Bio-inspired motion estimation with event-driven sensors

Francisco Barranco[1,2], Cornelia Fermuller[1], and Yiannis Aloimonos[1]

[1]University of Maryland,
UMIACS, CFAR, A.V. Williams Bldg, College Park (MD), USA
[2]University of Granada
CITIC, ETSIIT, Daniel Saucedo Aranda sn, Granada, Spain
{fbarranco,cfer,yiannis}@umiacs.umd.edu

Abstract. This paper presents a method for image motion estimation for event-based sensors. Accurate and fast image flow estimation still challenges Computer Vision. A new paradigm based on asynchronous event-based data provides an interesting alternative and has shown to provide good estimation at high contrast contours by estimating motion based on very accurate timing. However, these techniques still fail in regions of high-frequency texture. This work presents a simple method for locating those regions, and a novel phase-based method for event sensors that estimates more accurately these regions. Finally, we evaluate and compare our results with other state-of-the-art techniques.

Keywords: Bio-inspired systems; Neuromorphic engineering; Motion estimation; Event-driven sensors; Asynchronous sensors

1 Introduction

Biological strategies refined by evolution, have been emulated in various fields and been applied to a wide range of application [18]. We are interested in the adaption of biological vision neural systems to computational principles for the estimation of visual features. Current cameras are operating synchronously, acquiring image frames at a fixed sampling frequency, but capture visual information in a continuous world that works asynchronously. This quantization limits applications requiring high-speed maneuvering in autonomous navigation and robotics. When there are no changes, the same visual information is recorded in different frames. This translates into a demand for plenty resources and memory in order to process redundant data.

In the last few years frame-free sensors have become increasingly popular due their accurate time triggering, low-latency, real-time performance, and low-resource requirements. Emulating the human neural vision system, these sensors are driven by events. There are two main retina-brain pathways in the human early vision system: the sustained pathway, that is believed to conduct information such as color, texture, or intensity patterns, and the transient pathway, that only responds to changes. Sensors have focused on the first one while frame-free sensors emulate the second. In our case, the DVS (Dynamic Vision Sensor [15]) triggers new events when the reflectance for a specific location changes. In case of no change, no information is generated. Instead of an external clock that defines the sampling frequency, the control is transferred to the individual pixels that handles it independently.

From the Computer Vision point of view, the efforts in motion estimation have been devoted to improving accuracy. The main challenges are due to the difficulty in estimating at object boundaries, where there are discontinuities in the flow field. Here is where event-based computation represents an alternative, since motion estimation is very accurate at boundaries even when the contours of the objects are not provided in advance. This is because of the high working frequency of these sensors that allow to obtain image motion by observing events between neighboring pixels. Thus one can obtain image motion only from edges without crossing different moving regions.

Current motion estimation techniques for event sensors provide accurate results, even allowing us to track pixels between neighboring positions (especially in [6, 3]). Their underlying assumption is that events that are fired during a short period of time at close-by positions are due to changes in the same edge. Consequently, these methods work accurately for strong contrast edges. However, high-frequency textures violate the assumption. At such locations, frequency-based approaches can increase the accuracy. This paper presents an event-driven method based on local phase (in the frequency domain). Moreover, we also describe a method to localize the contours of textured objects. Our evaluation shows that the method achieves more accurate estimates for highly textured areas than previous approaches.

The paper is structured as follows: section §2 describes the sensor and Section §3 motivates the motion estimation. Section §4 describes the contour localization and Section §5 the new phase-based technique for frame-free sensors. Finally, Section §6 presents results and Section §7 gives the conclusions.

2 Asynchronous frame-free sensors

For a conventional camera, a sequence of $M \times N$ frames, are taken at a given sampling rate f_s, such that the value of each pixel (x, y) with $x \in [0, M-1]$ and $y \in [0, N-1]$ accounts for the intensity recorded during a time $\Delta t_s = f_s^{-1}$. The Dynamic Visual Sensor (DVS [15]) fires asynchronous address-events that signal reflectance changes at the time they occur. Its spatial resolution is 128×128 and its maximum temporal resolution is 15 μs. The DVS does not use a fixed sampling rate. Instead, its samples the input every time the variation with respect to the last measured value at the same position (x, y) exceeds a fixed amount. In other words, for an image point (x, y), at time t an event $ev(x, y, t, p)$ is fired, where p is the polarity $+1$ or -1, if the log of the intensity I increases or decreases by a global threshold ΔI_s (see Eq. 1). For example, if as in our case, the threshold is 0.1, it means that the intensity has to change by 10% to fire an event. The logarithmic input helps dealing with high dynamic ranges. Fig. 1

