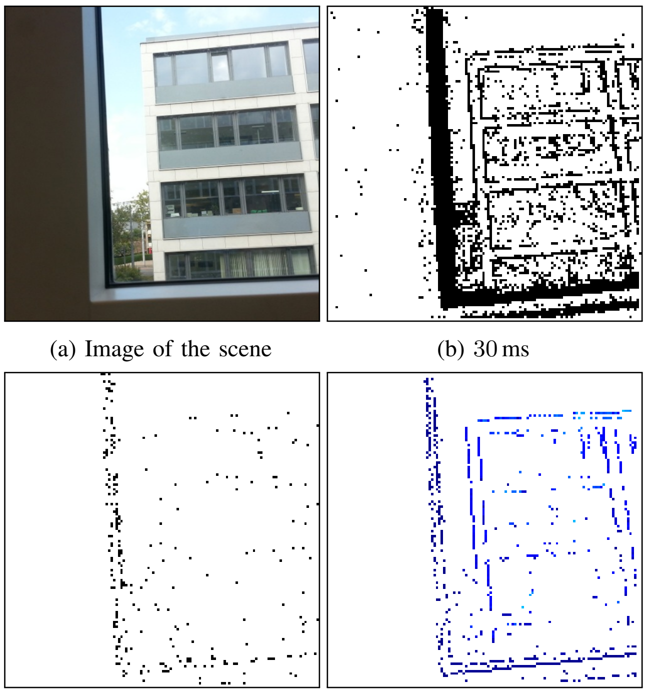
Lifetime Estimation of Events from Dynamic Vision Sensors

Elias Mueggler, Christian Forster, Nathan Baumli, Guillermo Gallego and Davide Scaramuzza

***Abstract*— We propose an algorithm to estimate the "lifetime" of events from retinal cameras, such as a Dynamic Vision Sensor (DVS). Unlike standard CMOS cameras, a DVS only transmits pixel-level brightness changes ("events") at the time they occur with *micro*-second resolution. Due to its low latency and sparse output, this sensor is very promising for high-speed mobile robotic applications. We develop an algorithm that augments each event with its lifetime, which is computed from the event's velocity on the image plane. The generated stream of augmented events gives a continuous representation of events in time, hence enabling the design of new algorithms that outperform those based on the accumulation of events over fixed, artificially-chosen time intervals. A direct application of this augmented stream is the construction of sharp gradient (edge-like) images at any time instant. We successfully demonstrate our method in different scenarios, including high-speed quadrotor flips, and compare it to standard visualization methods.**



(a) Image of the scene (b) 30 ms

(c) 1 ms (d) Our method

Fig. 1: The Dynamic Vision Sensor (DVS) is moved in front of a window frame diagonally, from bottom-left to top-right (a). Since the window frame is much closer than the buildings, its apparent motion is significantly larger. Thus, if we use a fixed accumulation interval, the images will either be blurred, if the interval is too long (b), or some structures will not be clearly visible, if the interval is too short (c). Our method estimates the lifetime of each event independently and displays the event for that period of time (d).

I. Introduction

A. Motivation

Event-based (retinal) vision sensors [1], such as the Dynamic Vision Sensor [2], offer great potential for robotic applications: since only pixel-level brightness *changes* are transmitted, less bandwidth is required and less data must be processed. In addition, these changes are transmitted at the time they occur with minimal latency, which is in the order of a few *micro*-seconds. Due to the asynchronous nature of these changes, they are called *events*.

However, since an event stream is fundamentally different from video streams of standard CMOS cameras, new algorithms are required to deal with this data. Event-based adaptations of iterative closest points [3] and optical flow [4], [5] have been proposed. Recently, event-based visual odometry [6], [7], tracking [8], [9], and Simultaneous Localization And Mapping (SLAM) [10] algorithms were also presented. The design goal of such algorithms is that each incoming event can asynchronously change the estimated state, thus preserving the event-based nature of the sensor.

While all of these algorithms implicitly buffer a certain number of past events, we propose to explicitly model the set of active events. We consider an event active as long as the brightness gradient causing this event is visible by the pixel. The estimation of such a set of active events has several applications, such as the generation of sharp gradient images at any point in time, clustering of events for tracking of multiple objects, etc. Here, we focus on the first one (see Fig. 1). This also allows applying standard computer-vision algorithms on these images without modification.

B. Related Work

Due to its low latency and low bandwidth, the DVS [2] is a promising sensor for robotic systems with limited computational power and short time constants. An impressive demonstration of these capabilities was presented in [11]. Using two DVS, the authors implemented a pencil-balancing system on a highly-reactive platform free to move on a plane. A robotic goalkeeper with a reaction time of 3 ms was presented in [12]. More recently, robot localization was demonstrated using a DVS during high-speed maneuvers [9], where rotational speeds of up to 1,200 °/s were measured during quadrotor flips.

Standard computer-vision algorithms cannot be applied directly to the output of event-based vision sensors, since

The authors are with the Robotics and Perception Group, University of Zurich, Switzerland—`http://rpg.ifi.uzh.ch`. This research was supported by the Swiss National Science Foundation through project number 200021-143607 (Swarm of Flying Cameras), the National Centre of Competence in Research Robotics, and Google.

they do not provide grayscale intensity images. A straightforward workaround is to generate such intensity images by accumulating events over a fixed time interval and then apply standard frame-based algorithms. An event-to-frame converter was presented in [13] and tested on two conventional stereo-vision algorithms. Another example of DVS event accumulation was shown in [14], where events were accumulated in artificial time slots of 5–50 ms and used in stereo vision for tracking moving objects. In both cases, the event-to-frame conversion was a time-consuming process that introduced some latency and, therefore, the asynchronous data delivery and high temporal resolution of the DVS was not used very efficiently.

