INVITED PAPER

Contour Motion Estimation for Asynchronous Event-Driven Cameras

Emulating the relevant transient retinal responses holds the key to efficient tracking in a visual motion camera.

By Francisco Barranco, Cornelia Fermüller, and Yiannis Aloimonos

ABSTRACT | This paper compares image motion estimation with asynchronous event-based cameras to Computer Vision approaches using as input frame-based video sequences. Since dynamic events are triggered at significant intensity changes, which often are at the border of objects, we refer to the event-based image motion as "contour motion." Algorithms are presented for the estimation of accurate contour motion from local spatio-temporal information for two camera models: the dynamic vision sensor (DVS), which asynchronously records temporal changes of the luminance, and a family of new sensors which combine DVS data with intensity signals. These algorithms take advantage of the high temporal resolution of the DVS and achieve robustness using a multiresolution scheme in time. It is shown that, because of the coupling of velocity and luminance information in the event distribution, the image motion estimation problem becomes much easier with the new sensors which provide both events and image intensity than with the DVS alone. Experiments on synthesized data from computer vision benchmarks show that our algorithm on combined data outperforms computer vision methods in accuracy and can achieve real-time performance, and experiments on real data confirm the feasibility of the approach. Given that current image motion (or so-called optic flow) methods cannot estimate well at object boundaries, the approach presented here could be used complementary to optic flow techniques, and can provide new avenues for computer vision motion research.

KEYWORDS | Asynchronous event-based vision; motion contour; neuromorphic devices; real-time systems

I. INTRODUCTION

Computational approaches to the interpretation of motion, after many years of research, still face significant challenges in real-world scenarios. Classic frame-based image acquisition often does not provide sufficiently high temporal resolution to obtain motion signals accurately enough to compute world models. Furthermore, the images contain much more information than is needed for processing motion, thus resulting in significant amounts of unnecessary memory usage and computational resources [1].

Transient retinal responses are the biological solution to that problem. The transient pathway in the retina asynchronously transmits in parallel the responses from cells that are sensitive to temporal variations in the luminance [2]–[4]. Neuromorphic engineers have been working for a long time on determining the origin and mimicking the behavior and transmission of these retinal responses; the dynamic vision sensor (DVS) is the result of this work. It provides event-based asynchronous responses to changes in the scene reflectance. This means that with this kind of sensor we efficiently transmit only the information from changing elements in the scene, providing not only their position but also the very precise

Manuscript received May 17, 2014; revised July 28, 2014; accepted July 30, 2014. Date of publication September 11, 2014; date of current version September 16, 2014. This work was supported by the European Union (EU) under the grant Poeticon++ in the Cognitive Systems program, and by the National Science Foundation under an INSPIRE grant in the Science of Learning Activities program. The work of F. Barranco was supported by an EU Marie Curie fellowship (FP7-PEOPLE-2012-IOF-33208), and the CEI-GENIL Granada (PYR-2014-4).

F. Barranco is with the Department of Computer Science and the University of Maryland Institute for Advanced Computer Studies (UMIACS), University of Maryland, College Park, MD 20742 USA, and also with the Department of Computer Architecture and Technology, CITIC, ETSIIT, University of Granada, Granada E-18071, Spain (e-mail: barranco@umiacs.umd.edu).

C. Fermüller and **Y. Aloimonos** are with the Department of Computer Science and the University of Maryland Institute for Advanced Computer Studies (UMIACS), University of Maryland, College Park, MD 20742 USA (e-mail: fer@umiacs.umd.edu; yiannis@umiacs.umd.edu).

Digital Object Identifier: 10.1109/JPROC.2014.2347207

time at which they occur. We consider these sensors an opportunity to create real-time solutions to important problems in computer vision while at the same time saving a lot of resources.

Our goal is to understand the computational advantages we gain from event-based information. Thus, let us look at the basic visual motion processes that any system has to solve. The early visual motion pathway in mammalian vision includes the processes of image motion estimation, estimation of the observer's 3-D motion (or egomotion), and segmentation.

The conventional approach of computer vision is to compute, in a first stage, the image motion field (called optical flow field) everywhere in the image, and then use it in further stages for segmentation and 3-D interpretation of the scene. However, the many feedback loops found from neural studies [5]–[7] indicate that biological systems do not perform the computations in that way. Considering the computational challenges involved, we argue that the different processes are intertwined and should be solved by iterative refinement. This suggests an idea for a better computational approach [8], [9]: early on we derive a segmentation into differently moving objects and different surfaces and compute a rough estimate of image motion; then, we compute the egomotion, and after that we can estimate a more accurate scene segmentation and image motion.

The role of the DVS in this approach is essential. The first stage of segmentation and image motion estimation is provided by the event-based asynchronous signal. The motion contours are salient with this sensor, and therefore the problem lies in the software solution for their estimation.

This has been understood by other researchers, who have developed solutions to some of these problems. Examples are: optical flow and depth estimation [1], [10]–[12], tracking [13]–[15], segmentation and 3-D reconstruction [13], [16], and classification [17].

In this paper, we first describe a method for estimating image motion from DVS data only based on the local spatio–temporal event distribution. Though there are other previous approaches of image motion estimation on DVS data, the proposed method is the first to consider the effect of the gray levels of real images on the value of the velocity. Second, we introduce the first method for estimating image motion from both event data and gray-level data. We show, that if we want to account fully for the effects of the gray levels we would need quite intensive computations if using DVS data alone, but by using both DVS events and intensities we can estimate image motion very accurately and in real time.

The paper is structured as follows. In Section I-A, we describe the new asynchronous event-driven neuromorphic sensor. Section II presents the main problems that conventional motion estimation strategies face. Section III describes the new proposed algorithms for contour motion estimation. Section IV shows how these event-based algorithms solve the problems presented in Section II. The experimental results and the validation of the proposed methods are detailed in Section V. Finally, Section VI presents the conclusions.

A. Dynamic Vision Sensors

There are several examples of event-driven sensors in the literature. The DVS [1] uses address–event representation (AER), has a resolution of 128×128 pixels, and simulates the transient responses based on the variations in the image scene reflectance. Every pixel (x, y) of the sensor independently fires an event when detecting a difference in the log of the intensity I that is greater than a fixed threshold τ, i.e., when

$$|\Delta(\log(I(x, y, t)))| > \tau. \tag{1}$$

In our case, τ is 0.1 (i.e., the intensity changes by 10%). A generated event contains information on the position, the time (the temporal resolution is approximately 1 μs), and the polarity ($+1$ or -1) of the event.

Fig. 1 shows an example of DVS output. The left image is a 2-D representation of the accumulation of events for a period of 250 ms (positive and negative are represented as bright and dark gray values). The right image illustrates the spatio–temporal distribution of the events for that time interval.

The proliferation of these sensors in the last few years has demonstrated an increasing interest from the vision research community. For example, Costas-Santos *et al.* [18] present an AER retina based on spatial contrast as sustained cells in the retina; Azadmehr *et al.* [19] show a foveated AER retina; and Mallik *et al.* [20] present a synchronous AER sensor sensitive to temporal changes based on a threshold, although, in this case, with frame-based temporal resolution.

The relevance of the new form of image acquisition for problems of computer vision has been realized, and new sensors such as the asynchronous time-based image sensor (ATIS) [21] and the dynamic and active-pixel vision sensor (DAVIS) (former ApsDVS [22]) have recently been developed. These sensors, in addition to the AER, also include absolute scene reflectance values to avoid blindness to static regions. The ATIS sensor asynchronously updates the gray-scale values, estimated as the inverse of the time it takes to accumulate a certain photon count after a DVS event occurs at a point. The DAVIS sensor, on the other hand, integrates the asynchronous DVS with the so-called active pixel sensor (APS), a frame-based synchronous sensor acquiring intensity. The combination of dynamic and static information in these sensors is considered to provide a framework for solving problems such as tracking and segmentation (with the DVS data),

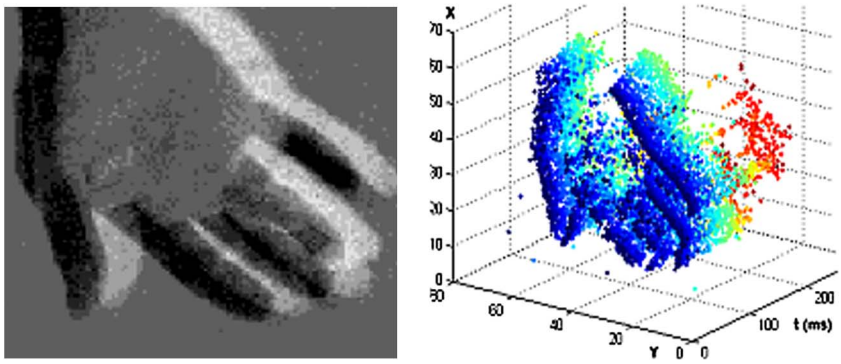

Fig. 1. *Dynamic vision sensor output for a waving hand. The left shows an image formed by collecting events for 250 ms; the right shows a view of the events' spatio–temporal representation.*

and recognition and classification (with the frame-based information). Here we show that the static intensity information also helps to improve image motion estimation.

II. MOTION ESTIMATION

In the field of computer vision the representation of motion in the image is termed the optical flow, and is computed from the change of the brightness pattern in the image over time. Many techniques have been developed, but accurate estimation of optical flow is still considered a challenging problem. Existing techniques are classified into gradient-based methods, frequency-domain methods, and correlation methods, depending on whether the computations are based on the derivatives of the image intensity, local image frequencies derived from filters, or region-based matching. The top algorithms today use image gradients and follow the formulation in the seminal work of Horn and Schunck [23] with different modifications. Let us denote the image intensity as a function $I(x, y, t)$ of the spatial variables (x, y) and time t, and denote the instantaneous motion of a point in the image by a velocity vector $\vec{v}$ with horizontal and vertical components (u, v). The basic formulation states that the intensity at a point over a small time interval δt from time t to $t + \delta t$ is constant, i.e.,

$$I(x, y, t) = I(x + u\delta t,\ y + v\delta t,\ t + \delta t). \tag{2}$$

Developing the above equation using a first-order Taylor expansion, we obtain the following constraint relating the spatial and temporal derivatives I_x, I_y and I_t and the optical flow:

$$I_x u + I_y v = -I_t. \tag{3}$$

Equation (3), called the optical flow constraint, defines at a point the component of the flow in the direction of the spatial image gradient (I_x, I_y). Clearly, from one equation, only one component of the 2-D optical flow (u, v) is defined. This is often referred to as the aperture problem: the image intensity within a small neighborhood usually has one dominant gradient direction, and thus we only can observe the movement parallel to the gradient, or perpendicular (normal) to the edge in the image. Thus, this component is termed the normal flow. To solve for the two components of the optical flow, additional assumptions about the flow in a neighborhood have to be enforced.

In this work, in our comparison, we will use normal flow only. The reason is that deriving the optical flow from the normal flow is an independent computation, and it is not interesting for the evaluation. Furthermore, the DVS provides data only at locations of strong contrast and, at many locations, one major gradient direction making the computation of optical flow ill-posed. Since most DVS events occur at the border of objects, we refer to the normal flow computed there as contour motion, i.e., the image motion perpendicular to contours. In the form of equations, the normal flow (or the contour flow) is the projection of the optical flow on the gradient direction. From (3), we can write the normal flow vector (u_n, v_n) at a point on the contour as

$$(u_n, v_n) = \frac{-I_t}{\|\nabla I\|^2} \nabla I \tag{4}$$

where I_t is the temporal derivative, and $\nabla I = (I_x, I_y)$ is the vector of spatial derivatives of the intensity (the gradient) at the contour point.