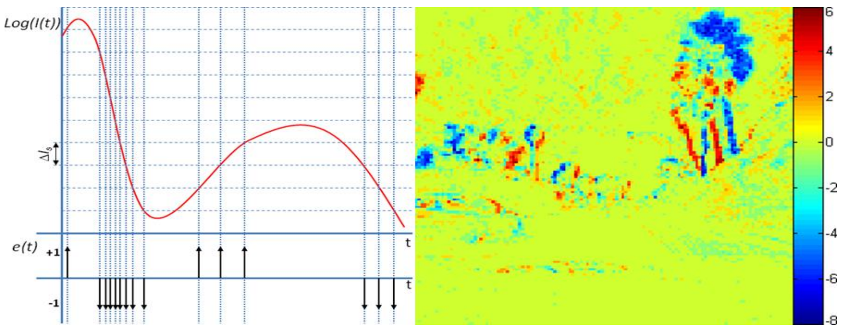


Fig. 1. Left: Top part shows the log Intensity and bottom the events fired when the input changes by ΔI_s. Events are + or - according to the sign of the change. Right: Image to visualize the events accumulated during 50 ms: warm colors represents positive counts, cold colors are negative counts and green, the lack of events.

illustrates the workings of the sensor.

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

Asynchronous frame-free sensors help us address some of the challenges of motion processing arising in classical Computer Vision. First, since the DVS only transmits data where and when changes happen, it helps saving a lot of resources. Second, since the events are fired at a very high frequency, with a temporal resolution of a few microseconds, there is no blur due to fast motion. Moreover, because the input is logarithmic, the sensor can deal with high dynamic range.

3 Motion estimation for asynchronous frame-free sensors

Conventional motion estimation techniques work well in smooth textured areas, but they have problems at object contours [21]. The greatest challenge are fast motions and particularly scenes with both: large and small motions. On the other hand, data from frame-free sensors is spatially highly localized, and because of the high temporal resolution, potentially allows to locate edges without crossing different moving regions leading to very accurate boundary motion. Furthermore, fast motions are not a problem for these sensors. Because of these advantages, frame-based motion processing has great potential for computational motion processing, either alone or combined with classic processing [3].

In event-based space, only normal flow can be estimated directly. Normal flow is the projection of the flow in the direction of the spatial gradient, and it is sufficient for computing 3D motion as shown in [11, 10].

Authors in [7] adapt the Lucas-Kanade method [16]. The Optical Flow Constraint is re-written assimilating the number of events accumulated during a time interval Δt into intensity. Denoting as $e(\mathbf{x}, t)$ the event that happens at position

$\mathbf{x}$ and time t, the derivatives of e are approximated as in Eq. 2, where $t_1 < t$.

$$\nabla e(\mathbf{x},t) \approx \sum_{t-\Delta t}^{t} e(\mathbf{x},t) - \sum_{t-\Delta t}^{t} e(\mathbf{x}-(1,1),t), \qquad e_t(\mathbf{x},t) \approx \frac{\sum_{t1}^{t} e(\mathbf{x},t)}{t-t_1} \tag{2}$$

In [6], the authors define a function $\mathcal{T}_e(\mathbf{x})$ that assigns only the time of every new event fired at position $\mathbf{x}$. Then, the gradient vector of $\mathcal{T}_e$ provides the inverse of the local velocity vector as in $\nabla\mathcal{T}_e = (1/\mathbf{v})^T$. The velocity is estimated as the inverse of the horizontal and vertical derivative of this plane. Although accurate, at broad edges different close-by pixels can fire events at almost the same time, making the estimation not reliable.

Finally, [3] addresses the issue of broad edges, and presents an alternative that reconstructs the contrast at object edges and tracks them. The method first locates for a certain time interval all the events of the moving edge, the so-called motion boundary. Then, the velocity is estimated as the ratio of all events in the motion boundary over the events at a single point, normalized for time. Accumulating events over multiple time intervals and averaging provides robustness. This work is also the first that combines asynchronous events and synchronous frames with the new DAVIS sensor [8].