In [3], the events in a sliding window of fixed duration were selected as input of an Iterative Closest Point (ICP) algorithm that was used to guide a micro gripper to grasp an object with a mean update rate of 4 kHz. In this particular setup, such a fixed duration could be chosen for all the pixels of the DVS, because the gripper was moving at almost constant speed parallel to the image plane. In a general configuration, however, such a time interval does not exist.

C. *Contributions and Outline*

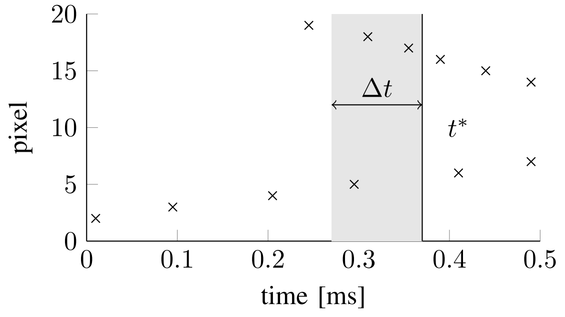
In this paper, we present a method to augment data streams from event-based cameras with their *lifetime* and the velocity of each event, while simultaneously filtering noise. Our method is based on the event-based optical flow [4], [5] to estimate the velocity of an event from where we can estimate its lifetime. As a direct application, this method allows rendering *sharp* gradient images at *any* point in time, as illustrated in Fig. 2.

In contrast to previous algorithms, our method does not depend on a temporal window $[t - \Delta t, t + \Delta t]$ around the event time t. Thus, we eliminate both a tuning parameter (Δt) and its corresponding latency (our method uses only *past* events). Our method is also robust against noise because we use RANSAC [15] and a regularization term. The output of the method can be used to apply standard computer-vision algorithms to the output of event-based cameras.

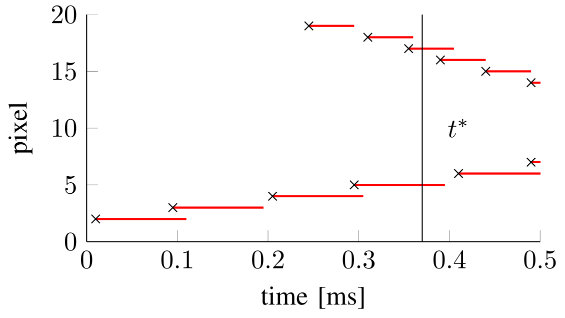
The remainder of the paper is organized as follows. In Section II, we characterize the Dynamic Vision Sensor (DVS). The developed algorithm to calculate the lifetime of an event is described in Section III and evaluated in Section IV.

II. Dynamic Vision Sensor

Standard CMOS cameras send full frames at fixed frame rates. On the other hand, event-based (retinal) cameras such as the DVS [2] have independent pixels that generate spike events at local relative brightness changes in continuous time. These events are timestamped and transmitted asynchronously at the time they occur using sophisticated digital circuitry. Each event e is a tuple $\langle x, y, t, p\rangle$, where x, y are the pixel coordinates of the event, t is the timestamp of the event, and $p \in \{-1, +1\}$ is the polarity of the event, which is the sign of the brightness change. This representation is sometimes also referred to as Address-Events Representation



(a) Raw event stream



(b) Event stream augmented with lifetime

Fig. 2: In this illustration, we consider a single pixel row of a DVS, which observes two edges moving at different speeds (cf. Fig. 4). The events are marked with crosses. To visualize the events at time t^*, the active events at that time are plotted. If a fixed accumulation interval Δt is chosen (a), some regions become blurred (the upper, fast edge is represented with two events) or others are not complete (if we chose a shorter interval, the slow edge would not be considered). Our method (b) assigns a *lifetime* to each event (shown in red), thus the active events at t^* are known.

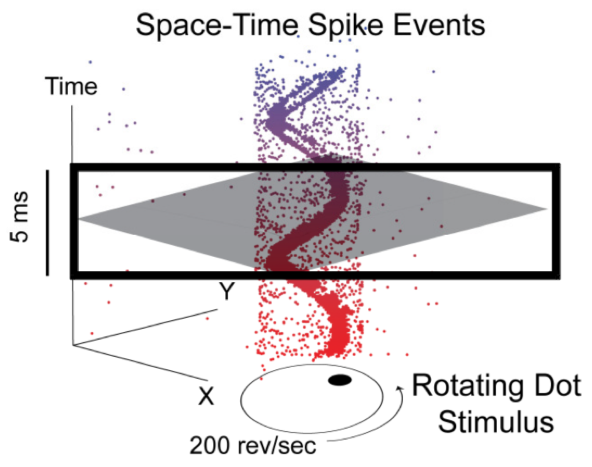


Fig. 3: Visualization of the output of a DVS looking at a rotating dot. Colored dots mark individual events. The polarity of the events is not shown. Events that are not part of the spiral are caused by sensor noise. Figure adapted from [16].

(AER). The DVS has a resolution of 128×128 pixels and is connected via USB. A visualization of the output of the DVS is shown in Fig. 3.

Due to its low latency and high temporal resolution, both in the range of *micro*-seconds, the DVS is a very promising sensor for high-speed mobile robot applications. Since the data stream from the DVS is sparse (only *changes* are reported), the bandwidth and computational load are low. An additional advantage for robotic applications is the DVS' high dynamic range of 120 dB (compared to 60 dB of expensive computer-vision cameras), which allows both indoor and outdoor operation without changing parameters. Since all pixels are independent, these contrasts can also take place within the same scene.

III. Algorithm

In this section, we devise our algorithm to estimate the lifetime of each incoming event. The basic idea is to determine each event's velocity $\mathbf{v} = (v_x, v_y)^\top$ on the image plane and use this information to calculate the time interval that this event is considered active. The lifetime τ indicates how long it will take for the brightness gradient at the current event location to trigger a new event in a neighboring pixel. We assign zero lifetime to noise events, $\tau = 0$.