Standard optical flow methods, however, attempt to compute both components of the optical flow by enforcing constraints on the flow field. In mathematical terms, they

regularize the constraint in (3) by modeling how the flow varies in a spatial neighborhood. Most current algorithms model it as a variation of the first-order derivatives of the flow. The optical flow field (the flow at every image point), denoted by the horizontal and vertical components $u(x, y)$ and $v(x, y)$, then is computed by minimizing a global energy functional of the form

$$\iint \omega_d(I_x u + I_y v + I_t) + \omega_s(u_x, u_y, v_x, v_y) d_x\, d_y \quad (5)$$

where we dropped the dependence of the variables on the image locations x, y. ω_d and ω_s denote the penalty functions for the data (3) and the smoothness, respectively.

Many state-of-the-art algorithms use two consecutive frames only. They use different penalty functions; for example, the L_1-based minimizations have been shown to be very successful [24], [25]. They take advantage of new optimization methods and modern implementation practices. In addition, various preprocessing techniques, such as median filtering within the multiscale image warping, and extra steps to locate discontinuities near object boundaries help improve the accuracy.

Nevertheless, there still exist fundamental problems that mainly are due to the discontinuities in the motion field and lead to errors at the boundaries of objects. Referring to the minimizations above, first, even in the optical flow constraint equation (3), we implicitly assume that locally the image intensity is a smooth function, i.e., there is only one motion and the spatio–temporal image intensity is well approximated by a linear function. When the motion is large, or there are occlusions, these assumptions are violated. Second, the optical flow is not a smoothly varying function at the boundaries, and thus the regularization term is problematic, and has to be used carefully there. State-of-the-art algorithms use a combination of techniques to locate boundaries, and apply the smoothness constraints appropriately, but in many situations the problems are hard to solve.

Next, we analyze in more detail the major issues of frame-based optical flow estimation techniques.

- Fast motions cannot be accurately handled. In a frame-based framework, fast motions are equivalent to large interframe displacements. The current solution consists in using an iterative coarse-to-fine estimation approach. The image is down-sampled to several coarser spatial resolutions to create what is known as the pyramid. Then, motion estimation is performed for the coarsest image in the pyramid. This motion is used in a compensation function to warp the image at the next level in the pyramid. The residual motion is computed between the warped images, and this motion is used as input for the processing at the next lower level, and so on. This way the flow estimate is gradually refined. The main problem is that this strategy is prone to propagate errors along the scales. Using iterative warping at every scale and nonlocal regularization terms between scales to filter out outliers (such as median filters) helps alleviate the problem, but does not eliminate it. Especially, there are problems with large and small motion in the same image, and small objects moving very fast. Due to their high temporal resolution, DVSs are considered a good solution. As the DVS samples the intensity function more frequently, fast motion does not cause large image displacements between samplings, and it is thus not a problem. They also allow us to save a lot of resources, because the coarse-to-fine approaches are very demanding in terms of memory and computational load. Finally, we note that another solution to limit the problem of estimating fast 3-D motions would be to increase the frame rate using expensive high-speed cameras, but such cameras provide redundant information demanding much more processing. As mentioned, the DVS has a temporal resolution of 1 μs, simulating 10^6 frames per second (fps).
- Textured regions are more reliable than boundaries. All optical flow methods require constraints on the flow in a spatial region and a smoothing over spatio–temporal regions. This works well within image regions originating from textured 3-D surfaces. The major problem comes from the discontinuities; the image regions there cover at least two different physical motions and contain occlusions, causing techniques to fail. The origin of the problem lies in the difficulty of estimating accurately motion at the boundaries, but this also makes it difficult to localize the motion boundaries, turning this into a chicken-and-egg problem. On the other hand, DVS data give us the spatially localized information at very high temporal resolution. This means that we can look for events between neighboring pixels and localize differently moving regions, and then in turn compute well the image motion at the boundaries.
- Large memory resources and heavy computational processing are required for conventional sensors. By only firing events when and where there are intensity variations, DVS helps us save a lot of resources. Furthermore, we only estimate normal motion along the edges, which requires less computation, but is sufficient for computing 3-D motion (see [26]–[28]), and this can be used in turn to address structure from motion and accurate segmentation [9], [29].

III. EVENT-BASED MOTION CONTOUR ESTIMATION

First, this section describes recent work using asynchronous event-based data to estimate image motion. Next, we present our alternative to image motion estimation using asynchronous event-based data. Finally, we present a method for a new camera model, namely the DAVIS sensor, that combines asynchronous event-based data with synchronous frame-based intensity recordings.

A. Prior Work

The first approach to image motion estimation in the DVS uses a form of correlation [30]. The method basically matches events within spatial windows. First, an event-based orientation filter locates events that are close-by in time and lie on a line along some orientation. Then, a search is performed to find a similar pattern of events of the same orientation within a local window. The search is limited to the normal direction (because of the aperture problem), but this significantly reduces the accuracy in estimation, especially at nonstraight boundary segments.

A similar idea was developed in [14] for hand gesture recognition. Lee *et al.* propose a method for estimating the so-called direction selective filter (DSF), which detects the direction of motion within small regions using spiking neural networks. It consists of a two-layer structure, inspired by models of the primary cortex [31]: a first local layer with multiple neurons sensitive to different motion directions and their respective winner-take-all (WTA) circuits, and a second global layer with one WTA circuit for combining the weighted outputs of the first one. Again, the accuracy of the results and the aperture problem are the major issues.

In [32], a method for estimating optical flow, which adapts the concepts of gradient-based techniques, and specifically the Lucas–Kanade method [33], was presented. The flow constraint is reformulated using instead of intensity the accumulation of events during a small time interval. However, the approximation of the spatial gradient from accumulated events is quite inaccurate, and thus speed cannot be estimated well.

Benosman *et al.* [11] introduced a very interesting approach that takes advantage of the inherent temporal nature of events. The method makes use of the exact timing information provided by the DVS. The inverse velocity is computed from the horizontal and vertical derivatives of the function defined by the timestamps. To reduce the impact of outliers, a regularization based on plane fitting is used, similar to the total least squares minimization, previously employed with the optical flow constraint (3) [34]. In a 3×3 spatial neighborhood and predefined time interval, a plane is fitted to the time-stamps of the events.

The method works well and is very efficient. However, it also does not fully consider the effect of the spatial intensity gradient on the events. A strong contrast edge creates multiple events at a point. The method will give perfect estimates, if at every point the last of these multiple events has arrived, which is usually correct for strong contrast edges. In some way, the method assumes that local intensity edges are tracked and that the fitted plane is due to the firing of the same physical points. At wider edges (edges with ramp profiles) multiple points on an edge may fire within the same time interval, possibly causing a confusion. The regularization term makes the estimation more robust. Since the neighborhood used for estimation is small, at most locations the visual flow corresponds to the normal flow, and in textured regions it will correspond to the optical flow.

Finally, a spike-based neural implementation of the idea was developed in [12]. Time delays are added to the synapses to create receptive fields that are sensitive to specific spatio–temporal stimulus.

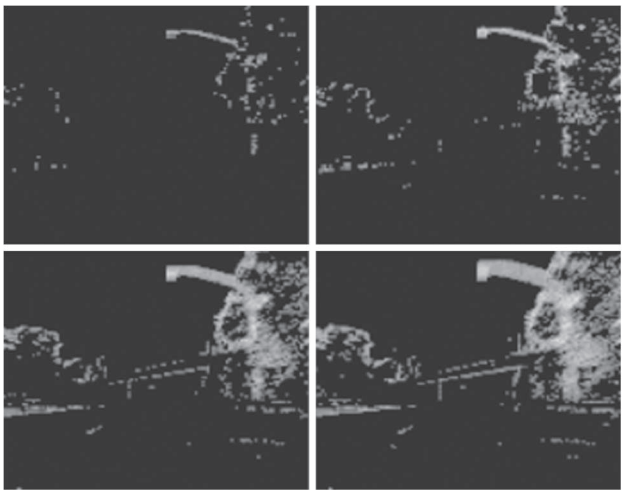

Fig. 2. *Events collected for 10, 25, 50, and 75 ms for the Pasadena highway sequence [37]. Note the difference in the width of the motion boundary for far and close objects in all the images.*

B. Event-Based Motion Estimation

Imagine a vertical contour translating in the horizontal direction. As the contour traverses from left to right, the DVS will record events. The amount of events at a single pixel location and the exact timing depend on both the image contrast and the speed; they are proportional to the product of contrast and speed. For example, an edge with a contrast 2τ (measured in the DVS as the change in the logarithm of intensity) moving with velocity 1 pixel/s will produce the same amount of events as an edge of contrast 1τ moving with velocity 2 pixels/s. Thus, using the local change of events only, as in gradient-based optical flow

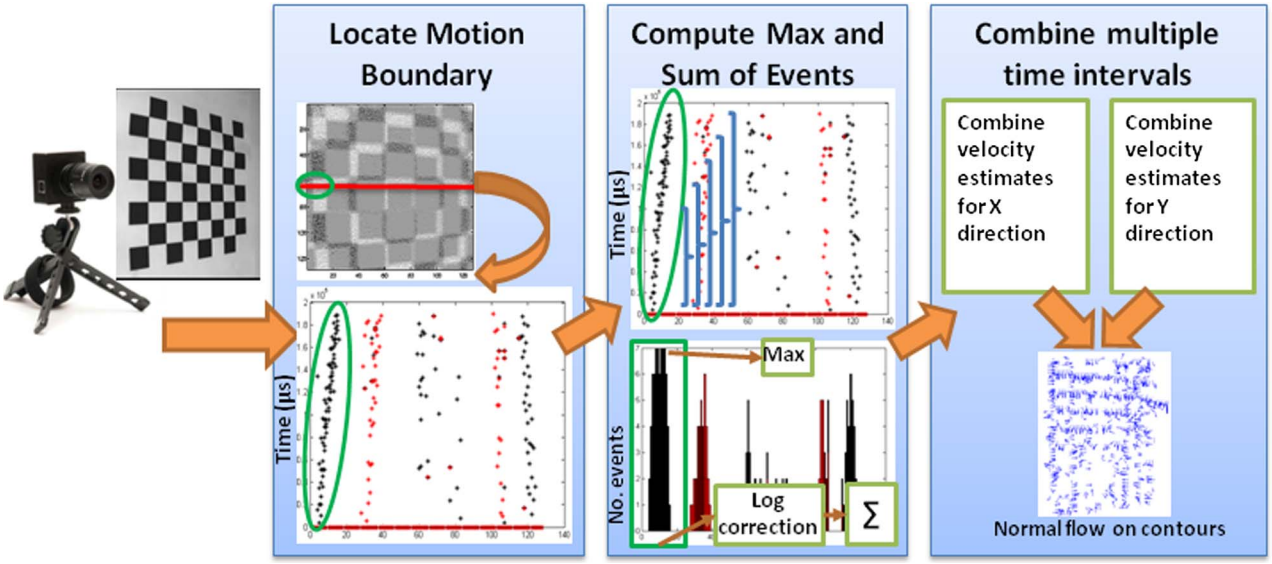

Fig. 3. ***Overview of the motion estimation on contours using DVS (from left to right): DVS sensor [1], motion boundary location for a receding chessboard pattern, estimation of maximum and sum of events on the contour for different temporal resolutions, combination for X- and Y-directions of multiple temporal estimates, and final motion estimation (normal flow) on the contour.***

techniques, it would not be possible to disentangle contrast and speed, to compute image velocity.

On the other hand, the distance traversed by points on the contour provides the speed. Thus, if we could accurately track contour points we could infer speed, but accurately tracking single points is very challenging. We thus take an approach that in some way combines tracking- and gradient-based ideas by observing the movement of contour points over different time intervals.