However, all these methods have problems with textures. First, at object contours moving on top of textured regions, the contrast between contour and background changes over a small time interval, causing different amount of events. Second, highly-textured areas introduce too much variability for fitting a plane, or for reconstructing the intensity. Frequency-based methods, adapted to asynchronous event space, can help with this problem.

4 How texture affects event-based frameworks

As mentioned, current techniques assume that nearby events, which are fired close-by in time, belong to the same structure. However, in highly textured areas or at broader edges next to textured background, the assumption is violated.

First, let us distinguish between edges and contours: *edges* are all the strong discontinuities in the intensity signal, while *contours* are only those edges that correspond to the boundaries of objects (depth discontinuities).

Next, we examine the different cases imposed by movement and texture. Case 1: Objects moving on static non-textured background. If the object is textured, events from contours and texture edges are similar. Event-based methods work well at contours, but they may not work well in a textured region, if it has high-frequency, and the method accumulates events over longer time intervals. If the object is plain, only at the boundary of the object events are fired, where classic event-based methods work well. Case 2: Moving objects on static textured background. Since the background is textured, the patterns of events on the contours over time are changing. Such situations pose a problem for event based methods. Case 3: Both the object and the textured background are moving. In this case, two different physical motions take place in regions next to object boundaries. This is the most difficult case for image motion estimation.

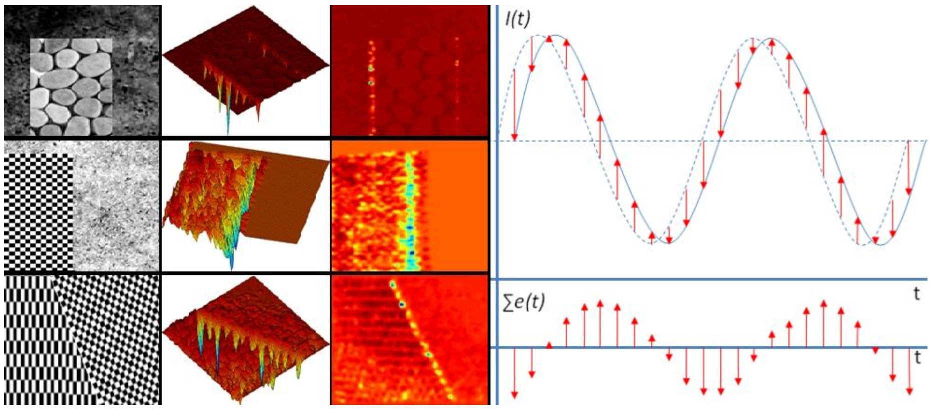


Fig. 2. Left: Contour localization using cross-correlation (left to right): original frame, 3D and 2D representation of the cross-correlation coefficient. Right: A sinusoid moving to the left shown as dashed line at time t and as solid line at $t+\Delta t$. The recorded events over a small time interval make a sinusoid function of same frequency. Red arrows are proportional in size to the amount of positive or negative events at the location.

4.1 Locating contours of textured objects

Contours can be separated from other edges if we consider the patterns of events at single pixels. The event pattern at texture edges should be shifted (since there is the same contrast between neighboring pixels over time.) But for contours the pattern changes due to a background texture. Considering separately every pixel, the timestamps of the events can be looked at as a 1D signal. Normalized cross-correlation measures the similarity of 1D signals, and it is widely used in signal processing [19]. Our idea is that if there is a texture, events from neighboring positions would have very high cross-correlation coefficients while in the case of broad edges or contours on top of textured backgrounds, the crosscorrelation is low. Let us assume we have the $1 \times N$ signal $f(t)$ and $1 \times M$ signal $g(t)$, Eq. 3 defines the normalized cross-correlation

$$ncc(d) = \frac{|\sum_{t=1}^{t=N} (f(t) - \overline{f})^2 (g(t-d) - \overline{g})^2|}{||(f(t) - \overline{f})^2||_2 ||(g(t-d) - \overline{g})^2||_2} \tag{3}$$

Where $d \in [0, M-1]$ is the lag or delay between the signals (in case they are shifted), and the normalized cross-correlation is an $M \times N$ vector. The maximum gives us the coefficient for measuring the similarity. As mentioned, due to the high-temporal resolution of the sensor, changes are tracked pixel by pixel. Then, the cross-correlation is performed for the 1D signals (composed by the series of timestamps of the events fired) of the neighboring pixels in a region of only 3×3. Next, to increase the robustness, more events are required since noise and distortions may limit the accuracy. For signal correlation, low-pass filters reduce signal noise. However, asynchronous frame-free sensors fire discrete events. Thus, instead of low-pass filters we use activity filters to reduce outliers and noise.