Our algorithm augments the stream of events $\langle x, y, t, p\rangle$ with the the lifetime τ and the event's velocity $\mathbf{v}$,

$$\langle x, y, t, p\rangle \mapsto \langle x, y, t, p, \tau, v_x, v_y\rangle. \qquad (1)$$

The velocity of the event $\mathbf{v}$ is computed using event-based visual flow, which is estimated based on the method introduced in [4], [5]. We first present our adaptation of the event-based visual flow and the computation of the lifetime for each event. Then, we detail the local plane-fitting algorithm including outlier rejection and regularization.

A. Event-Based Visual Flow and Lifetime

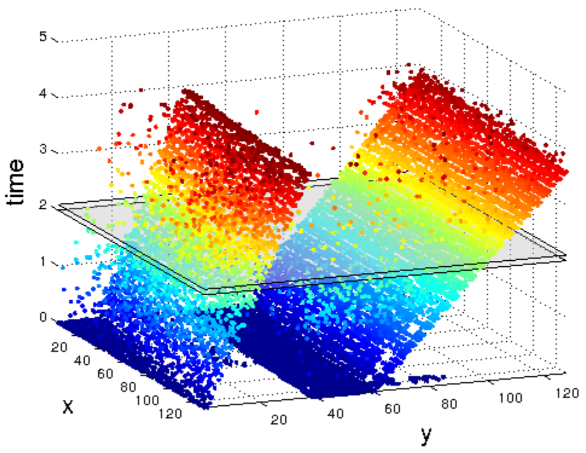
The Surface of Active Events (SAE) is defined in the three-dimensional spatio-temporal domain that is composed of the two-dimensional sensor frame and an additional dimension representing time [17]. Each incoming event generates or updates a point on the surface, such that, for each pixel position on the image plane, the time value of the surface is equal to the timestamp of the last event at this position. The SAE is given by the map $\Sigma_e : \mathbb{R}^2 \to \mathbb{R}$, $t = \Sigma_e(\mathbf{p})$, where $\mathbf{p} = (x, y)^\top$. In space-time, a point of the SAE is represented by the 3-vector $S(\mathbf{p}) = (x, y, \Sigma_e(x, y))^\top$. In this sense, $\Sigma_e(\mathbf{p})$ represents the SAE as an "elevation map".

Figure 4a shows the SAE of real data recorded with the DVS in the spatio-temporal domain. The recorded sequence contains two lines moving at different speeds, hence the different slopes. Sensor noise is clearly visible as isolated dots. Figure 4b shows the corresponding visualization of a 100 ms slice. The latter corresponds to what we refer to as the *naive* method that accumulates events over a fixed time interval.

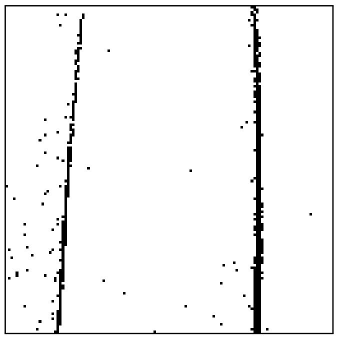
The planar approximation of the SAE at an event's location $\mathbf{p}$ is given by the first order Taylor expansion

$$S(\mathbf{p} + \Delta\mathbf{p}) \approx S(\mathbf{p}) + \big(S_x(\mathbf{p}), S_y(\mathbf{p})\big)\Delta\mathbf{p}, \qquad (2)$$

where $S_x = \frac{\partial S}{\partial x} = (1, 0, \frac{\partial \Sigma_e}{\partial x})^\top$, $S_y = \frac{\partial S}{\partial y} = (0, 1, \frac{\partial \Sigma_e}{\partial y})^\top$ are the first partial derivatives of S, representing vectors in the tangent space to S.



(a) Surface of Active Events with a slice shown in (b).



(b) All events between $t = 2\,\text{s}$ and $2.1\,\text{s}$.

Fig. 4: Surface of Active Events (SAE) of two lines moving to the right. The left line moves slower than the right one. Events caused by sensor noise are visible as isolated dots. A 100 ms slice starting at $t = 2\,\text{s}$ is shown in (b), which corresponds to the naive method that accumulates events over a fixed time interval (here, 100 ms). While the slow line appears sharp, the fast line is several pixels wide, which corresponds to motion blur.

As illustrated in Fig. 5, we define the lifetime of the event at $(\mathbf{p}, t)$ as the first order approximation of the maximum temporal increment of S for a displacement $\|\Delta\mathbf{p}\| = 1$ pixel:

$$\tau(\mathbf{p}) = \max \Delta t \quad \text{subject to} \quad \|\Delta\mathbf{p}\| = 1, \qquad (3)$$

where $\Delta t = \langle S(\mathbf{p} + \Delta\mathbf{p}) - S(\mathbf{p}), \mathbf{e}_3\rangle$, $\mathbf{e}_3 = (0, 0, 1)^\top$ is the direction of the time axis and $\langle\cdot,\cdot\rangle$ is the standard inner product in $\mathbb{R}^n$.

The lifetime τ, therefore, indicates the maximum amount of time before the brightness gradient at the current event location will trigger a new event in a neighboring pixel.