As the contour moves from left to right, it creates a trace that only depends on the speed [35], [36]. The idea of our method is to locate this trace. Visually this trace can be seen in the width of a boundary, which we call the motion boundary. That is, if we consider all the events within a small time interval, we obtain a motion boundary at the locations where the contour traces the image, whose width is a measure of the contour's speed. Fig. 2 illustrates the concept. It shows four images accumulating events for 10, 25, 50, and 75 ms, respectively. The data come from a driving scenario, with an onboard DVS recording from a car [37]. The scene shows a bridge in front, which crosses the road, trees on both sides, and a street light on the right. Examining the event distributions, we can see that farther objects have narrower boundaries because of lower image velocity, while closer objects such as the street light on the right, have wider boundaries, because of higher velocity.

Fig. 3 provides an overview of our method. First, the motion boundary has to be located. Then, we approximate the temporal and spatial derivatives using the maximum and the total number of events along the motion boundary, respectively. Next, values are corrected due to the logarithmic input. The same process is performed for multiple intervals of time, computing multiple velocity estimates for X- and Y-directions, that are eventually combined to obtain the final speed. Sections III-B1–III-B4 explain these components in detail.

1) Locating Motion Boundaries: First, we have to define the group of connected pixels which make up the motion boundary. We will estimate this boundary separately for the vertical and horizontal direction to obtain the vertical and horizontal components of the motion u_n and v_n. Thus, the following definitions are 1-D.

Let $ev(p,t)$ denote the individual events at spatial location p and time t, which can take on values $+1$ and -1. Let $\mathcal{C}(p)$ denote the group of connected pixels where p is the first position in the boundary with the first event in time, and e is the last position. Let $\mathcal{E}(q)$ denote the number of events at a point q. To refer to timestamps we use the notation $\mathcal{F}(q)$ and $\mathcal{L}(q)$ for the time of the first and last events at point q, respectively.

Since real-world data include noise, we first need to specify how we estimate the motion boundary. For a given time interval Δt events are collected. The motion boundaries are found by locating groups of connected pixels, with the condition that each pixel in the group has at least one event. Furthermore, for any pixel q with more than one event, we require the last event at location q to precede the last event at the consecutive position, and similarly for the first event. The index of the consecutive position (whether it is increasing or decreasing)

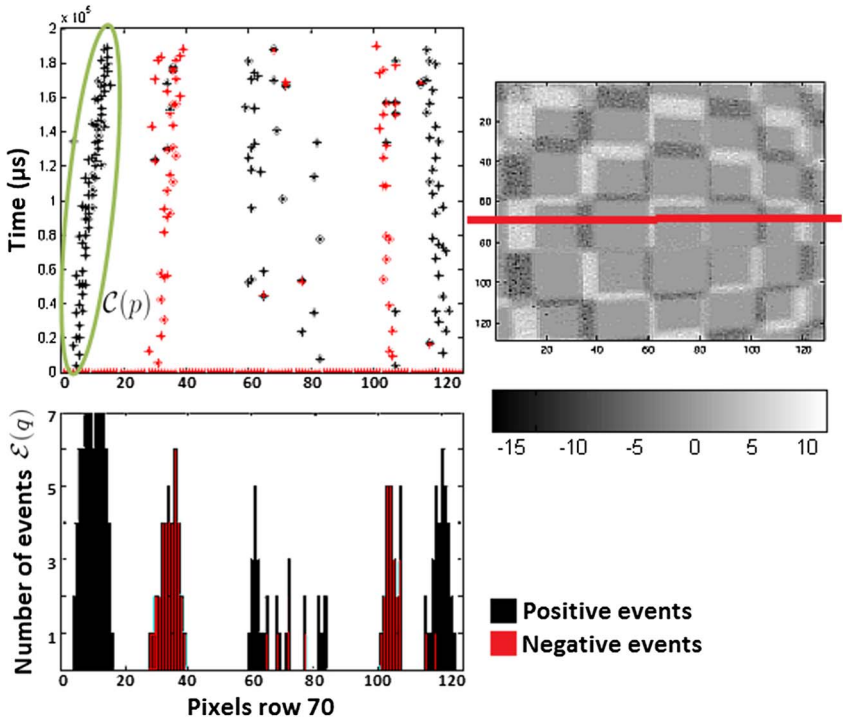

Fig. 4. *Events collected with the DVS camera zooming out from a chessboard pattern. The first row shows on the left the collected positive (black) and negative (red) events for row 70, and on the right an example of the output of the DVS, showing in grayscale the accumulated events per pixel (white for positive and black for negative events). The second row shows the events accumulated for every pixel of row 70. Finally, the ellipse marks all the events of the first motion boundary $\mathcal{C}(\boldsymbol{p})$, and the second row shows $\mathcal{E}(\boldsymbol{q})$ for every location $\boldsymbol{q}$.*

determines the sign of velocity. For practical purposes, we set a threshold T ($T = 50$ μs in the experiments) and require

$$\begin{cases} \sum\limits_{q\in\mathcal{C}(p)} \text{sign}(\mathcal{L}(q+1)-\mathcal{L}(q)) + \\ \sum\limits_{q\in\mathcal{C}(p)} \text{sign}(\mathcal{F}(q+1)-\mathcal{F}(q)) > T, \\ \qquad \text{if } \text{sign}(u(p)) = 1 \\ \sum\limits_{q\in\mathcal{C}(p)} \text{sign}(\mathcal{L}(q+1)-\mathcal{L}(q)) + \\ \sum\limits_{q\in\mathcal{C}(p)} \text{sign}(\mathcal{F}(q+1)-\mathcal{F}(q)) < -T, \\ \qquad \text{if } \text{sign}(u(p)) = -1 \\ \text{otherwise}, \qquad \mathcal{C}(p) = \emptyset. \end{cases} \tag{6}$$

The localization of motion boundaries is done for every row and every column, and separately for positive and negative events. Fig. 4 shows an example of real events from a sequence, where the DVS sensor recedes from a chessboard pattern. Looking at the event distribution for the positions in the same row (marked with a solid red line), we can see the different boundary widths. In this case, the image velocity increases from the center of the image to its border, and is positive for the left side of the image, and negative for the right.

A further illustration is given in Fig. 5 for the sequences "translation tree," one of the sequences used in the optical flow comparison (Section V-A). The events were simulated using the ground truth flow to interpolate 3300 "virtual" images between consecutive frames in the sequence. In the left column, the gray-level image is used, and in the right column, the same image is binarized, thus turning all edges into step edges. One can see that for the binarized image the motion boundary is well localized, and the velocity is positive. However, for the gray-level image localizing the boundary is not so obvious, and one needs to first localize the first and last events in order to even determine the sign of velocity.

2) Velocity Estimation: We accumulate events over some time interval Δt. The 1-D velocity $u(p)$ then is computed as $D(p)$, the displacement in the image that p moves during the time interval Δt. However, we require an estimate of the displacement at subpixel accuracy. We can obtain such an estimate from the number of events.

As a contour point in the image moves, it creates a trace. Let us for the moment ignore that the DVS records events from logarithmic changes of intensity, but assume a

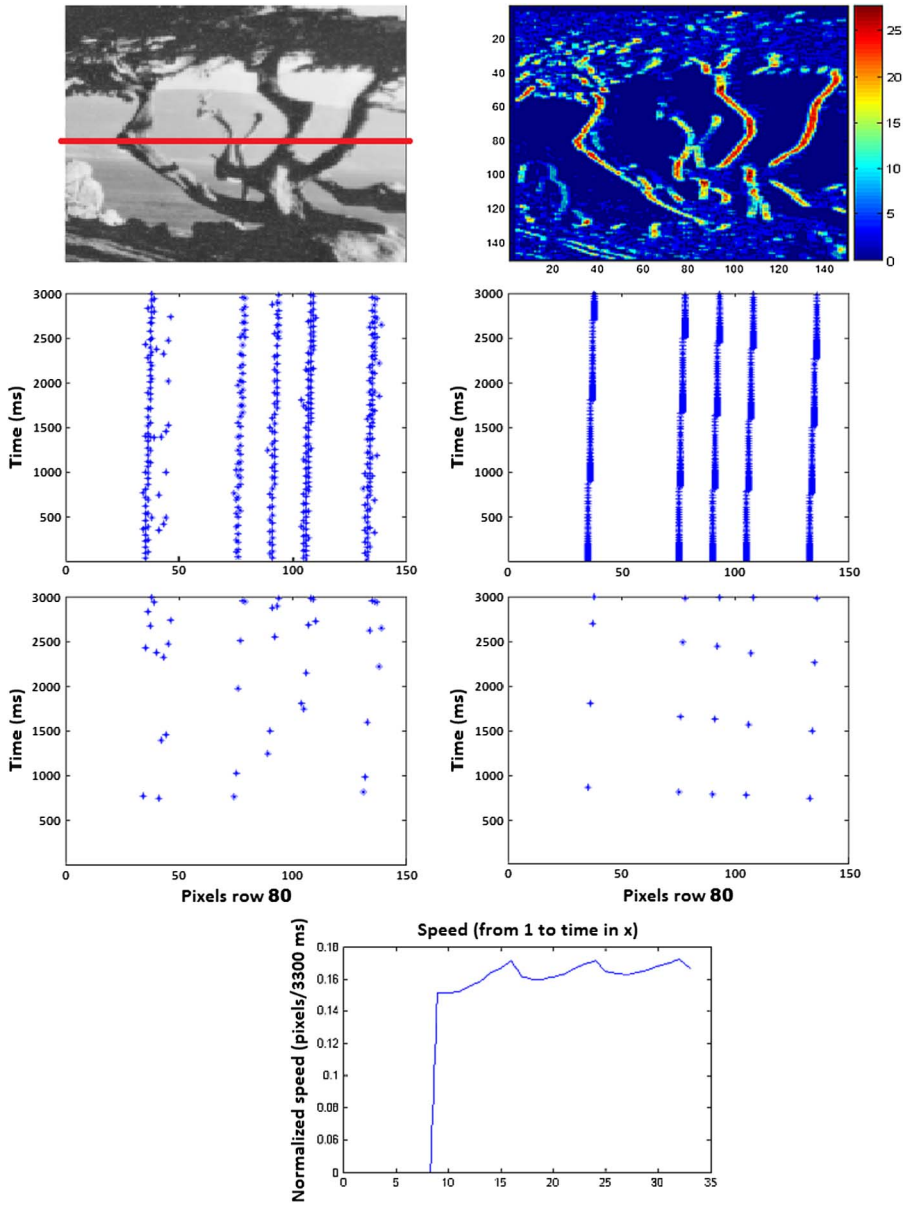

Fig. 5. ***Simulated events collected for row 80 of the "translation tree" sequence [38]. The first row shows the image and the accumulated positive events. The second and third rows show the positive events collected for row 80 of the image and the last event for every pixel. For the ideal situation with step edges (left), and for the real sequence (right). The last row shows the speed estimated for pixel (80, 40) for different time intervals (over 3300 time units at a step size of 100 time units).***

linear relation. Then, the number of events collected is proportional to the distance traversed times the contrast between the contour point and the traversed points. The contour has an intensity profile which is a ramp function (which can be narrow or broad). Assuming, Δt is sufficiently large, such that the whole contour traverses at least one pixel q, then the number of events collected at q is proportional to the contrast between the contour point and q. Thus, the displacement can be computed from the ratio of the two event counts (7). This formulation holds even for contours with an intensity profile stretching over multiple pixels, as illustrated in Fig. 6. Referring to the figure, the number of events in the motion contour is proportional to the (shaded) area enclosed by the intensity profiles at the start and end of the movement. This area is the same for the same horizontal displacements and does not depend on the slope of the edge; it is the area of the rectangle (or paralellogram) formed by the profiles of the step functions shown on the right. However, the computation becomes more accurate when the contours have a

