Computing Spatial Image Convolutions for Event Cameras

Cedric Scheerlinck[1], Nick Barnes[1,2], Robert Mahony[1]

Abstract— **Spatial convolution is arguably the most fundamental of 2D image processing operations. Conventional spatial image convolution can only be applied to a conventional image, that is, an array of pixel values (or similar image representation) that are associated with a single instant in time. Event cameras have serial, asynchronous output with no natural notion of an image frame, and each event arrives with a different timestamp. In this paper, we propose a method to compute the convolution of a linear spatial kernel with the output of an event camera. The approach operates on the event stream output of the camera directly without synthesising pseudo-image frames as is common in the literature. The key idea is the introduction of an internal state that directly encodes the convolved image information, which is updated asynchronously as each event arrives from the camera. The state can be read-off as-often-as and whenever required for use in higher level vision algorithms for real-time robotic systems. We demonstrate the application of our method to corner detection, providing an implementation of a Harris corner-response 'state' that can be used in real-time for feature detection and tracking on robotic systems.**

I. INTRODUCTION

Spatial image convolutions are a core pre-processing step in almost all robotic vision algorithms. For example, Gaussian smoothing, gradient computation, computation of the Laplacian, etc, are convolutional operations that underly fundamental vision algorithms such as: feature detection, optical flow computation, edge detection, etc. Classical image convolution requires a full image frame such as are generated by conventional synchronous cameras. Event cameras [1], [2] in contrast, provide asynchronous, data-driven measurements of grey-scale temporal contrast† at high temporal resolution and dynamic range. Event cameras have the potential to overcome many inherent limitations that conventional cameras display for robotic applications: motion blur in high-speed environments, under/overexposure in high-dynamic-range scenes, sparse temporal sampling (low frame rate), or very high bandwidth and data requirements (high frame rate). With such advantages, event cameras are an ideal embedded visual sensor modality for robotic systems [5]–[9]. However, the lack of a conventional image frame means that any image processing algorithm that relies on convolution cannot be directly applied to the output of an event camera.

In this paper, we propose a novel algorithm to compute the convolution of a linear kernel with the underlying radiometric scene information encoded by the output of an event camera. The key contribution of the paper is the introduction of an internal 'state' that encodes the convolved image information. Each pixel of the internal state carries a timestamp that encodes when the last event that updated that pixel arrived (surface of active events [10]), along with the latest state information, for example it could be values of: horizontal and vertical gradient, the Laplacian, or a Gaussian filtered intensity, etc.

The proposed algorithm uses continuous-time filter theory to compute a filtered or time-averaged version of the input event stream. Since spatial convolution is a linear process, it can be factored through the linear filter equations and applied directly to the event stream inputs. Thus, each event is spatially convolved with a linear kernel to generate a neighbouring collection of events, all with the same timestamp, which are then fed into pixel-by-pixel single-input-single-output continuous-time linear filters. The resulting filter equations can be solved explicitly, allowing asynchronous, discrete implementation of the continuous-time filter based on exact interpolation. Each asynchronous update of the internal state requires computation of one scalar exponential along with a small number of simple algebraic operations. The resulting algorithm does not require a motion-model for the camera and is truly asynchronous and highly efficient. Our method does not require reconstruction of pseudo-images, avoiding the latency and computational cost associated with synchronous reconstruction. The internal state can be separately read-off as-often-as and whenever required by a separate processing thread for use in higher level vision algorithms.

We demonstrate our approach using a variety of common kernels including Gaussian, Sobel and Laplacian kernels. To provide a more substantial example, we apply the method to the estimation of Harris corners. The approach taken is to augment the internal linear filter state with a (non-linear) Harris corner-response state. This 'state' is computed from the various gradients asynchronously as they are updated and provides a real-time measure of the Harris corner response of the underlying radiometric scene. The Harris corner state provides estimates of corners that we compare to a frame-based Harris detector, as well as state-of-the-art event-based corner detectors. We emphasise that in our algorithm no grey-scale image was required, or indeed is generated.