Substituting (2) and the expressions for S_x, S_y in (3) yields $\Delta t = \mathbf{e}_3^\top\big(S_x(\mathbf{p}), S_y(\mathbf{p})\big)\Delta\mathbf{p} = \langle\nabla\Sigma_e(\mathbf{p}), \Delta\mathbf{p}\rangle$, with $\nabla\Sigma_e(\mathbf{p}) = \big(\frac{\partial\Sigma_e}{\partial x}(\mathbf{p}), \frac{\partial\Sigma_e}{\partial y}(\mathbf{p})\big)^\top$. Hence we arrive at the equivalent definition

$$\tau(\mathbf{p}) = \max\,\langle\nabla\Sigma_e(\mathbf{p}), \Delta\mathbf{p}\rangle \;\text{ subject to }\; \|\Delta\mathbf{p}\| = 1. \qquad (4)$$

Since t is an increasing function, Σ_e is a monotonically increasing function of $\mathbf{p}$, thus it has nonzero gradient at any

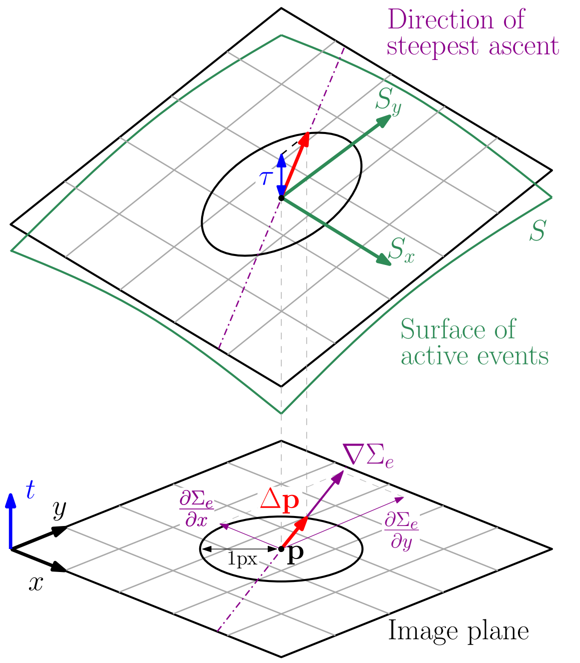


Fig. 5: Visualization of the lifetime τ: the maximum time increment Δt of the planar approximation to the Surface of Active Events (SAE) for a displacement of $\|\Delta \mathbf{p}\| = 1$ pixel. Both the optimal unit displacement $\Delta \mathbf{p}$ and τ are directly related to $\nabla \Sigma_e$, as summarized in (6).

point $\mathbf{p}$, and $\nabla \Sigma_e(\mathbf{p})$ is related to the velocities describing the visual flow (see [5]) according to

$$\nabla \Sigma_e(\mathbf{p}) = \left(v_x^{-1}(\mathbf{p}), v_y^{-1}(\mathbf{p})\right)^\top. \tag{5}$$

The local planar approximation is equivalent to assuming constant velocities v_x and v_y.

The maximum (4) is achieved for the unit vector $\Delta \mathbf{p} = \nabla \Sigma_e(\mathbf{p}) / \|\nabla \Sigma_e(\mathbf{p})\|$ (see Fig. 5). Hence,

$$\tau(\mathbf{p}) = \|\nabla \Sigma_e(\mathbf{p})\| = \sqrt{v_x^{-2} + v_y^{-2}}. \tag{6}$$

Next, we give a formula for τ in terms of the normal to the surface S at $\mathbf{p}$ (i.e., the normal of the tangent plane), $\mathbf{n}(\mathbf{p}) = (n_1, n_2, n_3)^\top$, which is assumed to be known by fitting a plane to the data (this step will be described next in Section III-B). We may further assume that $n_3 > 0$ since Σ_e is a monotonically increasing function. The normal is given by $\mathbf{n}(\mathbf{p}) \propto S_x(\mathbf{p}) \times S_y(\mathbf{p}) = (-(\nabla \Sigma_e(\mathbf{p}))^\top, 1)^\top$. Substituting (5) gives $\mathbf{n}(\mathbf{p}) \propto (-v_x^{-1}, -v_y^{-1}, 1)^\top$ in terms of the motion velocity and, identifying coordinates with $\mathbf{n}(\mathbf{p}) = (n_1, n_2, n_3)^\top$, we obtain $-v_x^{-1} = n_1/n_3$ and $-v_y^{-1} = n_2/n_3$, which finally implies

$$\tau(\mathbf{p}) = \sqrt{v_x^{-2} + v_y^{-2}} = \frac{1}{n_3}\sqrt{n_1^2 + n_2^2}. \tag{7}$$

B. Local Plane-fitting Algorithm

Our plane-fitting algorithm is based on [5], where all events in an $N \times N \times 2\Delta t$ window, centered around the current event, in the spatio-temporal domain are used to estimate the local plane. However, their algorithm has two

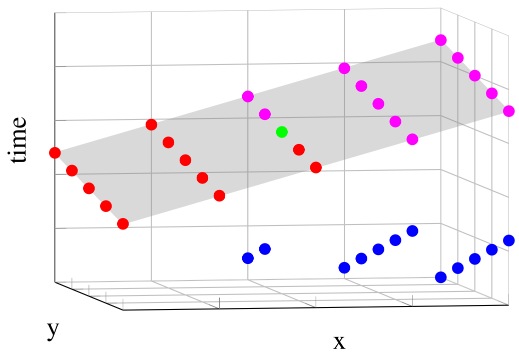


Fig. 6: When a new event (green) arrives, the SAE on a 5×5 patch around it includes the red and blue events. The events in red correspond to the brightness gradient that moves over this patch. The events in blue correspond to another gradient that moved over this patch previously. The events in magenta are future events to which we do not have access. However, under the local planar assumption, the magenta events will lie on the same plane (gray) as the red events. To avoid latency in our algorithm, we only use the newer half of events from the SAE (red events) to estimate the plane, while the blue events are not considered (see Section III-B).

undesirable properties: first, it introduces a tuning parameter (Δt) that limits the slowest detectable gradients (the slope of the plane). Second, events from the future are included, which translates to introducing a Δt latency. To overcome these issues, we only use *past* events in our estimation. Since we assume local smoothness, we can only use half of the events of the $N \times N$ window around the current event. This is illustrated in Fig. 6.