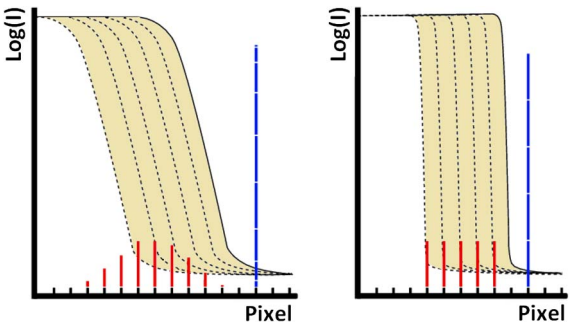

Fig. 6. *Intensity signals for time t and $t+\Delta t$ (solid and dashed lines, respectively). The red columns denote the number of events per pixel and the blue column denotes the sum of all these events along a broad edge (left) and step edge (right). The area of the region between the first and last curves is proportional to the number of events.*

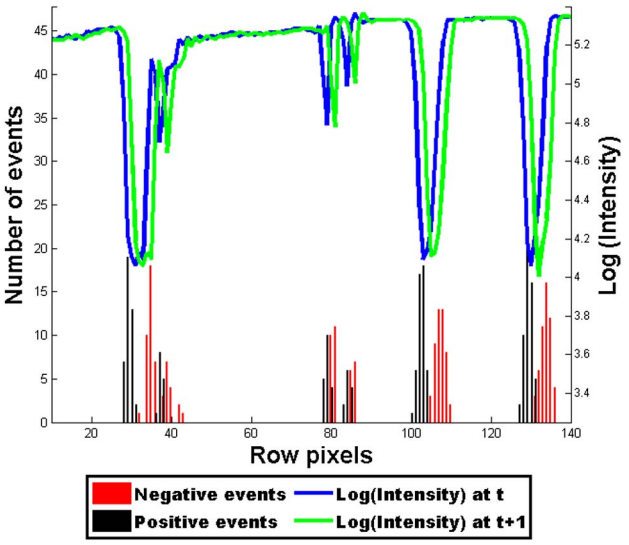

Fig. 7. *Intensity signals and simulated events for row 80 of the "translation tree" sequence [38]. The events were simulated for a time interval between two consecutive frames (denote t and $t+1$ in the figure). For real images, the intensity slightly varies because of background changes. Positive and negative events are handled separately until the final velocity computation.*

very steep ramp intensity profile, which usually is the case at discontinuities between objects in the scene and the background.

The edge contrast can be obtained from the pixel on the motion boundary with the maximum number of events. The displacement can then be computed as

$$D(p) = \text{sign}(u(p)) \frac{\sum_{q \in \mathcal{C}(p)} \mathcal{E}(q)}{\max_{q \in \mathcal{C}(p)} (\mathcal{E}(q))} \tag{7}$$

where $\text{sign}(u(p))$ is the sign of the displacement, $\sum_{q \in \mathcal{C}(p)} \mathcal{E}(q)$ is the number of all events in the motion contour, and $\max_{q \in \mathcal{C}(p)} (\mathcal{E}(q))$ is the number of events corresponding to the contrast. In practice, we do not compute the events of the contrast as the maximum number of events, but to robustify, we compute it as the median value from the five pixels with the largest amount of events (see Fig. 7).

The 1-D image velocity $u(p)$ then amounts to

$$u(p) = \frac{D(p)}{\Delta t}. \tag{8}$$

3) Temporal Multiresolution Velocity Estimation: So far we have only discussed the accumulation of events during a fixed time interval. However, in DVS event space, the information flow is continuous. We can make use of this fine-grained information in time. Instead of accumulating for a fixed time interval only, we can accumulate over multiple time intervals, and combine the information for a robust estimate.

Borrowing the concept of temporal integration from biology [39]–[41], we estimate velocities for different temporal resolutions. We implement a multiresolution method using different time intervals $\Delta t_i = [t_0, t_i]$, where t_0 and t_i are the timestamps of the first and last events in the motion boundary of the accumulated events. Let the set of time intervals be denoted $\{[t_0 t_1], [t_0, t_2], \ldots, [t_0, t_n]\}$ with $t_1 < t_2 < \cdots < t_n$, and let the velocity estimated from the events in time interval Δt_i be denoted $u_{\Delta t_i}(p)$. The final velocity estimate is derived as the median of these estimates, i.e.,

$$u(p) = \underset{i=1,\ldots,n}{\text{median}} (u_{\Delta t_i}(p)). \tag{9}$$

This estimate is given in pixels per second.

The last row of Fig. 5 depicts for a point in the "translation tree" sequence the image velocity estimates for different time intervals. In our implementation (described in Section V), we used $n = 6$ time scales.

4) Correction for Logarithmic Intensity: We have ignored that the DVS records logarithmic changes in the intensity. To obtain better subpixel accuracy displacement (and velocity), we provide a compensation. We will only do this for sharp edges, such as the contour edges analyzed in Fig. 5. For such edges, the fractional part of the displacement is due to the events in the last connected pixel of the boundary only. The distance traversed is linearly related to the intensity change, and we thus need to adjust events on the last pixel.

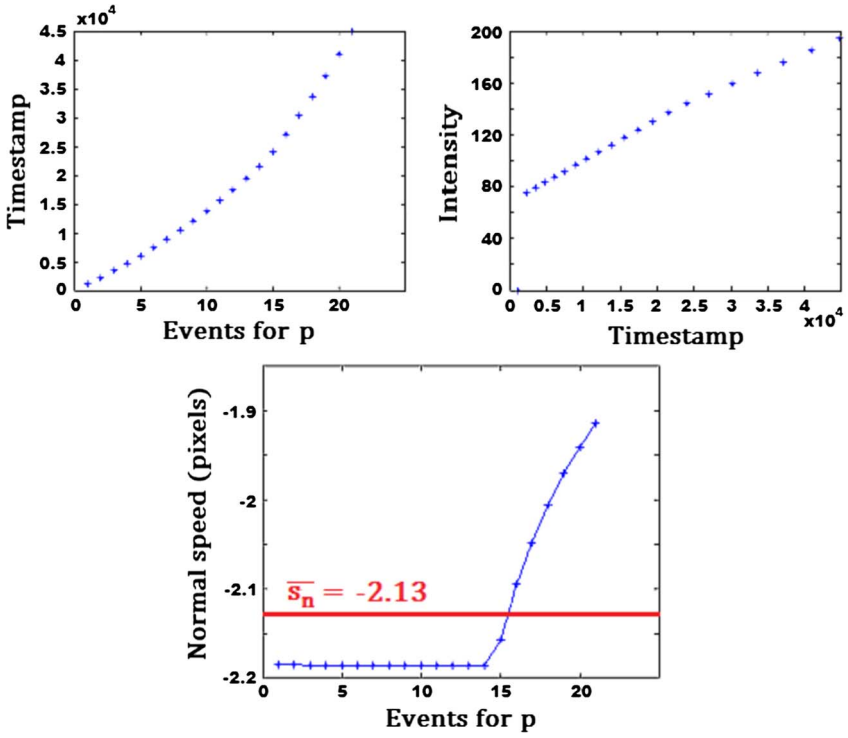

Fig. 8. *Image motion estimation for position p = 65,132 in the central frame of the translation tree sequence. First row: occurrences of events per time (left), relationship between events and intensity for position p (right). Second row: estimated value of the velocity. The red solid line indicates the estimated average value.*

The DVS records an event when the difference of the logarithms of intensities at time $t+1$ and time t reaches τ. From this, we obtain that the intensity I_n at the nth event is related to the original intensity I_o as $I_n = I_o e^{\tau n}$. Since the distance traversed is linearly related to the intensity change, we need to compute the number of reconstructed events $\mathcal{E}_{\text{rec}}$, which would be recorded if linear intensity changes were recorded. Using as before $\max(\mathcal{E}(q))$ to denote the number of events for the edge [corresponding to the maximum intensity change $(I_{\max} - I_o)$], the reconstructed number of events $\mathcal{E}_{\text{rec}}$ for the intensity I_e at the arrival of the $\mathcal{E}(e)$th event amounts to

$$\begin{aligned}\mathcal{E}_{\text{rec}} &= \frac{(I_e - I_o)}{I_{\max} - I_o}\max(\mathcal{E}(q)) \\ &= \frac{e^{\tau\mathcal{E}(e)} - 1}{e^{\tau\max(\mathcal{E}(q))} - 1}\max(\mathcal{E}(q)). \end{aligned} \tag{10}$$

Thus, the intensity adjusted estimation for the displacement amounts to

$$D(p) = \text{sign}(u(p))\left(\frac{\sum_{q\in\{\mathcal{C}(p)-e\}}\mathcal{E}(q)}{\max_{q\in\{\mathcal{C}(p)-e\}}(\mathcal{E}(q))} + \frac{e^{\tau\mathcal{E}(e)} - 1}{e^{\tau\max(\mathcal{E}(q))} - 1}\right). \tag{11}$$

C. Fusion of DVS and Intensity Signals

In the last few years, new bioinspired devices have appeared in the field, which combine asynchronous event-based and intensity data. These devices provide a framework similar to previously developed artificial retinas (see [18], [42], and [43]). Simulating two different kinds of ganglion cells, these sensors are a tool to study both the transient and sustained data pathways in mammalian vision; the transient through the asynchronous event-based information, and the sustained pathway through the intensity signal (although it is not provided in terms of spatial contrast or ON–OFF, OFF–ON responses, but in absolute terms). It is obvious that the intensity signal is essential for visual recognition, but as shown next, it also helps to improve the motion estimation on contours by adding some redundancy.

The main difference is that, with DVS data only, we have to reconstruct the contrast of the edge, and we can do this only from longer time intervals. Using the new sensors instead, we can compute the contrast directly by computing the spatial gradient of the data and thus perform our computations over short time intervals. There is no need to spend energy on localizing motion boundaries. Thus, the addition of the synchronous frame-based data makes our estimation faster and easier. The contrast is accurately computed from the intensity, and the events on the motion

boundary are computed as before, but now we can use narrow motion boundaries with as little as one pixel.

Given two neighboring pixels with intensity, I_p and I_{p+1}, their contrast will give rise to $\mathcal{E}_c$ events

$$\mathcal{E}_c = \log(I_{p+1}) - \log(I_p). \tag{12}$$

Thus, from (7) and (8), we obtain that

$$u(p) = \frac{\sum_{q \in \mathcal{C}(p)} \mathcal{E}(q)}{\mathcal{E}_c \Delta t}. \tag{13}$$

Again, we combine the estimates over multiple time intervals Δt_i using the median operation to make the approach more robust.

To obtain the normal flow (u_n, v_n) on the contour, we combine the estimates from the rows and columns. Measurements on pixels parallel to the rows provide u_n and measurements parallel to the columns provide v_n.

The method works well for contours. However, if used in highly textured regions, it will be less accurate, because it relies on the accuracy of the spatial gradient, which cannot be estimated well in these regions.

At contour points, where there is a depth discontinuity, we may not observe a single motion. Simply, accumulating events at the same position would lead to the same problems of frame-based methods. However, here, because we have many events, we can check for the continuity in image motion. In Fig. 8, we show an example of two motions on the same position for the translation tree sequence. The first and second graphics show the events per time and intensity (if available). The last one shows the value of the estimated normal velocity. The average estimate is -2.13 pixels while the ground truth is -2.186, with a relative endpoint error (EPE) of 2.5%. Although small, the error would increase as more events are accumulated.