The remainder of the paper is as follows: Section II reviews related works. Section III outlines mathematical formulation and methodology. Section IV presents experimental results and analysis. Section V concludes the paper.

II. RELATED WORKS

Event cameras such as the DVS128 [1] and DAVIS240 [2] provide asynchronous, data-driven contrast events, and

This research was supported by the Australian Research Council through the "Australian Centre of Excellence for Robotic Vision" CE140100016

[1]The Australian National University

[2]Data61, CSIRO

†We consider temporal contrast events (not grey-level events [3], [4]).

are popular among roboticists due to their high temporal resolution and dynamic range, and low bandwidth and power consumption. One approach to computing spatial image convolutions with event cameras is to reconstruct image intensity [11]–[16] and apply 2D spatial convolutions to the output. [11]–[13] convert event-streams into image frames by taking either fixed temporal-windows of events, or a fixed number of events per frame. Scheerlinck et al. [15] introduce the concept of a continuous-time image state that encodes image intensity, and is updated asynchronously with each event. The approaches of [14]–[16] combine image frames with events to estimate image intensity that is available at the same temporal resolution as events. Computing convolutions in this manner, however, is unattractive as it incurs additional computational cost in the original image reconstruction as well as introducing noise, and imposes latency as the user has to 'wait' for the frame.

Event-driven gradient maps have been explored from a SLAM perspective [17]–[19] where the aim is to simultaneously estimate pose and a map. SLAM methods require pose estimation and depend also on a motion model for the camera. Moreover, to apply image convolution, the gradient output of these algorithms must be converted to intensity images (e.g. via Poisson solvers [20], [21]).

Alternative representations for event data such as surface of active events [10] or exponentially decaying time-surfaces [22], [23], warped-event counts [24]–[27], and plane-fitting [10], [28] have also proved useful for tasks such as recognition, motion estimation, and feature detection and tracking. Event-based corner detection algorithms [28]–[32] have been proposed that aim to detect corners on some form of time-surface [29]–[31] or intersections of planes fitted to events in space-time [28]. These methods represent state-of-the-art in event-based corner detection, though they are not designed for generalised spatial image convolutions.

Event cameras such as ATIS [3], and [4] are capable of providing absolute intensity with each event. Ieng et al. [33] propose a method to asynchronously compute spatial convolutions that relies on grey-level events (provided by the ATIS [3]), and show that beyond 3 frames per second, asynchronous convolutions outperform frame-based in computational cost. Huang et al. [34] propose an on-chip module that causes events to trigger neighbouring pixels for gradient computation. These methods rely on event cameras that are able to provide grey-level events. An alternative approach to event-based, 2D image filtering is the VLSI architecture proposed in [35], capable of implementing any convolutional kernel that is decomposable into x and y components.

III. METHOD

The proposed method is formulated as a parallel collection of continuous-time filters that are solved asynchronously as discrete updates using exact interpolation. That is we compute the exact analytic solution to the associated ordinary differential equation of the filter in continuous time and evaluate at discrete time instances.

A. Mathematical Representation and Notation

Each pixel in the event camera responds independently and asynchronously to changes in brightness. When the change in log intensity relative to the previous reference level exceeds a preset threshold c,

$$|\log(I) - \log(I_{\text{ref}})| > c, \tag{1}$$

an event is triggered and the pixel reference I_{ref} resets to the new brightness level. For contrast event cameras [1], [2], each event contains the time-stamp (t; relative to a global clock), discrete pixel address $\boldsymbol{p} = (x,y)^T$, and polarity ($\sigma = \pm 1$ depending on the sign of the brightness change).

$$\text{event}_i = (t_i, \boldsymbol{p}_i, \sigma_i), \quad i \in 1,2,3... \tag{2}$$

The output of an event camera is a serial stream of asynchronous events.