To robustly fit the plane, we use the RANSAC algorithm [15]. We compute a candidate plane using the new event and two additional past events that were chosen randomly. We then check all other past events whether they support the candidate plane. A past event is considered an inlier, if its point-to-plane distance is below the inlier threshold μ. The second tuning parameter of the RANSAC algorithm is the estimated percentage of outliers ϵ, which can further be used to estimate the necessary number of iterations. If less than m inliers are found, the event is considered as noise and its lifetime is set to zero. We compute m as a function of half the events in the window of size N and the percentage of outliers ϵ,

$$m = \underbrace{(1-\epsilon)}_{\substack{\text{inlier}\\\text{ratio}}} \underbrace{N^2/2}_{\substack{\text{maximum}\\\text{support}}}. \tag{8}$$

Both parameters have to be empirically tuned and they vary for different scenes and DVS settings. We found $\mu = 10^{-4}$ and $\epsilon = 0.4$ to yield good results. Note that we split incoming events by their polarity, i.e., we run our algorithm separately for the positive and negative events, and combine the output of both for the final result.

1) Plane Fitting: Let $\mathbf{A}$ be the matrix of the n inliers obtained by the RANSAC algorithm,

$$\mathbf{A} = \begin{pmatrix} x_1 & y_1 & t_1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & t_n \end{pmatrix}, \tag{9}$$

where here $x_i, y_i, t_i, i \in \{1, \dots n\}$ are local coordinates relative to the current event.

The ordinary least-squares solution of the plane normal $\mathbf{n}_{LS}$ is given by

$$\mathbf{n}_{LS} = \arg\min_{\mathbf{n}} \|\mathbf{A}\mathbf{n} - \mathbf{b}\|^2, \tag{10}$$

where $\mathbf{b} = (1 \cdots 1)^\top$.

2) Regularization: To refine the estimate of the plane normal, we predict future events using the local constant-velocity assumption, which serves as regularization. Using the estimated velocity for a new event, we predict the time $\hat{t}$ for all neighboring pixels at which an event should occur. We then compare the time an event actually occurs with the predicted time. We use this difference as a measure of how much we trust the previously fitted plane. The absolute error

$$\Delta t_{\text{err}} = |t_i - \hat{t}_i| \tag{11}$$

between the predicted time $\hat{t}_i$ and the actual time t_i is, therefore, used for an error-dependent regularization weight $\lambda(\Delta t_{\text{err}})$. A regularized plane $\mathbf{n}_R$ is computed,

$$\mathbf{n}_R = \arg\min_{\mathbf{n}} \left(\|\mathbf{A}\mathbf{n} - \mathbf{b}\|^2 + \lambda(\Delta t_{\text{err}}) \|\mathbf{n} - \hat{\mathbf{n}}_i\|^2 \right), \tag{12}$$

where $\hat{\mathbf{n}}_i$ is the predicted plane normal.

The value of the error-dependent regularization weight $\lambda(\Delta t_{\text{err}})$ in (12) gives an indication about the preference of the prior information, e.g., for small prediction errors, the prior information is considered reliable and is therefore weighted stronger. Therefore, $\lambda(\Delta t_{\text{err}})$ should be big for small errors. An exponential approach is chosen to satisfy this condition. To enforce general smoothness on the motion, a constant value can be added to the exponential function. For the experiments described in this paper, the following function is used:

$$\lambda(\Delta t_{\text{err}}) = 9 + 100 \exp(-0.005 \Delta t_{\text{err}}). \tag{13}$$

3) Edge Thinning: If a gradient is accelerating and thus violating the constant-velocity assumption, the lifetime will be overestimated. Therefore, two neighboring events in the direction of motion will be active at the same time, causing a similar effect as motion blur. A simple solution to this problem is to use the velocity information of the new event to reset the lifetime of the neighboring pixel in negative velocity direction. This technique effectively suppresses motion blur caused by accelerating gradients.

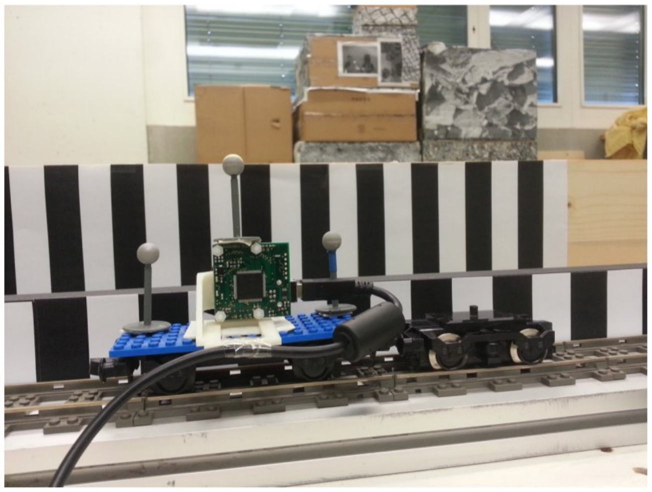


Fig. 7: The DVS is mounted on a train cart to achieve constant linear velocity. The scene is divided in three parts, which are at different depths: two line patterns (at $0.1\,\text{m}$ and $0.2\,\text{m}$) and the background consisting of boxes and windows (at $5\,\text{m}$).

IV. Experimental Evaluation

We evaluate our algorithm using four different experiments, going from controlled environments to urban settings. We visually compare the output of our method with that of the naive method that accumulates events over a fixed time interval.

A. Experiment 1: Line Pattern at Constant Velocity

1) Experimental Setup: The first experiment investigates the response to straight lines at different depths when the DVS moves parallel to the pattern at a constant velocity (see Fig. 7). To enforce constant linear velocity, the DVS is mounted on a train cart. Two boards with distinct vertical black and white bars are installed in front of the DVS at different distances. In the DVS, both boards as well as part of the background containing cardboard boxes and windows are visible.