We can find when to stop accumulating events. At a single point, assuming there is only one velocity, the intensity is linearly related to time. Since we have time and intensity, we just need to check if a newly added event fits the linear function defined by the previously collected events. This way, we obtain a stop condition and know how many events to accumulate.

IV. SOLUTIONS TO MOTION ESTIMATION PROBLEMS

As discussed in Section II, the state-of-the-art motion estimation methods of computer vision face major problems, which we try to solve with the new sensors. This section discusses the two main sources of error in frame-based methods: fast motions and occlusions (Section IV-A and B). Since frame-based methods are computationally intensive, performance is also a relevant issue to the community. Section IV-C shows some results of the real-time implementation of the method.

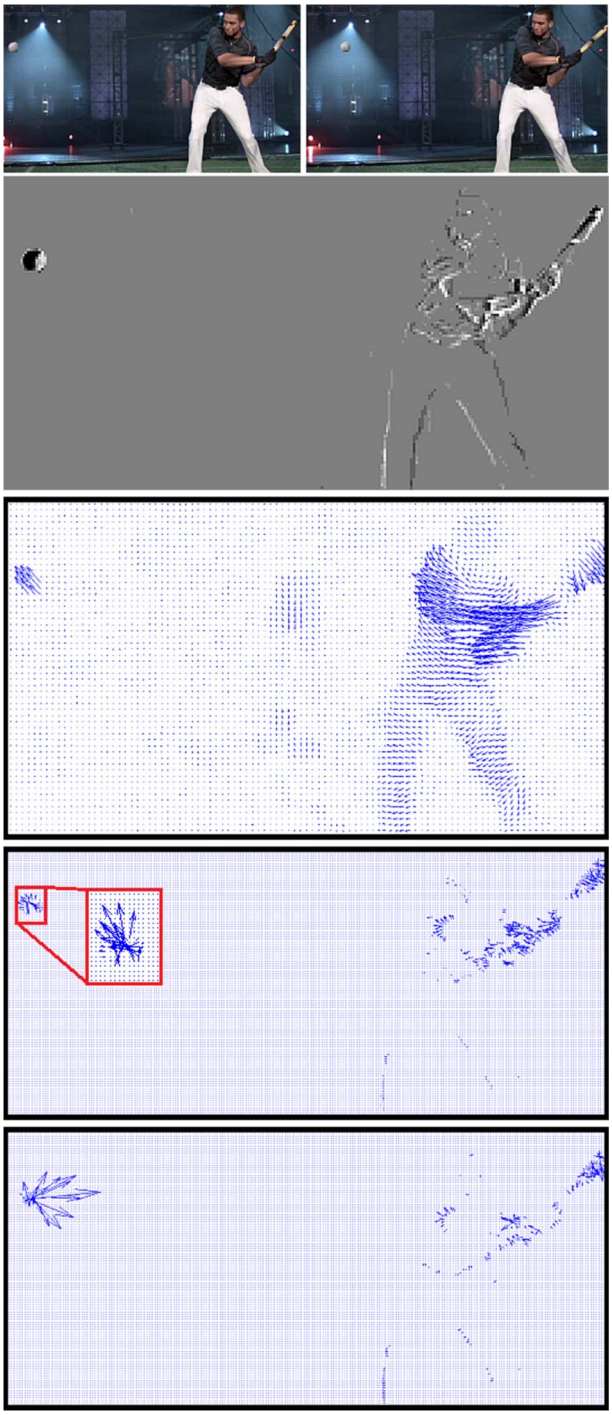

Fig. 9. *From top to bottom: two consecutive frames of the sequence, image of events, dense image motion estimation with classic+NL-fast [45] algorithm, contour motion estimation with classic+NL-fast, and contour motion event-based estimation. Note the wrong estimation of classic+NL-fast for the ball moving very fast, due to its size in a 1280 × 720 image.*

A. Fast Motions

Current methods include multiresolution schemes [44] using image pyramids enhanced with sophisticated

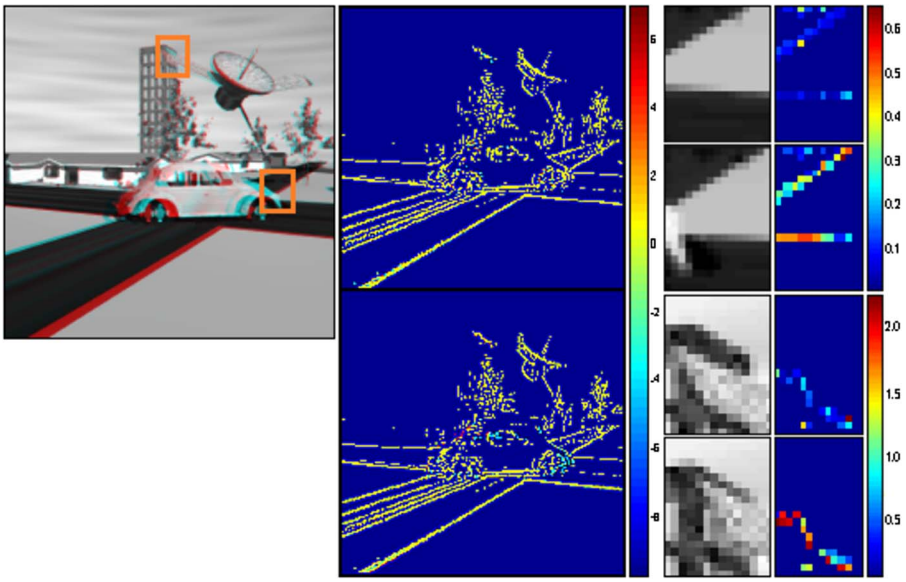

Fig. 10. ***Motion estimation for two consecutive frames of the satellite sequence [47]. Left: two frames superimposed in the blue and red channels. Center: Normal speed error for our event-based estimation and the classic+NL-fast algorithm. Right: Normal speed error for two occluded areas segmented from the original frame (see orange boxes).***

methods for filtering outliers to deal with large motions, and they require a large amount of computational resources. However, one of the problems still not resolved is how to obtain good estimates for small, fast moving objects, as pointed out in [45]. This kind of objects disappear at the lowest resolution, and when they pop out at higher resolutions, their fast motion makes them almost invisible.

Fig. 9 shows two frames from a video (available at "Sports Science: Tests the benefits of doctoring a baseball" http://www.youtube.com/watch?v=arlWwQ_smnM), depicting a baseball player that is going to hit a ball. If we compute the flow with one of the state of the art methods, e.g., *classic+NL-fast* from [45], whose source code is available, we find that the results fail to estimate the motion of the ball. The problem is that the multiresolution processing filters out the small fast moving ball at coarser levels. Due to the high temporal resolution of the DVS, there is no problem for our method. As we can see, we get lots of events and can estimate the velocity of the ball.

B. Occlusions

Current methods deal with the occlusion problem by trying to localize the occluded pixels to avoid the propagation of estimates to or from them. Mainly two strategies are used [46]: projection difference and flow divergence. In the first approach, every pixel of the second frame is projected into the first frame according to the estimated flow. Large differences between the projected frame and the actual frame indicate occlusion regions. In the second approach, a negative value of the divergence of the estimated flow is used as an indicator for occluded regions. However, in both approaches, the actual motion estimate is required. Since it is not available, it is computed iteratively (see [45]).

Fig. 10 illustrates the occlusion problem. The sequence shows a car moving to the left, a satellite moving up, and the camera is also rotating. In the figure, on the left, the original input frames are shown superimposed. We used RGB images to display the motion, with the R channel containing the first frame and the B channel containing the second. Thus, occluded regions are shown in red color, disoccluded regions in blue, and the rest is shown in gray. The middle images show the error with respect to the ground truth normal speed. The top image shows the normal speed error for the event-based estimation, and the bottom for the *classic+NL-fast* algorithm. For the latter, the error is not randomly distributed like in the event-based estimation. The highest errors are concentrated in the occluded and disoccluded regions, especially around the car, because the motion there is larger.

The right images show a zoom-in on two regions (orange boxes on the original). The colors represent the normal speed error (see the legend on the bar). We can see in detail here that the estimation for the conventional algorithm is wrong not only in the occluded area. Since a spatial smoothness term is used in the minimization, the errors are propagated from the occluded areas into the neighboring pixels as well.

TABLE 1 Relative Average Endpoint Error (AEPE in Percent) Results for Standard Optic Flow Methods and the Event-Based Method

Method	Trans.	Div.	Yosem.	Rubberwh.	Dimetr.
Event-based	9	33.6	18.6	27.3	23.6
Fleet-Jepson [51]	89.7	95.2	96.5	96.5	NA
Otte-Nagel [52]	17.8	21.9	24.2	35.4	NA
Uras [53]	31.5	31.7	60.3	43.2	48.6
Horn-Schuck [23]	52.6	28.8	37.8	56	72.4
Lucas-Kanade [33]	56.3	50.8	46.8	64.4	67.1
Sun [45]	0.9	13.9	7	15.4	3.0
Density (%)	5.43	0.40	0.46	1.22	1.41

Due to the high-temporal resolution of the DVS, the flow in these occlusion regions can be estimated well. Occlusions are not a problem for the DVS. However, as discussed, our method is geared for contours. In highly textured areas, where motion boundaries are not well defined, our event-based estimation is less accurate than conventional algorithms.

C. Real-Time Performance

The performance of image motion estimation with conventional methods is very low. State-of-the-art methods employ a combination of techniques, including sophisticated texture decomposition, multiresolution schemes, graduated nonconvexity weighing functions, spline-based bicubic interpolation, global smoothing terms, and nonlocal regularization terms. A processing time of a few minutes for the estimation of motion between two frames is very common. In addition, the possible strategies for further improvement seem to be exhausted. Some authors even argued that an early segmentation of motion boundaries is absolutely necessary to achieve further significant improvements [48].

The asynchronous nature of the DVS, the high-temporal resolution, and the ability of reconstructing the intensity with every new event allow us to present this work as a real-time accurate motion estimation. The simulated frame rate for the 200 × 200 satellite sequence using the method in Section III-C was found to be approximately 31.6 fps. The frames in this case correspond to the 15 000 events generated for a slice of 1 ms. The MATLAB implementation runs in a standard Intel i5 PC. Meanwhile, for the same sequence, using the same machine, the dense *classic+NL-fast* algorithm takes approximately 14.27 s to compute the flow between two images.

More importantly, the event-based motion estimation is scalable. For applications such as robotics, the real-time performance is crucial. The implementation of the event-based motion estimation is easy to parallelize in any massively parallel processing specialized hardware such as graphics processing units (GPUs), digital signal processors (DSPs), or field-programmable gate arrays (FPGAs). Contrary to conventional methods, here the normal speed for every position is updated with any new event and without any assumptions about the neighborhood.

V. EXPERIMENTAL RESULTS

This section has two parts: Section V-A uses simulated data and presents a comparison between classical computer vision methods and our event-based method and our combined event-based and intensity-based method; Section V-B uses real data to compare our algorithms to other event-based motion estimation implementations of the literature; approximate values of the ground truth are known.