Events can be modelled as Dirac-delta functions [36]. Define an event $e_i(\boldsymbol{p},t)$ as

$$e_i(\boldsymbol{p},t) := \sigma_i\, c\, \delta(t - t_i)\, \delta_{\boldsymbol{p}_i}(\boldsymbol{p}), \tag{3}$$

where $\delta(t)$ is a Dirac-delta function and $\delta_{\boldsymbol{p}_i}(\boldsymbol{p})$ is a Kronecker delta function with indices associated with the pixel coordinates of $\boldsymbol{p}_i$ and $\boldsymbol{p}$. That is $\delta_{\boldsymbol{p}_i}(\boldsymbol{p}) = 1$ when $\boldsymbol{p} = \boldsymbol{p}_i$ and zero otherwise. In this paper we use the common assumption that the contrast threshold c is constant [11], [18], [19], although, in practice it does vary somewhat with intensity, event-rate and other factors [14]. The integral of events is

$$\int_0^t \sum_i e_i(\boldsymbol{p},\tau) = L(\boldsymbol{p},t) - L(\boldsymbol{p},0) + \int_0^t \eta(\boldsymbol{p},\tau) d\tau, \tag{4}$$

where $L(\boldsymbol{p},t)$ is the log intensity seen by the camera with initial condition $L(\boldsymbol{p},0)$, and $\eta(\boldsymbol{p},t)$ represents quantisation and sensor noise. $L(\boldsymbol{p},0)$ is typically unknown and $\eta(\boldsymbol{p},t)$ is unknown and poorly characterised. If left unchecked, integrated error arising from $\int_0^t \eta(\boldsymbol{p},\tau)d\tau$ grows over time and quickly degrades the estimate of $L(\boldsymbol{p},t)$ [14]. A method to deal with error arising from $L(\boldsymbol{p},0)$ and $\eta(\boldsymbol{p},t)$ will be presented in section III-C.

B. Event Convolutions

Let K denote a linear spatial kernel with finite support. Consider the convolution of K with $L(\boldsymbol{p},t)$. Define

$$L^K(\boldsymbol{p},t) := (K * L)(\boldsymbol{p},t). \tag{5}$$

Using (3), (4) and omitting the noise term $\eta(\boldsymbol{p},t)$ in the approximation

$$\begin{aligned} L^K(\boldsymbol{p},t) &\approx (K*L)(\boldsymbol{p},0) + \int_0^t \sum_i (K * e_i)(\boldsymbol{p},\tau) d\tau, \\ &\approx (K*L)(\boldsymbol{p},0) + \int_0^t \sum_i \sigma_i\, c\, \delta(t-t_i)\, (K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p}) d\tau, \\ &\approx (K*L)(\boldsymbol{p},0) + \int_0^t \sum_i e_i^K(\boldsymbol{p},\tau) d\tau, \end{aligned} \tag{6}$$

where

$$e_i^K(\boldsymbol{p},\tau) := \sigma_i\, c\, \delta(t - t_i)\, (K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p}). \tag{7}$$

Note that $(K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p})$ is a local spatial convolution of the finite support kernel K with a single non-zero image pixel. The result of such a convolution is an image with pixel values of zero everywhere except for a patch centred on $\boldsymbol{p}_i$ (the same size as K) with values drawn from the coefficients of K. The convolved event $e_i^K(\boldsymbol{p}, \tau)$ can be thought of as a finite (localised) collection of spatially separate events all occurring at the same time t_i.

C. Continuous-time Filter for Convolved Events

It is possible to compute the direct integral (6) using a similar approach to the direct integration schemes of [11], [14]. The drawback of this approach is integration of sensor noise, which results in drift, and undermines low temporal-frequency components of the estimate $L^K(\boldsymbol{p},t)$ over time. Furthermore, we are often concerned with high temporal-frequency information (i.e. scene dynamics), especially in robotic systems scenarios where the scene around the robot is changing continually. This leads us to consider a simple high-pass filtered version of $L^K(\boldsymbol{p},t)$.