2) Results: Figure 8 shows the event-stream visualization using the naive method with a fixed accumulation interval of $1\,\text{ms}$ (a) and $30\,\text{ms}$ (b) along with our algorithm both without (c) and with (d) regularization. Both intervals are not suitable for this setup, resulting in motion blur ($30\,\text{ms}$) or hardly recognizable structure ($1\,\text{ms}$). Our algorithm detects the lines and estimates their velocities coherently. The colors in the visualization correspond to the lifetime. The slow apparent background motion, visible in Fig. 8b, is only partially captured by our method due to the bad signal-to-noise ratio for slow apparent motion. Qualitative comparison of the output with and without regularization shows that both perform similarly, with regularization performing slightly better when it comes to noise suppression. In this case of pure translation, the lifetime is proportional to the inverse depth of the scene. Hence, another application of the estimation of the lifetime of the events is the recovery of the structure (i.e., depth) of the scene.

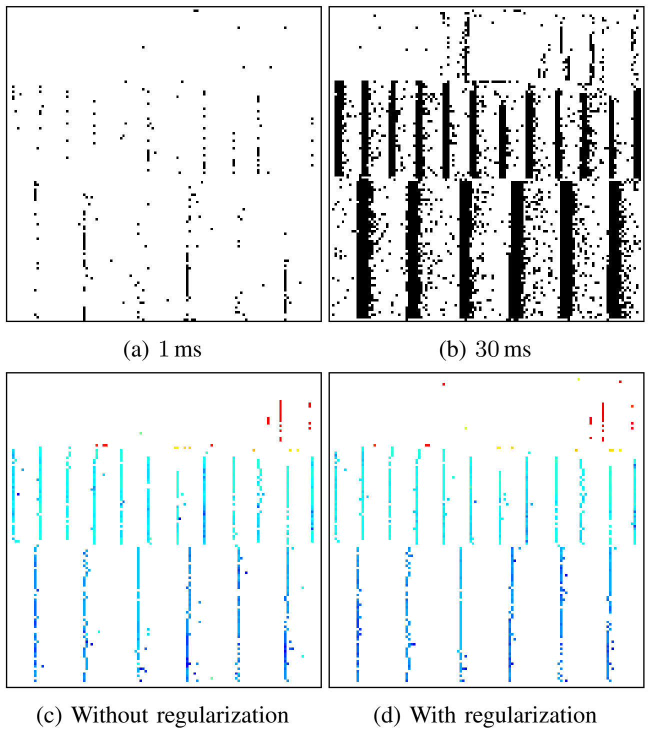


Fig. 8: Experiment 1: DVS moves at constant velocity in front of a striped pattern. The experimental setup is shown in Fig. 7. Accumulating events over a fixed time interval results in either unclear structures (a) or motion blur (b). Since the apparent motion changes over the image space, no fixed interval exists that can render sharp images. Our method delivers sharp images as well as suppresses noise. Long and short lifetimes are depicted in red and blue, respectively.

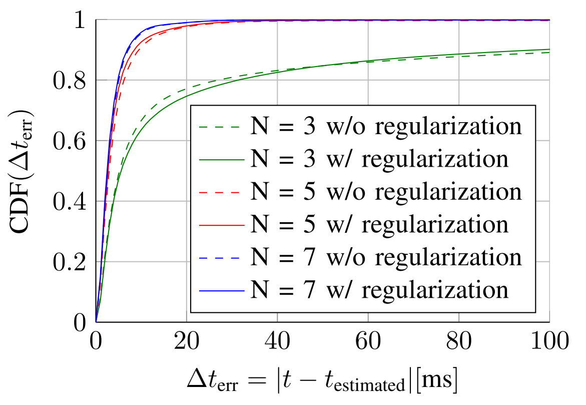


Fig. 9: Prediction error analysis of Experiment 1 for different window sizes both with and without regularization.

Figure 9 shows the fraction of cumulated predictions plotted against their absolute error. For instance, using $N = 5$ more than 90 % of all predictions have a smaller error than 10 ms. This quantitative evaluation for different window sizes does not show a significant difference between the algorithm with or without regularization for window sizes of $N = 5$ and $N = 7$. The effect of the regularization becomes much clearer when looking at the distribution of the estimated lifetimes (Fig. 10). There is a stronger segregation of the two main expected lifetimes as a result of the regularization.

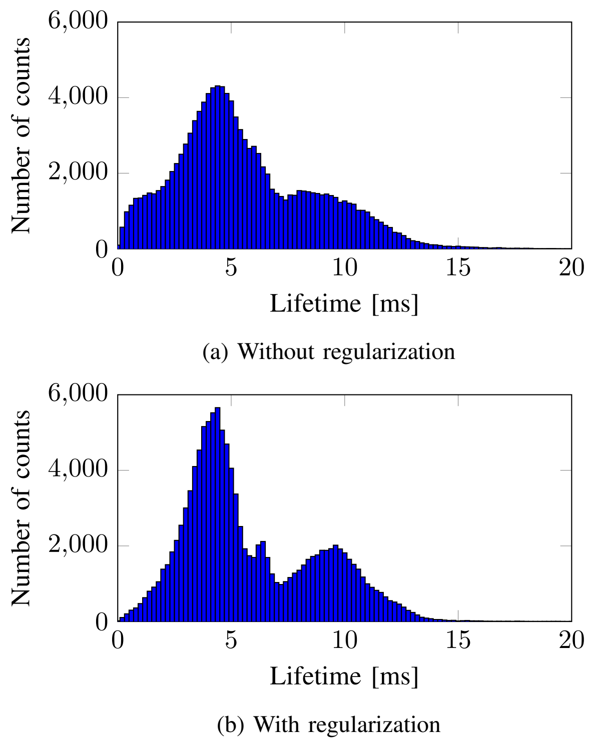


Fig. 10: Lifetime histogram of Experiment 1 for $N = 5$: events with an assigned lifetime within 0.2 ms are binned. The two peaks correspond to the close and far lines in the scene (cf. Fig. 7). Note that with regularization (b), the two peaks are much sharper than without (a).