A. Simulated DVS Data

To compare the accuracy of the speed estimation, we simulated the outputs of the DVS for the well-known "dimetrodon" and "rubberwhale" sequences from the Middlebury database [49] and the classic "Yosemite," "translation tree," and "diverging tree" sequences from [38]. "Dimetrodon" and "rubberwhale" are real sequences from a static camera and multiple moving objects with a strong component of texture in the background. "Yosemite" is an artificially created sequence simulating a fly through the

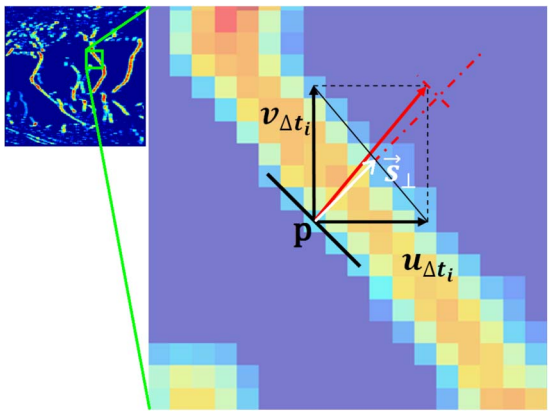

Fig. 11. *Normal speed estimation for a pixel in the "translation tree" sequence.*

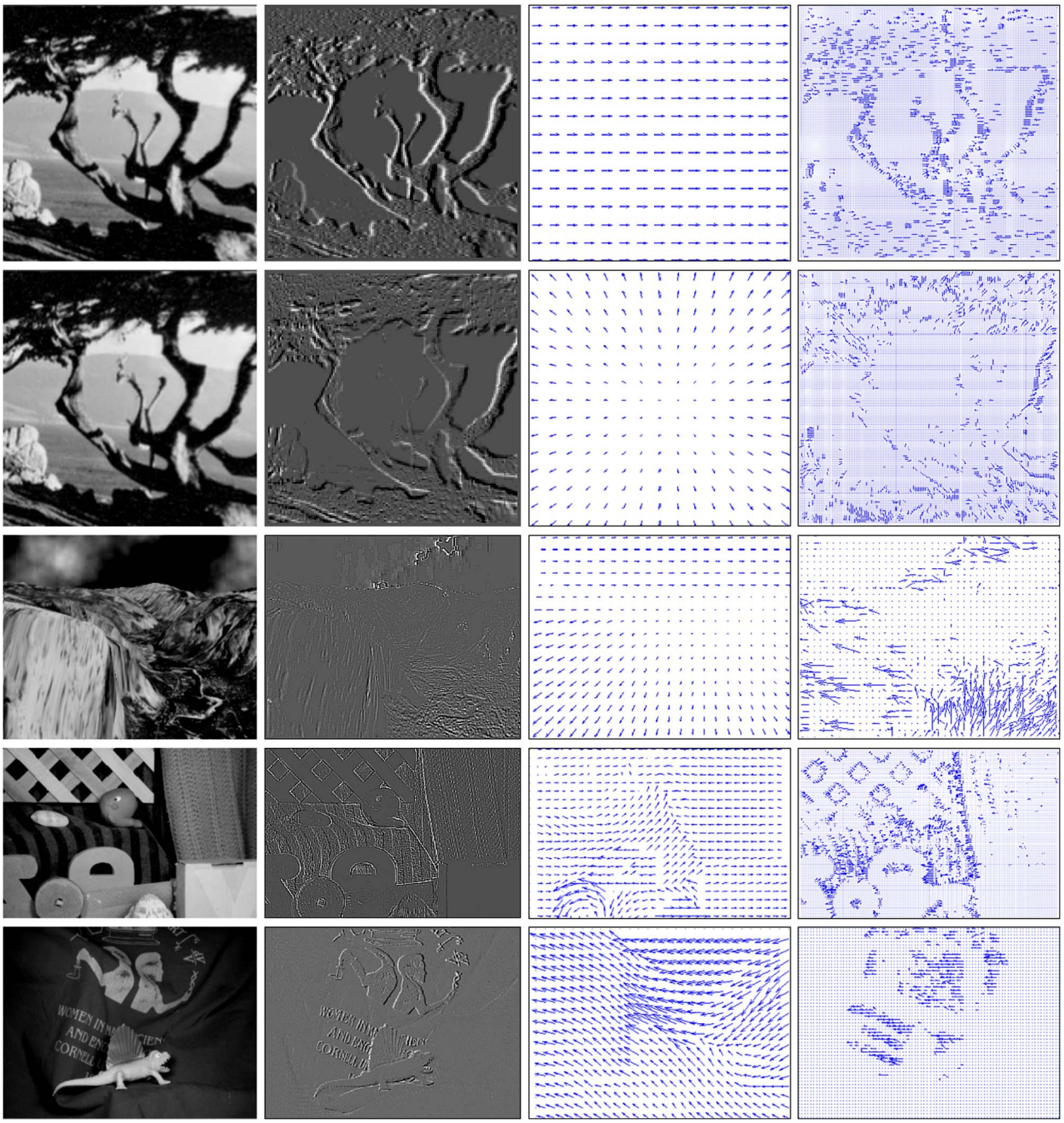

Fig. 12. ***Contour motion estimation. From left to right: original image of the sequence, events accumulated for frames 7 and 8 of the original sequences, ground truth optical flow field, and event-based contour motion.***

so-named valley, and the "tree" sequences are two artificial sequences generated from a real image of a tree, with the camera either translating parallel to or approaching the scene (see Fig. 12).

The reason for using simulated events is so that we can compare the accuracy of our methods to classic computer vision approaches and make use of the benchmarks. The simulated data were created as follows: the central frames for the mentioned sequences were selected. Using the ground truth motion (which is provided) the motions are computed for very small time intervals (e.g., 50 000 time units between two frames), and these motions are used to interpolate intermediate images. For the occluded regions, the ground truth is not available, but these regions are also

TABLE 2 Absolute and Relative Average Endpoint Errors (AEPE)

Sequence	AEPE abs	AEPE rel (%)	AEPE abs [45]	AEPE rel (%) [45]	Density (%)
Translation tree	**0.0000389**	**0.0029**	0.128	7.5	9.7244
Diverging tree	0.24	**15.4**	**0.146**	19.4	5.391
Yosemite	0.183	12.8	**0.14**	**11.7**	1.365
Rubberwhale	**0.28**	**25.2**	0.449	40.1	0.529
Dimetrodon	**0.142**	**7.2**	0.165	9.1	0.778

not considered in the benchmarks. Unfortunately, this last issue does not allow us to show one of the main advantages of our method.

We should note that our methods using DVS data are designed to estimate motion on contours. At this point, however, we do not use a method to distinguish between contour and high-contrast texture. Normal flow was computed at every edge with sufficiently high contrast.

Our measure of comparison is based on the (commonly used) relative average end point error (AEPE) in

$$E = \frac{1}{N}\sum_{i,j}\frac{\sqrt{uGT_{i,j}^2+vGT_{i,j}^2}}{\sqrt{(u_{i,j}-uGT_{i,j})^2+(v_{i,j}-vGT_{i,j})^2}}. \qquad (14)$$

In the above equation, (u, v) and (uGT, vGT) denote our estimation and the ground truth. (i, j) denotes the position, and N is the number of contour points. Our method computes normal motion, i.e., the projection of the image motion in the direction normal to the contour. For comparison, we projected the ground truth image velocities at the contours onto the normal direction computed by our event-based estimation. Table 1 shows the results of the comparison for different algorithms: the Sun *et al.* algorithm [45], and five classical optical flow methods available in [50]. The term "density" in this table denotes the percentage of image points for which we obtained flow values between 0.52% and 9.7%). The estimates were obtained as the average over 2000 repetitions.

Our method generally performs better than the classical algorithms, except for the diverging tree sequence, because the texture and the global motion in this sequence greatly benefit any motion estimation techniques using assumptions about smoothed global flow. The Sun *et al.* method [45] outperforms our method.

However, the problem with the evaluation is that since we simulate the motion, we cannot take full advantage of our proposed solution of locating the stopping point in time to find the time interval for which to accumulate events. The two-frame optical flow methods have an advantage in the evaluation, because in many sequences in the data sets the motion is globally smooth at most locations. Furthermore, there is no estimation available for the occlusions. Thus, there is no need to estimate in these areas, but only to stop the propagation of wrong estimates originating from the occluding areas. Finally, the sequences are not very complex. In all these sequences, the interframe distances are small, and they are free of blurring and image artifacts.

Fig. 11 illustrates the normal flow estimation. Once we have computed the horizontal and vertical velocity components u_n, v_n, the normal direction to the contour is computed as parallel to the vector u_n, v_n (denoted in the figure as $\widetilde{\mathbf{s}}_\perp$). The optical flow values computed by the various algorithms are projected on this direction. The results for the different sequences are shown in Fig. 12.

1) Combined Event-Based and Frame-Based Data: Table 2 compares the relative and absolute AEPE of our method using both events and intensity information (Section III-C) against the method of [45]. The results show that our method is more accurate in estimating contour motion, with the exception of the diverging tree sequence where there is a lot of texture in the tree. This is detrimental for the evaluation of our method but benefits [45], which because of a single global motion has accurate estimates in these regions due to the spatial smoothness constraints. Additionally, our method is slightly worse for the Yosemite sequence. In this case, it might be due to some well-known artifacts of this particular sequence and, again, the global motion of the sequence. However, note that we compare our method against one of the best available methods in the literature, ranked first until December 2013.

B. Real DVS/DAVIS Data

Fig. 13 shows examples of contour motion estimation on real data with the DAVIS, but there is no ground truth available for these scenes. The first row has a building sequence with the camera moving to the right. The second row has an indoor sequence with the camera static and a person. The bottom row contains a cluttered sequence with the camera moving in front of a bookshelf.

The next experiment demonstrates on a simple pattern that our algorithm can accurately estimate flow. Fig. 14 shows the estimated normal flow on the contours of a black star-shape pattern on white background, which in the first row moves to the right, and in the second to the left. The sequence was recorded with the DVS sensor and the normal flow on the contours was estimated with the first method (without using intensity information). Events were accumulated for 30 ms, but for clarity the vectors are shown at four times their value. Regarding the accuracy of our estimations, we found that the error for the first

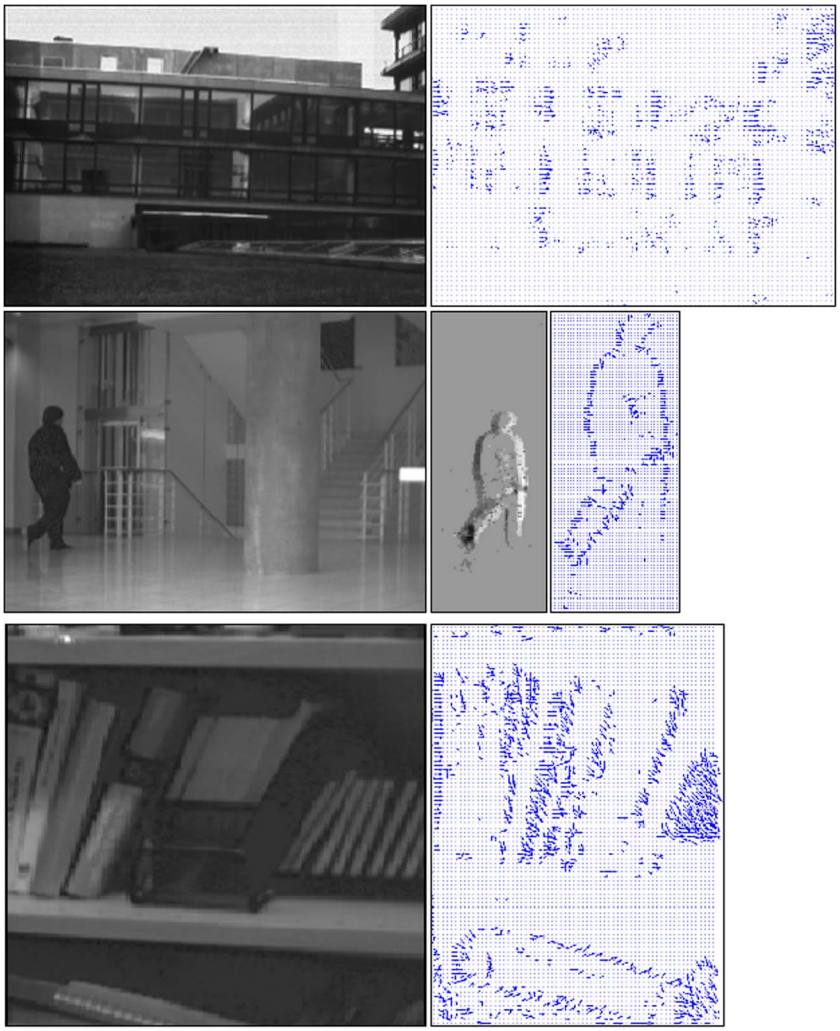

Fig. 13. ***Contour motion estimation for some real DAVIS sequences. First frame of the sequence (left) and contour motion estimation (right) results. In the second row, a zoom-in on the moving human is shown.***

sequence is 6.2% (0.29 pixel), and for the second it is 4.3% (0.15 pixel). The ground truth was found by manually tracking a number of well-defined points (such as corners) for 100 ms, and estimating the rigid motion of the star. Then, we estimated the normal flow by projecting the flow onto the gradient (which was known because we created the pattern).