Events do not occur isolated in time and/or space, so events that are not very close spatio-temporally to other groups of events are discarded (we use 5 ms intervals and 3×3 windows). Moreover, robustness increases with the number of samples. Assuming longer time intervals increases the latency and reduces performance. In our work, we propose using larger spatial windows (3×3) and compact all their events into a single 1D signal.

The left part of Fig. 2 shows the cross-correlation coefficient. The first row shows a synthetic sequence with a texture moving on top of a textured background (case 1). The first image shows the original frame, the last two images illustrate the cross-correlation coefficient with 3D and 2D views. The minimum correlation values show the object boundaries (in yellow/green). Row 2 shows an example for case 2. There is negative correlation, since for real-world sequences, the background generates a lot of activity at the contour of the object due to the texture and noise. Finally, row 3 shows an example for case 3, in which the negative cross-correlation coefficient values separate two different textures.

5 Phase-based motion estimation

This work presents for the first time the use of frequency-based concepts for motion estimation with event-based sensors. Frequency-based methods or energy-based methods on classical image frames use the responses from a bank of filters tuned to different spatio-temporal frequencies. Different methods consider the energy of these responses [1, 25, 13]. These techniques usually estimate at a point the local spatial and temporal angular frequencies by computing responses to a bank of filters tuned to different frequencies. In our case, instead of directly using the frequency, we use the local phase [12]. Since the Gabor responses from the filter bank are complex valued, they can be expressed as in Eq. 4

$$g(\mathbf{x}, t) = Gabor(\mathbf{x}, t) * I(\mathbf{x}, t) = \rho(\mathbf{x}, t) e^{i\phi(\mathbf{x}, t)} \tag{4}$$

where ρ is the amplitude and ϕ the phase. Therefore, the filter responses can be used to extract the local phase $\phi(\mathbf{x}, t)$ (the angle of the odd and even parts). Now, an adapted Optical Flow Constraint that assumes the constancy of the spatio-temporal contours using the phase is formulated. Due to the aperture problem, only the velocity in the direction of the spatial phase gradient $\nabla_{\mathbf{x}}\phi$ can be computed. In this case, the normal flow $\mathbf{v}_n$ can be estimated as $\mathbf{v}_n = -\phi_t / ||\nabla_{\mathbf{x}}\phi||_2$ with ϕ_t the temporal derivative of the phase.

5.1 Frame-free estimation

Let us consider for example, a 1D sinusoid that is moving to the left. The events collected over a small time interval Δt are due to the difference of the sinusoid and its phase-shifted copy, as shown in the right part of Fig. 2 by the solid and dashed lines respectively. The signal of accumulated events shown in the same axis is another sinusoid of the same frequency. For the sake of clarity, this figure ignores that the input is logarithmic for our DVS.

Let us consider that the first sinusoid equation is $acos(\omega x(t))$, with a denoting the amplitude. Then the accumulated events can be expressed as a sinusoid of with a different amplitude but the same frequency than the original, and shifted.

$$S(x(t),t) = acos(\omega x(t)) - acos(\omega x(t) + \Delta t) = 2asin\Big(\frac{\Delta t}{2}\Big)cos\Big(\omega x(t) + \frac{1}{2}(\Delta t + C)\Big) \tag{5}$$

The signal of accumulated events (see Eq. 5) is shifted, but this shift is always the same and only depends on Δt. Therefore, an alternative estimation based on the accumulations of events is possible. Moreover, we assume that the local phase of the signal of accumulated events remains constant over a short time interval. The first step consists in defining the function of accumulated events for a small time interval as $\mathcal{S}\colon \mathbb{N}^2, \mathbb{R} \to \mathbb{N}$, that assigns to every position $\mathbf{x}$ the number of events that occur during that time interval at that position (Eq. 6).

$$\mathcal{S}(\mathbf{x},t) = k \sum_{t-\Delta t}^{t} e(\mathbf{x},t), \qquad k \in \mathbb{R} \tag{6}$$