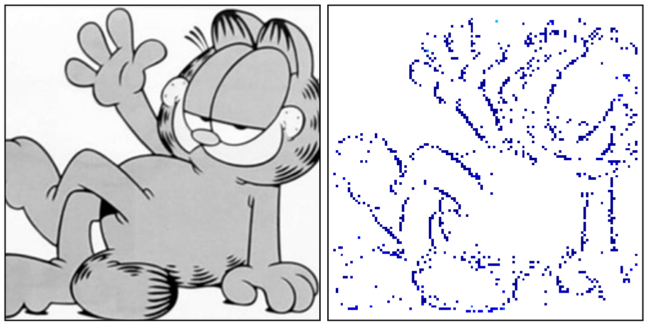
B. *Experiment 2: Complex Patterns at Constant Velocity*

1) Experimental Setup: The second experiment investigates the response to complex patterns. We used the same train cart to move the DVS at constant velocity parallel to a complex pattern (see Fig. 11a). In this experiment, however, the entire pattern has constant distance to the DVS.

2) Results: Figure 11b shows the output of our algorithm for Experiment 2. The silhouette of “Garfield” is clearly visible even though most edges are curved. Many details are preserved well. However, some details are too small to be captured by the low resolution of the DVS, which is 128×128 pixels. Horizontal edges are not visible, as they are parallel to the apparent motion and, therefore, do not trigger any events. In this setup, a well-tuned fixed lifetime would achieve similar results, but without noise suppression.

C. *Experiment 3: Quadrotor Flips*

1) Experimental Setup: In this experiment, we mounted the DVS on a quadrotor in a front-looking configuration. The quadrotor first hovers in front of a black square attached to a white wall. It then performs a flip around the optical axis of the DVS and settles down to hover condition again. Figure 12 shows the quadrotor performing a flip, when it reaches rotational speeds of up to 1,200 °/s.



(a) Image of the scene (b) Output of our algorithm

Fig. 11: Experiment 2: DVS moves at constant velocity parallel to the image shown in (a). The output of our algorithm captures many details (b). However, fine details are not preserved due to the low resolution of the DVS (128×128 pixels).

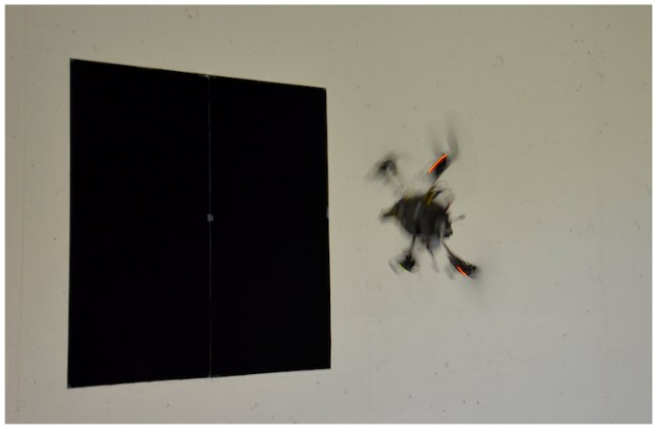


Fig. 12: Setup of Experiment 3: a quadrotor equipped with a front-looking DVS performs a flip in front of a square pattern. Rotational speeds are measured to be as high as 1,200 °/s during the flip.

2) Results: Figures 13 and 14 compare the output of both the naive and proposed methods at two different time instances during the experiment. Figure 13 shows the output during hovering, while Fig. 14 corresponds to the flip.

During hovering, a fixed accumulation interval of 1 ms hardly captures any structure (Fig. 13a), while 30 ms yields an almost sharp image (Fig. 13b). During the flip, an interval of 1 ms yields a sharp image of the square (Fig. 14a), while an interval of 30 ms causes heavy motion blur (Fig. 14b). Thus, choosing such a accumulation interval results in a trade-off between completeness of the image and motion blur, which is visible as “thickening”. In contrast, our method provides a sharp image in both situations, showing the applicability and adaptability of the algorithm to varying velocities in both rotational and translational motion.

D. Experiment 4: Urban Environment

Figure 1 shows an experiment in an urban environment. The scene constists of a window frame at a close distance with office buildings outside (cf. Fig. 1a). While the naive method either misses elements in the scene (the buildings, see Fig. 1c) or causes motion blur (the window frame, Fig. 1b), our method performs well in capturing both fast and slow edges (Fig. 1d).

V. Conclusion

We developed a method to augment the stream of events from a retinal camera with a measure of the *lifetime* of each event. Such a measure is defined based on the visual flow in the image plane and is computed based on a local planar approximation of the surface of active events. To this end, we designed an event-based, robust plane fitting algorithm with minimum latency (by considering only past events in the neighborhood of the current event) and optional regularization. In contrast to the previous work of visual-flow estimation, we did not rely on any temporal window or the use of future events.

The generated stream of augmented events opens up new possibilities to design other event-based algorithms that operate on a continuous-time representation of the events. This significantly departs from the algorithmic paradigm of event accumulation over artificially-chosen intervals of constant duration at discrete times, which suffer from the “completeness – motion blur” trade-off. We demonstrated the usefulness of our method with several experiments in a visualization application: the rendering of sharp gradient images at any time instant. Additionally, we included datasets acquired by a DVS onboard a quadrotor during agile maneuvers. Our method outperformed that of constant event-accumulation interval, which implicitly assigns the same lifetime to all events, since it can cope with scenes containing structures at different apparent velocities. In addition, it is able to filter a significant amount of events caused by sensor noise. Our method performs well despite the low resolution of the DVS (128×128 pixels).

References

[1] T. Delbruck, B. Linares-Barranco, E. Culurciello, and C. Posch, “Activity-driven, event-based vision sensors,” in *Intl. Conf. on Circuits and Systems (ISCAS)*, May 2010, pp. 2426–2429.