Next, we present an indoor scene in which the camera is static and a person is moving to the left (see Fig. 15). In this case, the sequence was recorded with a DAVIS prototype, which provides asynchronous event data and synchronous intensity. The second image shows the normal flow on the contours of the person; nothing else is moving. The computation was performed for the events between two consecutive frames (since the frame rate was 17 fps, the time period was 58.8 ms). We estimated the flow by tracking points on the person and computed the mean flow of the walking person. For comparison, we projected this flow onto the direction of the gradients (extracted from the intensity signal). The evaluation of our method against the other methods in the literature gave the following results: the normal flow on the contours estimated with our first method (using only the events) has an error of 1.34 pixel. For the method in [32], the error is 1.8 pixel/s, and in the second method in [11], the error is 1.87 pixel/s. Finally, the normal flow estimated with our second method, which combines both signals (events and intensity), has an error of 0.79 pixel (approximately 15%).

Finally, Fig. 16 provides another comparison of more detail for a real sequence recorded with the DAVIS sensor. The camera is translating from left to right, and due to the difference in depth, the three main objects (boxes) move at different speeds (closer objects move faster). We computed the velocity from the correspondence of points in the different frames within the four highlighted blocks denoted as A–D. Specifically, between each pair of

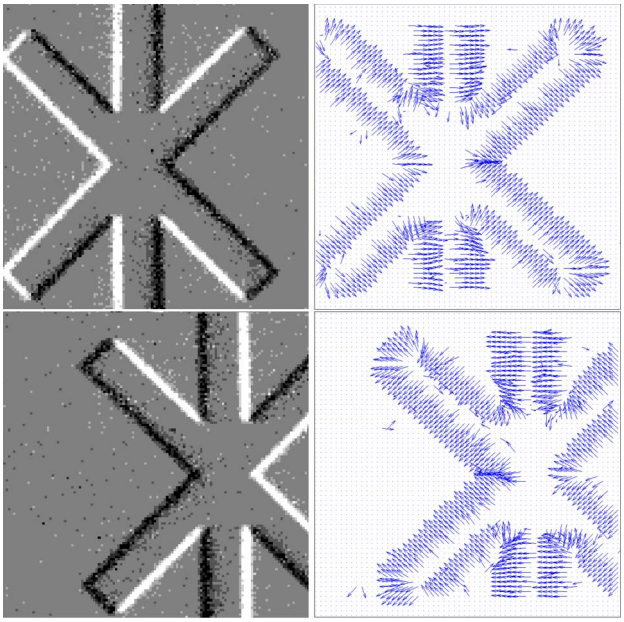

Fig. 14. *Contour motion estimation for two DVS sequences of a black star pattern in a white background. The accumulations of events are shown on the left, and the contour motion estimations using only events, on the right. In the first row, the pattern moves to the right and in the second one to the left.*

consecutive frames we computed the average speed for each block. Since the image sequence was recorded at 17 fps and there are six intensity frames, we have five chunks of DVS recordings in between the frames, each of 58.8 ms (in the figure, the five chunks are called fr1–fr5).

This second sequence was recorded with a new DAVIS prototype, and we experienced several problems. There are some bands of different intensity in the image causing artifacts at their border, and most important, the positive events were affected by a hardware problem that made them switch their polarity.

For the evaluation, we ran our implementations of four event-based motion estimation methods: the event-based contour motion proposed in this work, the event-based visual flow method proposed in [11], the event-based optical flow Lucas–Kanade method proposed in [32], and our method combining event and intensity signals. The methods of [11] and [32] were implemented following the description in the papers. For the algorithm in [11] we used the parameters described in the paper, except for $th2$, which was set to 0.15. For the implementation of (in [32]), we set Δt to 500 μs, and we used 5×5 neighborhoods for the least square estimation of optic flow.

Referring to the reported results in Fig. 16, the first value of the series represents the ground truth for the different regions (denoted as GT). The other values show the estimates for the frames in the five chunks of DVS data (fr1–fr5). On average, all methods behave reasonably well. Let us note that we removed outliers in all methods (with speeds higher than 170 pixel/s). Our method using only DVS data behaved the best, and is more stable than others with an average standard deviation of 3.09 pixel/s. Our method that combines intensity and events could not provide good estimates for blocks B and D, because, as mentioned before, there is a hardware problem with the DAVIS prototype that makes some of the positive events (both present in these two blocks) switch their polarity. However, the estimates for blocks A and C are very stable as well (the average standard deviation is 7.3 pixel/s). The main problem for the algorithm of [32] might be the smoothing process used to estimate the optical flow. For [11], the results are also good, but the error is higher than with our method, especially at ramp edges (most noted at blocks A and C, which have the most prominent ramp-profile edges). In both cases, the main challenges are the gray contrast and broadness of these edges.

The presented results show that our algorithms based on events can estimate well motion in natural scenes by making use of the high temporal resolution of the data in a

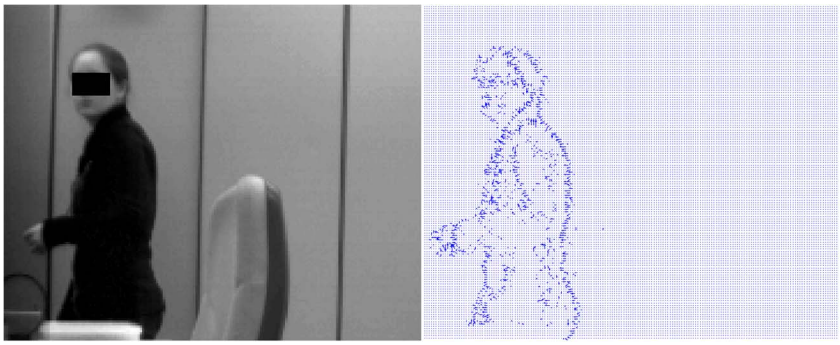

Fig. 15. *Contour motion estimation with DAVIS camera for an indoor sequence. The camera is static and a person is moving to the left in the scene. The first image shows the intensity signal and the second one the normal flow on the contours.*

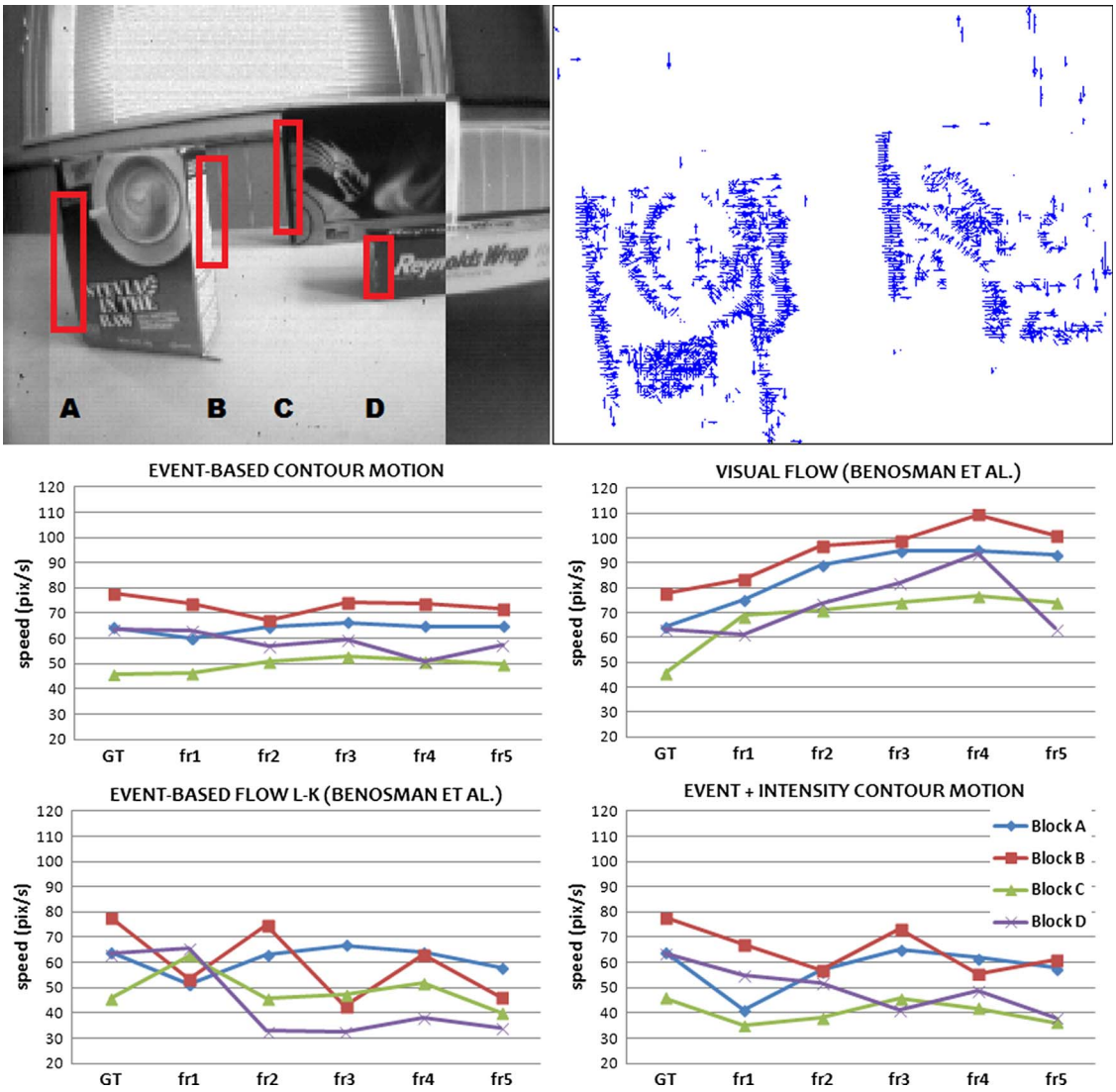

Fig. 16. *Motion evaluation for real DAVIS sequence for four methods. The first row shows the first frame of the sequence recorded with the DAVIS sensor (left) and the normal flow estimated with our event-based method (right). The four overlaid rectangles denote the regions for which we evaluated the estimates. The second and third rows show the estimated speed within the different blocks, for our event-based method, the visual flow method [11], the event-based optical flow Lucas–Kanade method [32], and our method, which combines event and intensity data.*

bioinspired way. As can be seen qualitatively, the methods give correct estimates not only for step edges, but also for broader edges stretching along multiple pixels. Additionally, we have shown that the methods are efficient and require little resources, thus providing the basis for a new framework in computer vision utilizing bioinspired event-based hardware.

