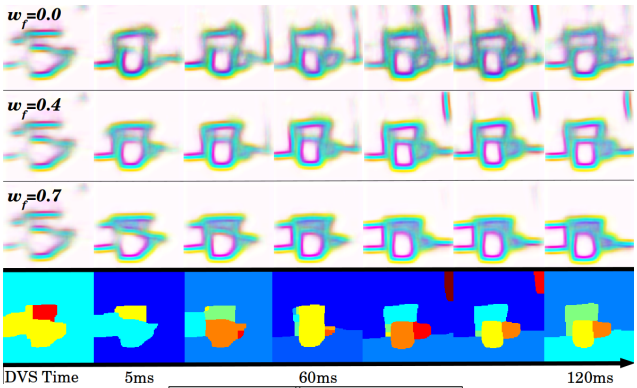


Figure 4. Precision-Recall of boundary prediction accuracy for all DVS sequences. Top row (L-R): "Rotation", "Translation" and "Zoom", "Complex". Bottom row (L-R): "NewObj-NewBG", "Complex-C", and "Cars". See text for details.



SRF	Complex-Continuous
R_{ns} [$w_f = 0.0$]	0.181, **0.324**, 0.252
R_{sq} [$w_f = 0.4$]	**0.212**, 0.310, **0.261**

Figure 5. (Above) Predictions of the sequential SRF, R_{sq} (top to bottom), for $w_f = \{0.0, 0.4, 0.7\}$, using 120 ms of the "Complex-Continuous" sequence. Predictions retain more history with increasing w_f. The last row shows the segmentation results after refining the predictions for $w_f = 0.4$, and the previous segmentation. The new results remove false segments and are more stable. (Below) Evaluation results comparing R_{ns} (non-sequential SRF) with R_{sq}: each cell encodes $\{F_{own}$, ODS, $F_c\}$ scores.

"Complex-C" sequence and compare its results with the non-sequential variant, R_{ns}. We illustrate the effects of using three values of w_f in Fig. 5 (above) over the "Complex-C" sequence: a small w_f results in more noisy predictions while a large one retains more temporal history, some of which are propagated to the subsequent DVS times. We have determined that a value of $w_f = 0.3$ to 0.4 provides reasonable predictions that removes temporally inconsistent predictions while reinforcing the strongest predictions over time. This is confirmed experimentally as shown in Fig. 5 (below) where the sequential variant of the SRF, R_{sq}, outperforms the non-sequential variant R_{ns} (by setting $w_f = 0.0$) in the combined F-score, F_c. Most of the improvement is derived from improving ownership accuracy, at the slight expanse of boundary accuracy (due to the blurring of edges across time), which is also observed in their corresponding P-R curves (Fig. 4 (bottom-right)).

5. Segmentation

An application for the boundary and border ownership predictions from DVS data is segmentation. Next we describe a preliminary segmentation procedure. The initial segmentation, S_{owt}, is first estimated from the predictions $\mathcal{E}_o$ of the SRF via a learned threshold, determined from training data (see §3.2). Next, these segments are refined by enforcing motion coherence between segments (§5.1). Finally, we describe an extension that exploits temporal history from the "Complex-C" sequence for improving the final segmentation prediction over time in §5.2.

5.1. Event-based segmentation and refinement

The initial segments S_{owt}, are further improved by combining/separating regions that exhibit similar/different *motion* patterns. We first estimate the motion pattern per segment using the method in [2], that provides a very precise and sparse motion estimation for DVS. It estimates normal flow at contours (the projection of the flow on the gradient direction). We note here that we did not use the same estimated motion as in the training of the SRF, since this image motion is already encoded in the SRF, and thus would not add new information for refining the segmentation. Abutting segments that have similar motion patterns are merged if they are similar and split otherwise via an iterative search procedure. We use only the motion cues estimated inside these regions and close to the boundaries. The key idea is that texture information in objects far from contours makes the estimation from normal flow more complex. The procedure ends when no new regions can be merged or split anymore. For the split/merge decision, we use a similarity measure defined by the average motion estimates from different regions, which were projected onto the same di-

Sequence	Before refinement, S_{owt}	After motion refinement, S_M
Rotation	0.91, 0.91, 0.40	**0.93**, **0.93**, **0.37**
Translation	0.92, 0.92, 0.42	**0.93**, **0.93**, **0.40**
Zoom	**0.80**, **0.81**, **0.64**	0.79, **0.81**, 0.91
Complex	0.88, 0.89, **0.51**	**0.91**, **0.91**, **0.51**
Complex-C	0.65, 0.66, 1.45	**0.67**, **0.68**, **1.30**

Table 3. Improvement of segmentation accuracy in S_M when motion information is used over several DVS sequences compared to S_{owt}. The performance metrics reported are {ODS, RI, VI}: Mean GT Cover , Random Index and Variation of Information used in [1]. For VI, a smaller value is better.

rection for comparison. Isolated regions or regions with insufficient valid estimates (where the number of estimates is less than 5% of the number of pixels in each region) were assigned according to the initial segmentation S_{owt}. The final motion refined segmentation is denoted as S_M.