Instead of image intensity we use this signal in the event-based framework, and the assumption of constant local image phase in [12] is rephrased as assuming the constancy of spatio-temporal contours of the signal of accumulated events. This way, the solution for the new Optical Flow Constraint is based on the estimation of the phase gradient of accumulated events and computed as in Eq. 7

$$\nabla\phi(\mathbf{x},t) = \frac{Im[g^*(\mathbf{x},t)\nabla g(\mathbf{x},t)]}{Re[g(\mathbf{x},t)]^2 + Im[g(\mathbf{x},t)]^2}, \tag{7}$$

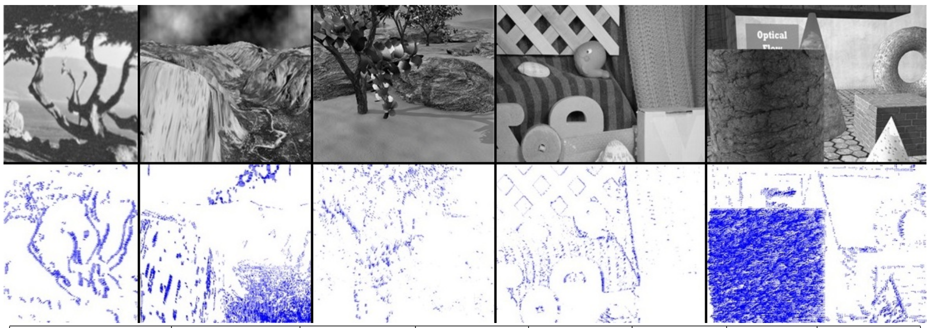
where $\nabla\phi$ is the phase gradient, with $g(\mathbf{x},t)$ the complex response of the Gabor filter bank for the signal $S(\mathbf{x},t)$, *Im* and *Re* are the imaginary and real parts, and g^* stands for the complex conjugate.

6 Results

This section compares our method to other event-based algorithms as well as conventional frame-based methods. We have created three benchmark datasets: in §6.1 using conventional benchmarks and creating synthetic events, in §6.2 using 3D scene model and motion ground-truth and creating synthetic events, and in §6.3 collecting events with the DVS mounted on a robotic mobile platform that provides its egomotion.

6.1 Classic synthetic scenes for event-based data

Due to the lack of event-based benchmark datasets, we created simulated data to compare our method to state-of-the-art Computer Vision algorithms. We select the central frames of sequences from the Middlebury dataset [2]. Using the provided ground-truth, the ground truth motion is derived for very small time intervals: e.g., we simulate about 50000 frames for every two frames. This means that



Method	Translation Tree	Diverging Tree	Yosemite	Rubberwhale	Dimetrodon	Grove2	Urban2
Event phase	16.6 (9.6%)	**29.8 (5.5%)**	**21.6 (12.4%)**	30.7 (5.2%)	42.5 (1.6%)	23.1 (5.3%)	**21.7 (2.2%)**
Fleet & Jepson [12]	98.4	94.7	97.3	95.4	NA	98.8	99.6
Otte-Nagel [20]	17.6	46.7	39.7	24.9	NA	38.9	86.0
Uras [24]	37.5	47.0	45.1	53.1	54.2	48.7	85.5
Horn-Schuck [14]	47.9	48.7	61.8	34.7	72.0	84.9	94.9
Lucas & Kanade [16]	51.8	55.3	61.3	44.6	61.6	79.2	88.3
Sun [22]	**1.6**	46.6	27.0	**11.6**	**4.7**	**3.4**	23.0
Events [3]	9 (5.4%)	33.6 (0.4%)	18.6 (0.5%)	27.3 (1.2%)	23.6 (1.4%)	NA	NA

Fig. 3. Phase-based motion estimation for "Translation tree", "Yosemite", "Rubberwhale", "Grove2" (from Middlebury [2]), and "029_Brickbox2t2" (from UCL dataset [17]). The table reports the relative AEPE (in %) and the density of valid values (in % and in parenthesis) for standard optic flow methods and event-based methods. Highlighted values correspond to the lowest errors but, when comparing event-based methods we highlight multiple methods, if the error is similar but the density is significantly different.