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

[3] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Regnier, “Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics,” *IEEE Trans. Robotics*, vol. 28, pp. 1081–1089, 2012.

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

[5] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, “Event-based visual flow,” *IEEE Trans. Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 407–417, 2014.

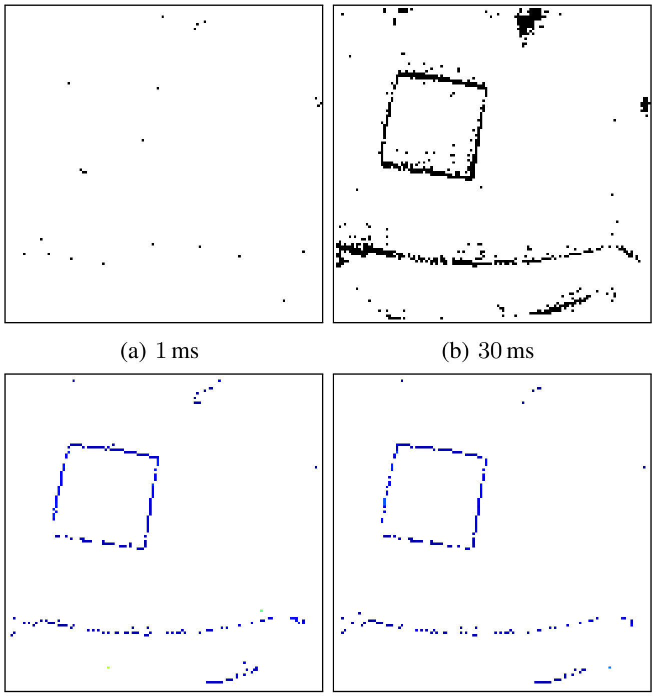
[6] A. Censi and D. Scaramuzza, “Low-latency event-based visual odometry,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[7] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison, “Simultaneous mosaicing and tracking with an event camera,” in *British Machine Vision Conf. (BMVC)*, 2014.

[8] D. Weikersdorfer and J. Conradt, “Event-based particle filtering for robot self-localization,” in *IEEE Intl. Conf. on Robotics and Biomimetics (ROBIO)*, 2012.

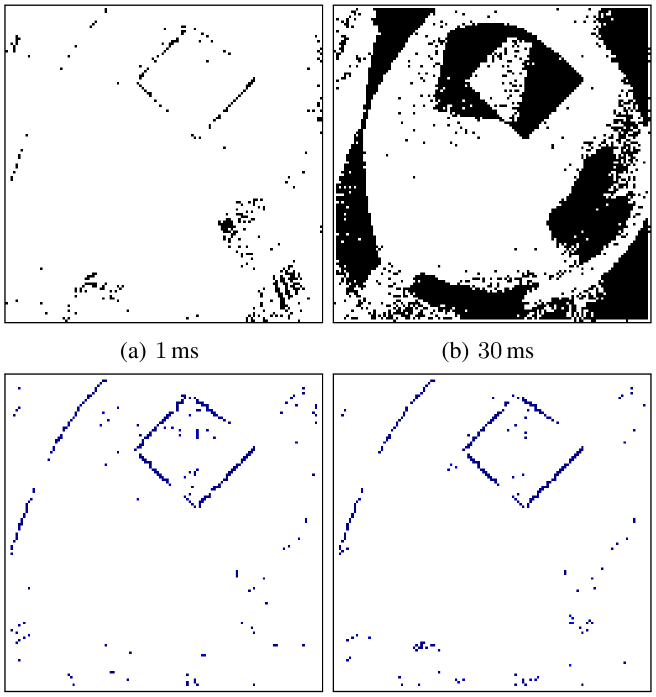
[9] E. Mueggler, B. Huber, and D. Scaramuzza, “Event-based, 6-DOF pose tracking for high-speed maneuvers,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[10] D. Weikersdorfer, R. Hoffmann, and J. Conradt, “Simultaneous localization and mapping for event-based vision systems,” in *Intl. Conf. on Computer Vision Systems (ICVS)*, 2013.


(a) 1 ms (b) 30 ms

(c) Without regularization (d) With regularization

Fig. 13: Experiment 3: Quadrotor during *hovering*. The experimental setup is shown in Fig. 12. A fixed accumulation interval of 1 ms captures hardly any structure (a), while an interval of 30 ms yields a sharp image (b). Our method produces even sharper images and reduces noise (c), (d).


(a) 1 ms (b) 30 ms

(c) Without regularization (d) With regularization

Fig. 14: Experiment 3: Quadrotor during the *flip*. The experimental setup is shown in Fig. 12. A fixed accumulation interval of 1 ms yields a sharp image (a), while an interval of 30 ms produces heavy motion blur (b). Our method produces sharp images and reduces noise (c), (d).

[11] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. Douglas, and T. Delbruck, "A pencil balancing robot using a pair of AER dynamic vision sensors," in *Intl. Conf. on Circuits and Systems (ISCAS)*, 2009.

[12] T. Delbruck and M. Lang, "Robotic goalie with 3ms reaction time at 4% CPU load using event-based dynamic vision sensor," *Frontiers in Neuroscience*, vol. 7, no. 223, 2013.

[13] J. Kogler, C. Sulzbachner, and W. Kubinger, "Bio-inspired stereo vision system with silicon retina imagers," in *Computer Vision Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5815, pp. 174–183.

[14] S. Schraml, A. Belbachir, N. Milosevic, and P. Schön, "Dynamic stereo vision system for real-time tracking," in *Intl. Conf. on Circuits and Systems (ISCAS)*, 2010.

[15] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[16] S.-C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current Opinion in Neurobiology*, vol. 20, no. 3, pp. 288–295, 2010.

[17] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion," *J. Opt. Soc. Am. A*, vol. 2, no. 2, pp. 284–299, 1985.