We evaluate the refined segmentation using S_M from the first four DVS sequences (excluding "Complex-C"). We use the standard segmentation evaluation metrics: GT-Cover (ODS), Random Index (RI) and Variation of Information (VI) as proposed by [1] and compare S_M with their respective S_{owt} as shown in Table 3. With the exception of the "Zoom" sequence, we can see that imposing motion coherence in general improves the final segmentation using all three metrics. The improvement in the segmentation is largely correlated with the ease of estimating the motion of [2]: the largest improvement occurs in "Rotation" and "Translation", which have simple motions while the improvement is less pronounced for the "Complex" sequence. As reported in [2], the estimates for zoom-like motions are the most difficult, and our slight drop in segmentation accuracy further confirms this observation.

5.2. Continuous refinement for segmentation

In this section, we show how the proposed segmentation approach can be refined continuously over time, as more DVS data becomes available. We do this by first estimating the motion refined segmentation, S_M^n, from S_{owt}^n via the procedure in §5.1 for the current DVS time at n ms. At the next DVS time $n+20$ms, we augment the hierarchical oriented watershed transform (owt) structure E_{owt}^{n+20} from the current DVS time with the *previous* refined segment, S_M^n, to obtain a new owt structure, $U_{owt}^{n+20} = (1-w_{sf})E_{owt}^{n+20} \cup w_{sf}S_M^n$. Using a learned threshold, we can then recover $S_{owt}'^{n+20}$ from U_{owt}^{n+20} and obtain S_M^{n+20} as before. Similar to w_f in R_{sq}, w_{sf} is a weight parameter that determines the amount of temporal history that the current segmentation retains: a larger w_{sf} will retain more history while a smaller w_{sf} may be too noisy. We empirically determined that a value of $w_{sf} = 0.3$ yields the best results. We show in Fig. 5 (above) results of the final motion refined segmentation, S_M, over the "Complex-Continuous" sequence obtained via this procedure. Note that the initial segmentation is very coarse. When more events from these contours are obtained over time, the segmentation is improved and, as we see in the final segmentation provided after 60 ms, the quality of the segmentation is good enough for us to accurately determine the number of objects and their individual shapes. This improvement is confirmed quantitatively by the results reported in the last row of Table 3: the segmentation accuracy of S_M with temporal refinement outperforms the original S_{owt} in all three segmentation performance metrics.

6. Conclusions

We have presented a method for locating object contours with data from the Dynamic Vision Sensor (DVS). The method is based on a Structured Random Forest classifier, that uses as input a set of event-based features. These features represent spatial structure, image motion and temporal evolution of the events, and thus intuitively capture the essential spatio-temporal information characterizing object boundaries. The classifier predicts the locations of objects boundaries and their border ownership assignment, with the inference taking approximately 0.05 s for input from events in time intervals of 20 - 100ms.

Extensive ablation studies on data with different rigid motions and backgrounds, number of objects, and environments (indoors and outdoors) with static and moving objects were conducted. Timestamp features were shown to be the most useful cue for prediction and the method achieved an average combined F-score, F_c, of 0.41 using all features.

To demonstrate the usefulness of the approach, we applied it to the problem of segmentation. In a simple implementation, an initial segmentation obtained through clustering was augmented with depth order relationships, and improved using motion cues in a split/merge iterative process. In future work, we plan to develop more sophisticated segmentation methods that make use of the border ownership assignment and contour detection using a variational framework. We also plan to combine the contour information from DVS with intensity provided along with the events by the new experimental cameras (ATIS and DAVIS).

We consider the proposed method, which we will make available to the community[1], a useful tool for further motion-based processes, such as more sophisticated segmentation, estimation of rigid motion, and motion-based recognition.

Acknowledgments: This work was supported by an EU Marie Curie grant (FP7-PEOPLE-2012-IOF-33208), the EU Project Poeticon++ under the Cognitive Systems program, the National Science Foundation under INSPIRE grant SMA 1248056, the Junta de Andalucia VITVIR project (P11-TIC-8120), and by DARPA through U.S. Army grant W911NF-14-1-0384.

[1]Source code, data and more results are available at http://www.umiacs.umd.edu/research/POETICON/DVSContours/

References

[1] P. Arbelaez, M. Maire, C. C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *CVPR*, pages 2294–2301, 2009.