Sequence	Sun [22]	Phase events	Barranco [3]	Benosman [6]	Benosman [7]
Brickbox1	68.6 (15.5%)	**59.8** (15.5%)	73.8 (10.6%)	88.5 (10.0%)	88.8(17.9%)
Brickbox2	41.2 (32.7%)	**38.8** (32.7%)	68.5 (5.9%)	86.0 (31.7%)	71.7 (42.1%)
robot#1	NA	**44.2** (9.9%)	62 (10.3%)	88.7 (14.3%)	63.5 (2.5%)
robot#2	NA	**45.8** (8.9%)	66.2 (10.71%)	83 (3.4%)	60.5 (3%)
robot#3	NA	**42.3** (10.8%)	48.3 (8.75%)	80 (11.8%)	71 (9.4%)
robot#4	NA	**44.3** (7.11%)	51.2 (11.2%)	85.8 (12.2%)	77.1 (6.1%)

Table 1. Relative AEPE (in %) and density of valid values (in % and in parenthesis) for event-based motion estimation methods.

if conventional frames are obtained at 20 fps, our simulation provides 1000000 fps. Using the image motion for the short time intervals, interpolation gives us frame-by-frame the changes in intensity, from which we then derive the events and their timestamps at $1\mu s$ temporal resolution. Occluded and disoccluded regions are handled separately, since most of the conventional benchmarks do not provide the image motion for occluded regions. If the scene in the region and the camera are static, we fill in the intensity values with the ones extracted from the previous frame (for disoccluded regions), or the next frame (for occluded regions). If there is motion, for a texture-less region we also assume the background motion of the neighboring region, otherwise, we discard it.

The relative Average End Point Error is used as a measure of the accuracy of the estimation. We simulated the events for the following scenes: "Diverging Tree", "Translation Tree", and "Yosemite". These are synthetic sequences simulating a moving camera approaching a tree image, moving to the right in front of the same tree, and a fly-through over the Yosemite Valley. We also used the the real sequences "Dimetrodon" and "Rubberwhale", which were taken by static camera with different moving objects and the two scenes "Urban2" and "Grove2' with stronger textures and global motion.

Fig. 3 compares the proposed method, our contour-based event method [3] and top frame-based algorithms, among them the method of Sun [22], which ranked the first in 2014 in [2]. Since we are estimating normal flow, to compare with the ground-truth, we project the actual flow onto the gradient direction computed by our event-based method. In the figure, the 'density' denotes the percentage of points with normal flow data for a method, out the total number of points (as in [5, 4, 9, 23]), except the points for which there is no ground-truth available. Note that the error is reported for the exact same locations when comparing with the conventional frame-based techniques. Bold fonts highlight the lowest error in the comparison of the phase-based method to classical methods. Regarding to the contour-based method [3], it is included separately only for the sake of clarity. The error reported for this last method does correspond to different locations and thus, the densities are not the same.

Our phase-based method performs significantly better than many conventional algorithms, specifically [12, 20, 24, 14, 16]. The frame-based estimation provides dense estimates, but is bad at contours, where most of the events are coming from. The top method of Sun [22] used in the comparison, employs a number of sophisticated Computer Vision techniques, such as a hierarchical matching scheme for large motions, median filtering to reduce outliers, occlusions, etc. Even though it outperforms our method for four scenes, we are still doing better for the others. Conventional image-based algorithms perform better with global motion, since the flow changes smoothly along the scene, and this is precisely what many methods such as the one in Sun assume. Analyzing the sequences, our method outperforms Sun for the sequences with a strong texture component and with no global motion. The phase based method is better than the event-based method [3] for three scenes. But the methods should not be compared generally, as address different regions, which is reflected in the different densities.

Standard datasets are not well suited for evaluating event-driven methods since often they have carefully chosen small inter-frame distances, there is no data at occlusions, and there is no 3D information. Nevertheless, the average error for our phase-based method is 26.5 with 5.9% density, while the method in [3] achieves 22.4 but with only 1.78% valid values (3 times less).

6.2 Virtual scenes for event-based data

We created a dataset of synthetic sequences for evaluation of event-based methods. We used the sequences "026_Brickbox1t1", "029_Brickbox2t2" from the dataset [17]. These are two sequences with differently textured objects and with large inter-frame displacements. Using the know 3D models and 3D motion, we simulated the events to obtain our the ground-truth. Using this data we