VI. CONCLUSION

We have presented algorithms for estimating image motion on contours from asynchronous event-based information. Conventional image motion estimation cannot handle depth discontinuities well, while our methods provide accurate estimates on contours, thus solving the problem of conventional approaches.

The first technique presented, which uses as input DVS data, reconstructs the contrast at contours from the events and follows the contour points over time. The accuracy of the technique is due to the use of exact timestamps, and a temporal multiresolution integration.

The second method uses a combination of event-based motion information and intensity information. We showed that using both sources of data brings great advantages. The method, in essence, computes contrast from the intensity, and temporal derivatives from the event-based

motion information, and it also combines measurement over multiple time intervals for robustness. The method is very accurate and can be realized in real time.

At this point, the work presented is still largely conceptual. The DAVIS sensor is still under development, and benchmarks with ground truth for DVS data have not been created yet. As sensors become available and better in performance, the community will need to create new benchmarks with ground truth for both neuromorphic and regular cameras, and we will then be able to tune our methods to the specific hardware constraints.

Regarding further software development, so far, we have not developed a reliable way of distinguishing between contours at boarder and texture. As discussed, the stopping condition will play a major role in distinguishing. In addition, we plan to study constraints that exploit the different spatial distributions of the events. Further work then involves studying how contour motion can be used for other early vision processes to address the whole early motion pathway.

Finally, we argue that rather than considering event-based computations in isolation and as an alternative to conventional motion processing, they could be looked at as a way to overcome the major issues that conventional motion estimation methods of computer vision are trying to solve. Since the conventional strategies are becoming exhausted, this new idea of integrating synchronous and asynchronous, frame-based and event-based mechanisms could be a viable alternative. As shown here, through the integration, we can achieve a great reduction in the use of computational resources and an increase in performance. ■

REFERENCES

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128× 128 at 120 dB 15 micros latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[2] G. M. Shepherd, Ed., *The Synaptic Organization of the Brain*. Oxford, U.K.: Oxford Univ. Press, 1990.

[3] B. Roska, A. Molnar, and F. S. Werblin, "Parallel processing in retinal ganglion cells: How integration of space-time patterns of excitation and inhibition form the spiking output," *J. Neurophysiol.*, vol. 95, no. 6, pp. 3810–3822, 2006.

[4] B. Roska and F. Werblin, "Rapid global shifts in natural scenes block spiking in specific ganglion cell types," *Nature Neurosci.*, vol. 6, no. 6, pp. 600–608, 2003.

[5] S. Zeki, *Vision of the Brain*. New York, NY, USA: Wiley, 1993.

[6] L. Líden and C. Pack, "The role of terminators and occlusion cues in motion integration and segmentation: A neural network model," *Vis. Res.*, vol. 19, no. 39, pp. 3301–3320, 1999.

[7] J. Chey, S. Grossberg, and E. Mingolla, "Neural dynamics of motion grouping: From aperture ambiguity to object speed and direction," *J. Opt. Soc. Amer. A*, vol. 10, no. 14, pp. 2570–2594, 1997.

[8] C. Fermüller, R. Pless, and Y. Aloimonos, "Families of stationary patterns producing illusory movement: Insights into the visual system," *Proc. Roy. Soc. Lond. B*, no. 264, pp. 795–806, 1997.

[9] C. Fermüller, T. Brodsky, and Y. Aloimonos, "Motion segmentation: A synergistic approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1999, no. 2, DOI: 10.1109/CVPR.1999.784633.

[10] R. Benosman, S.-H. Ieng, P. Rogister, and C. Posch, "Asynchronous event-based Hebbian epipolar geometry," *IEEE Trans. Neural Netw.*, vol. 22, no. 11, pp. 1723–1734, Nov. 2011.

[11] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 407–417, Feb. 2014.

[12] G. Orchard, R. Benosman, R. Etienne-Cummings, and N. Thakor, "A spiking neural network architecture for visual motion estimation," in *Proc. IEEE Biomed. Circuits Syst. Conf.*, 2013, pp. 298–301.

[13] Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Regnier, "Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1081–1089, Oct. 2012.

[14] J. H. Lee *et al.*, "Touchless hand gesture UI with instantaneous responses," in *Proc. 19th IEEE Int. Conf. Image Process.*, Sep. 2012, pp. 1957–1960.

[15] D. Drazen, P. Lichtsteiner, P. Häfliger, T. Delbrück, and A. Jensen, "Toward real-time particle tracking using an event-based dynamic vision sensor," *Exp. Fluids*, vol. 51, no. 5, pp. 1465–1469, 2011.

[16] J. Carneiro, S.-H. Ieng, C. Posch, and R. Benosman, "Event-based 3D reconstruction from neuromorphic retinas," *Neural Netw.*, vol. 45, pp. 27–38, 2013.

[17] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Front. Neurosci.*, vol. 7, no. 178, 2013, DOI: 10.3389/fnins.2013.00178.

[18] J. Costas-Santos, T. Serrano-Gotarredona, R. Serrano-Gotarredona, and B. Linares-Barranco, "A spatial contrast retina with on-chip calibration for neuromorphic spike-based AER vision systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 7, pp. 1444–1458, Jul. 2007.

[19] M. Azadmehr, J. P. Abrahamsen, and P. Hafliger, "A foveated AER imager chip [address event representation]," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2005, no. 3, pp. 2751–2754.

[20] U. Mallik, M. Clapp, E. Choi, G. Cauwenberghs, and R. Etienne-Cummings, "Temporal change threshold detection imager," in *Dig. Tech. Papers IEEE Int. Solid-State Circuits Conf.*, 2005, no. 1, pp. 362–603.

[21] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 db dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, Jan. 2011.

[22] R. Berner, C. Brandli, M. Yang, S.-C. Liu, and T. Delbruck, "A 240 × 180 10 mW 12 us latency sparse-output vision sensor for mobile applications," in *Proc. Symp. VLSI Circuits*, Jun. 2013, pp. C186–C187.

[23] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185–203, 1981.

[24] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime TV-L^1 optical flow," in *Proc. 29th DAGM Conf. Pattern Recognit.*, 2007, pp. 214–223.

[25] P. Krähenbühl and V. Koltun, "Efficient nonlocal regularization for optical flow," *ECCV European Conference on Computer Vision*, vol. 7572, A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Germany: Springer-Verlag, 2012, pp. 356–369.

[26] C. Fermüller, "Passive navigation as a pattern recognition problem," *Int. J. Comput. Vis.*, vol. 14, no. 2, pp. 147–158, 1995.

[27] T. Brodsky, C. Fermüller, and Y. Aloimonos, "Structure from motion: Beyond the epipolar constraint," *Int. J. Comput. Vis.*, vol. 37, no. 3, pp. 231–258, 2000.

[28] J. Hui and C. Fermüller, "A 3D shape constraint on video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 1018–1023, Jun. 2006.

[29] A. S. Ogale, C. Fermüller, and Y. Aloimonos, "Passive navigation as a pattern recognition problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 988–992, Jun. 2005.

[30] T. Delbruck, "Frame-free dynamic digital vision," in *Proc. Int. Symp. Secure-Life Electron.*, Tokyo, Japan, Mar. 2008, pp. 21–26.

[31] E. Kandel and R. Wurtz, "Constructing the visual image," in *Principles of Neural Science 4*, N. Y. M. Hill, R. Kandel, J. H. Schwartz, and T. M. Jessel, Eds. New York, NY, USA: McGraw-Hill, 2002, pp. 492–506.

[32] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow," *Neural Netw.*, vol. 27, pp. 32–37, Mar. 2012.

[33] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Conf. Artif. Intell.*, 1981, vol. 2, pp. 674–679.

[34] B. Jähne, "Spatio-temporal image processing," in *Theory and Scientific Applications*, vol. 751, Berlin, Germany: Springer-Verlag, 1993.

[35] M. Edwards, "Motion streaks improve motion detection," *Vis. Res.*, vol. 47, no. 6, pp. 828–833, 2007.

[36] W. S. Geisler, "Motion streaks provide a spatial code for motion direction," *Nature*, vol. 400, no. 6739, pp. 65–69, 1999.

[37] T. Delbruck, "jAER open source project," 2014. [Online]. Available: http://sourceforge.net/p/jaer

[38] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *Int J. Comput. Vis.*, vol. 12, pp. 43–77, 1994.

[39] W. Bialek, F. Rieke, R. de Ruyter van Steveninck, and D. Warland, "Reading a neural code," *Science*, vol. 252, no. 5014, pp. 1854–1857, 1991.

[40] F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek, *Spikes: Exploring the Neural Code*. Cambridge, MA, USA: MIT Press, 1999.

[41] E. J. Chichilnisky and R. S. Kalmar, "Temporal resolution of ensemble visual motion signals in primate retina," *J. Neurosci.*, vol. 23, no. 17, pp. 6681–6689, 2003.

[42] K. Boahen, "A retinomorphic chip with parallel pathways: Encoding increasing, on, decreasing, and off visual signals," *Analog Integr. Circuits Signal Process.*, vol. 30, no. 2, pp. 121–135, 2002.

[43] S. Kameda and T. Yagi, "An analog VLSI chip emulating sustained and transient response channels of the vertebrate retina," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1405–1412, Sep. 2003.

[44] P. Burt, *The Pyramid as a Structure for Efficient Computation*, A. Rozenfeld, Ed. New York, NY, USA: Springer-Verlag, 1984.

[45] D. Sun, S. Roth, and M. Black, "A quantitative analysis of current practices in optical flow estimation and the principles behind them," *Int. J. Comput. Vis.*, vol. 106, no. 2, pp. 115–137, 2014.

[46] P. Sand and S. Teller, "Particle video: Long-range motion estimation using point trajectories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2006, vol. 2, pp. 2195–2202.

[47] B. McCane, K. Novins, D. Crannitch, and B. Galvin, "On benchmarking optical flow," *Comput. Vis. Image Understand.*, vol. 84, no. 1, pp. 126–143, 2001.

[48] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 2432–2439.

[49] S. Baker *et al.*, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, Mar. 2011.

[50] F. Raudies, "Optic flow," *Scholarpedia*, vol. 8, no. 7, p. 30724, 2013, rev. 134420.

ABOUT THE AUTHORS

Francisco Barranco received the M.Sc. degree in computer and network engineering and the Ph.D. degree in computer science from the University of Granada, Granada, Spain, in 2008 and 2012, respectively.

He is a European Union (EU) Marie Curie Fellow at the University of Maryland, Collge Park, MD, USA, and works for the Computer Science Department. His main research interests concern robotics, embedded real-time machine vision, bioinspired processing, and cognitive vision. He has participated in different EU projects for adaptive learning and real-time computer vision.

Cornelia Fermüller received the M.S. degree from the Graz University of Technology, Graz, Austria, in 1989 and the Ph.D. degree from Vienna University of Technology, Vienna, Austria, in 1993, both in applied mathematics.

She is a Research Scientist at the University of Maryland Institute for Advanced Computer Studies (UMIACS), University of Maryland, College Park, MD, USA. Her research interest has been to understand principles of active vision systems and develop bioinspired methods, especially in the area of motion. Her recent work has been centered on developing robot vision for interpreting human manipulation actions.

Yiannis Aloimonos received the Ph.D. degree in computer science from the University of Rochester, Rochester, NY, USA, in 1987.

He is a Professor of Computational Vision and Intelligence at the Department of Computer Science, University of Maryland, College Park, MD, USA and the Director of the Computer Vision Laboratory at the Institute for Advanced Computer Studies (UMIACS). He is interested in the integration of perception, action, and cognition (bridging signals and symbols).

Dr. Aloimonos received the Presidential Young Investigator Award (PECASE) for his work on active vision.