[2] F. Barranco, C. Fermüller, and Y. Aloimonos. Contour motion estimation for asynchronous event-driven cameras. *Proc. of the IEEE*, 102(10):1537–1556, 2014.

[3] F. Barranco, C. Fermüller, and Y. Aloimonos. Bio-inspired motion estimation with event-driven sensors. In *Advances in Computational Intelligence*, volume 9094 of *LNCS*, pages 309–321. Springer International Publishing, 2015.

[4] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi. Event-based visual flow. *IEEE Trans. on Neural Networks and Learning Systems*, 25(2):407–417, 2014.

[5] T. Brodsky, C. Fermüller, and Y. Aloimonos. Simultaneous estimation of viewing geometry and structure. In *ECCV*, pages 342–358, 1998.

[6] T. Brox, A. Bruhn, and J. Weickert. Variational motion segmentation with level sets. In *ECCV*, pages 471–483. Springer, 2006.

[7] B. Catanzaro, B.-Y. Su, N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer. Efficient, high-quality image contour detection. In *ICCV*, pages 2381–2388, 2009.

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

[9] T. Delbruck. Frame-free dynamic digital vision. In *Int. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, pages 21–26, 2008.

[10] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, pages 1841–1848. IEEE, 2013.

[11] R. F. Hess, K. A. May, and S. O. Dumoulin. *Contour Integration: Psychophysical, Neurophysiological, and Computational Perspectives*. Oxford University Press, 2014.

[12] T. K. Ho. Random decision forests. pages 278–282, 1995.

[13] M. Hofstatter, M. Litzenberger, D. Matolin, and C. Posch. Hardware-accelerated address-event processing for high-speed visual object recognition. In *IEEE Int'l Conf. on Electronics, Circuits and Systems*, pages 89–92, 2011.

[14] D. Hoiem, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. *IJCV*, 91(3):328–346, 2011.

[15] G. Kanizsa and W. Gerbino. Convexity and symmetry in figure-ground organization. *Vision and artifact*, pages 25–32, 1976.

[16] P. Kontschieder, S. R. Bulo, H. Bischof, and M. Pelillo. Structured class-labels in random forests for semantic image labelling. In *ICCV*, pages 2190–2197. IEEE, 2011.

[17] I. Leichter and M. Lindenbaum. Boundary ownership by lifting to 2.1D. In *CVPR*, pages 9–16. IEEE, 2009.

[18] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128×128 120db 15μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circuits*, 43(2):566–576, 2008.

[19] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, 2004.

[20] E. Mueggler, B. Huber, and D. Scaramuzza. Event-based, 6-dof pose tracking for high-speed maneuvers. In *IROS*, pages 2761–2768, 2014.

[21] A. Ogale, C. Fermüller, and Y. Aloimonos. Motion segmentation using occlusions. *PAMI*, 27(6):988–992, 2005.

[22] S. E. Palmer. *Vision science : photons to phenomenology*. MIT Press, 1999.

[23] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, pages 1777–1784, 2013.

[24] J. Perez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets. *PAMI*, 35(11):2706–2719, 2013.

[25] X. Ren. Multi-scale improves boundary detection in natural images. In *ECCV*, volume 5304, pages 533–545. Springer Berlin Heidelberg, 2008.

[26] X. Ren, C. C. Fowlkes, and J. Malik. Figure/ground assignment in natural images. In *ECCV*, pages 614–627. Springer, 2006.

[27] M. E. Sargin, L. Bertelli, B. S. Manjunath, and K. Rose. Probabilistic occlusion boundary detection on spatio-temporal lattices. In *ICCV*, pages 560–567, 2009.

[28] A. N. Stein and M. Hebert. Occlusion boundaries from motion: Low-level detection and mid-level reasoning. *IJCV*, 82(3):325–357, 2009.

[29] P. Sundberg, T. Brox, M. Maire, P. Arbeláez, and J. Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *CVPR*, pages 2233–2240, 2011.

[30] C. L. Teo, C. Fermüller, and Y. Aloimonos. Fast 2D border ownership assignment. In *CVPR*, pages 5117–5125, 2015.

[31] R. von der Heydt, T. Macuda, and F. T. Qiu. Border-ownership-dependent tilt aftereffect. *J. Opt. Soc. Am. A*, 22(10):2222–2229, Oct 2005.

[32] J. Y. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Trans. on Image Processing*, 3(5):625–638, 1994.

[33] H. Zhou, H. Friedman, and R. von der Heydt. Coding of border ownership in monkey visual cortex. *J. Neuroscience*, 20:6594–6611, 2000.