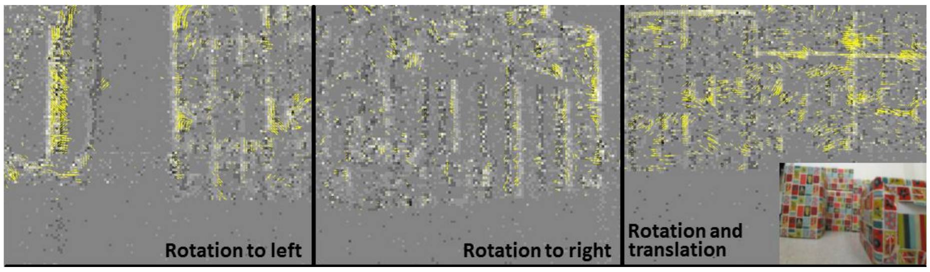


Fig. 4. Example of the scenes collected with the sensor mounted on a mobile platform. The image in the right shows a capture of the set up captured with a camera.

evaluated the accuracy of different different event-based algorithms [3, 7, 6] in Table 1. To allow for comparison to conventional techniques, the table also provides the results for the Sun method [22] . The motion estimation for the "029_Brickbox2t2" sequence is shown in Fig. 3.

From Table 1 we can infer that our method achieves better results in the relative AEPE for these high-frequency texture sequences. Our method performs even better than Sun's method, with an error reduction of 5 points (considering same density). In comparison to previous works our method achieves an average improvement of 30% with 4.1% more estimates. We use the same parameters described in the paper for [3], for [6] we set $th1 = 1000\mu s$ and $th2 = 0.25$, and for [7] we use $\Delta t = 500\mu s$ and 5×5 windows for the least-squares estimation. We assumed that the real sequences were recorded at 40 fps, and we simulated a temporal resolution of $10\mu s$ with 2500 samples between two consecutive frames.

6.3 Real scenes for event-based data

We also collected real data using the DVS camera on-board a mobile platform. We collected four sequences: "robot#1" and "robot#2" are pure zoom-in motion(translations for Z = 0.2 m/s); "robot#3" is a pure rotation (roll = 0, pitch = 0, yaw = $\pi/4$) and "robot#4" has both rotation (roll = 0, pitch = 0, yaw = $\pi/4$) and translation (Y = -0.15 m/s, Z = 0.5 m/s). The estimated motion is for 1s. The ground-truth data was derived from the odometry of the robot, the sensor parameters (focal length and optic center), and depth measurement of the objects, which were positioned fronto-parallel to the sensor.

The setup consisting of three textured boxes at different depths, and the estimated motion for three sequences superimposed on the image of accumulated events are show in Fig. 4. As seen in Table 1, the algorithm presented in this paper achieves better results than any of the previous event-based methods in the literature. The difference is most significant for "robot#3", where due the rotation, the vertical components of the textures cause more events and reduce the accuracy of other methods. In these scenes the edges are not clean but appear broader caused by the textures in the background. This causes previous methods to fail while the method presented in the paper still achieves reasonable accuracy.

The average error reduction in the worst case is about 13%, and in the best case about 40%, considering a similar number of image points with estimates.

7 Conclusions

Current event-based image motion estimation techniques have good performance at contours, where classical conventional methods fail. Using address events or a combination of event data with the intensity signal they have been shown to reduce computational complexity and perform at real time. However, existing methods rely on the assumption that close-by pixels from the same edge fire events also close in time, which is false for textured edges. In this paper we introduced an approach, that deals with such textured edges. We first presented a simple method for locating such textured edges. Second, we presented a method that uses the local phase of the event signal to accurately estimate image motion. Using the local phase instead of trying to reconstruct the intensity signals as in previous works, allows us to avoid the problem with textured edges.

We presented experimental results comparing the method to classic Computer Vision methods and other event-based approaches using synthetic sequences, and sequences collected with a sensor mounted on a mobile platform. Our method reduces the error in synthetic sequences, but it is hard to evaluate the significance the results. The results are more obvious for real scenes with textured objects, where the method was shown to decrease the error up to 30%.

We suggest that an optimal strategy, would be to use different techniques for image motion estimation, depending on the structure of data near a pixel. In future work we plan to develop a systematic method to decide for a given pixel, which method to use. Current event-based techniques are good for clean contours, as there they are accurate and have real-time performance, while our phase-based method requires extended time support for the filters, but has higher accuracy for textured contours.

Acknowledgments. The support of the European Union under the EU Marie Curie fellowship (FP7-PEOPLE-2012-IOF-33208) and the Cognitive Systems program (project POETICON++), the Junta de Andalucia VITVIR project (P11-TIC-8120), the National Science Foundation under INSPIRE grant SMA 1248056, and by DARPA through U.S. Army grant W911NF-14-1-0384 under the Project: Shared Perception, Cognition and Reasoning for Autonomy are gratefully acknowledged.

References

1. Adelson, E.H., Bergen, J.R.: Spatiotemporal energy models for the perception of motion. Journal of Optical Society of America, A 2(2), 284–299 (1985)
2. Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. Int. Journal Computer Vision 92(1), 1–31 (2011)

3. Barranco, F., Fermuller, C., Aloimonos, Y.: Contour motion estimation for asynchronous event-driven cameras. Proc. of the IEEE 102(10), 1537–1556 (2014)
4. Barranco, F., Tomasi, M., Diaz, J., Vanegas, M., Ros, E.: Parallel architecture for hierarchical optical flow estimation based on FPGA. IEEET on VLSI 20(6), 1058–1067 (2012)
5. Barron, J.L., Fleet, D.J., Beauchemin, S.S.: Performance of optical flow techniques. Int. Journal Computer Vision 12, 43–77 (1994)
6. Benosman, R., Clercq, C., Lagorce, X., Ieng, S.H., Bartolozzi, C.: Event-based visual flow. IEEET Neural Networks Learning Systems 25(2), 407–417 (2014)
7. Benosman, R., Ieng, S.H., Clercq, C., Bartolozzi, C., Srinivasan, M.: Asynchronous frameless event-based optical flow. Neural Networks 27, 32–37 (Mar 2012)
8. Berner, R., Brandli, C., Yang, M., Liu, S.C., Delbruck, T.: A 240 x 180 10mw 12us latency sparse-output vision sensor for mobile applications. In: VLSI Circuits (VLSIC), 2013 Symposium on. pp. C186–C187 (June 2013)
9. Brandt, J.: Improved accuracy in gradient-based optical flow estimation. Int. Journal of Computer Vision 25(1), 5–22 (1997)
10. Brodsky, T., Fermüller, C., Aloimonos, Y.: Structure from motion: Beyond the epipolar constraint. Int. Journal Computer Vision 37(3), 231–258 (2000)
11. Fermüller, C.: Passive navigation as a pattern recognition problem. Int. Journal of Computer Vision 14(2), 147–158 (1995)
12. Fleet, D.J., Jepson, A.D.: Computation of component image velocity from local phase information. Int. Journal of Computer Vision 5(1), 77–104 (1990)
13. Heeger, D.J.: Optical flow using spatiotemporal filters. Int. Journal of Computer Vision 1(4), 279–302 (1988)
14. Horn, B.K.P., Schunck, B.G.: Determining optical flow. AI 17, 185–203 (1981)
15. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128x128 at 120db 15us latency asynchronous temporal contrast vision sensor. IEEE SSC 43(2), 566–576 (2008)
16. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. Conf. on Artificial Intelligence 2, 674–679 (1981)
17. Mac Aodha, O., Humayun, A., Pollefeys, M., Brostow, G.: Learning a confidence measure for optical flow. IEEET Pattern Analysis Machine Intelligence 35(5), 1107–1120 (2013)
18. Mead, C.: Neuromorphic electronic systems. P. of IEEE 78(10), 1629–1636 (1990)
19. Orfanidis, S.: Introduction to Signal Processing. Prentice Hall international editions, Prentice Hall (1996)
20. Otte, M., Nagel, H.H.: Optical flow estimation: Advances and comparisons. In: European Conference Computer Vision, vol. 800, pp. 49–60 (1994)
21. Sun, D., Roth, S., Black, M.J.: Secrets of optical flow estimation and their principles. In: Computer Vision and Pattern Recognition. pp. 2432–2439 (2010)
22. Sun, D., Roth, S., Black, M.: A quantitative analysis of current practices in optical flow estimation and the principles behind them. Int. Journal of Computer Vision 106(2), 115–137 (2014)
23. Tomasi, M., Barranco, F., Vanegas, M., Díaz, J., Ros, E.: Fine grain pipeline architecture for high performance phase-based optical flow computation. Journal of Systems Architecture 56(11), 577–587 (2010)
24. Uras, S., Girosi, F., Verri, A., Torre, V.: A computational approach to motion perception. Biological Cybernetics 60(2), 79–87 (1988)
25. Watson, A.B.: Model of human visual-motion sensing. Journal of The Optical Society of America A-optics Image Science and Vision 2 (1985)