Frequency domain. We design the high-pass filter in the frequency domain, and later implement it in the time domain via inverse Laplace transform. For $\alpha > 0$, a scalar constant, we define a high pass filter $F(s) := s/(s+\alpha)$ and apply it directly to the integrated event stream (6). Let $\mathscr{L}^K(\boldsymbol{p},s)$ denote the Laplace transform of the signal $L^K(\boldsymbol{p},t)$. Let $\hat{\mathscr{G}}(\boldsymbol{p},s)$ denote the high-pass filtered version of $\mathscr{L}^K(\boldsymbol{p},s)$. That is:

$$\begin{aligned}\hat{\mathscr{G}}(\boldsymbol{p},s) &:= \frac{s}{s+\alpha}\mathscr{L}^K(\boldsymbol{p},s),\\ &= \frac{s}{s+\alpha}\frac{1}{s}\sum_i \mathscr{E}_i^K(\boldsymbol{p},s) + \frac{s}{s+\alpha}\frac{1}{s}(K*L)(\boldsymbol{p},0),\\ &= \frac{1}{s+\alpha}\sum_i \mathscr{E}_i^K(\boldsymbol{p},s) + \frac{1}{s+\alpha}(K*L)(\boldsymbol{p},0), \qquad (8)\end{aligned}$$

where $\mathscr{E}_i^K(\boldsymbol{p},s) = \sigma_i c \exp(-t_i s)(K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p})$ is the Laplace transform of $e_i^K(\boldsymbol{p},t)$. The DC term associated with the unknown initial condition has an exponentially decreasing time-response $e^{-\alpha t}(K*L)(\boldsymbol{p},0)$ in the filter state and is quickly attenuated. The high-pass filter naturally attenuates low-frequency components of the noise signal $\eta(\boldsymbol{p},t)$.

Time domain. Ignoring the $(K*L)(\boldsymbol{p},0)$ initial condition, the time domain signal $\hat{G}(\boldsymbol{p},t)$ can be computed by taking the inverse Laplace of (8) and solving the resulting ordinary differential equation††

$$\frac{\partial}{\partial t}\hat{G}(\boldsymbol{p},t) = -\alpha \hat{G}(\boldsymbol{p},t) + \sum_i e_i^K(\boldsymbol{p},t), \qquad (9)$$

for each pixel $\boldsymbol{p}$. Here, $\hat{G}(\boldsymbol{p},t)$ is a pixel-by-pixel internal state that provides an estimate of the high-pass component of $(K*L)(\boldsymbol{p},t)$.

The continuous-time differential equation (9) is a constant coefficient linear differential equation except at time instances when an event occurs and can be solved explicitly. To exploit this property we store the timestamp of the latest event at each pixel $t^{\boldsymbol{p}}$ and use the explicit solution of (9) to asynchronously update the state when (and only when) a new event at that pixel occurs.

The constant-coefficient, first-order ODE for (9) assuming no events is

$$\frac{\partial}{\partial t}\hat{G}(\boldsymbol{p},t) = -\alpha \hat{G}(\boldsymbol{p},t). \qquad (10)$$

Let t_i denote the timestamp of the current event and denote the limit to t_i from below by t_i^- and the limit to t_i from above by t_i^+. Integrate (10) from $t^{\boldsymbol{p}}$ (the timestamp of the previous event at $\boldsymbol{p}$) to t_i^-

$$\hat{G}(\boldsymbol{p},t_i^-) = \exp(-\alpha(t_i - t^{\boldsymbol{p}}))\hat{G}(\boldsymbol{p},t^{\boldsymbol{p}}). \qquad (11)$$

Next integrate (9) over the convolved event, i.e. from t_i^- to t_i^+

$$\int_{t_i^-}^{t_i^+} \frac{\partial}{\partial t}\hat{G}(\boldsymbol{p},t)dt = \int_{t_i^-}^{t_i^+} -\alpha \hat{G}(\boldsymbol{p},t) + e_i^K(\boldsymbol{p},t)dt.$$

The integral of the right-hand side is $\sigma_i c (K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p})$ since $\hat{G}(\boldsymbol{p},t)$ is continuous and its infinitesimal integral is zero, and the Dirac delta integrates to unity. Thus, one has

$$\hat{G}(\boldsymbol{p},t_i^+) = \hat{G}(\boldsymbol{p},t_i^-) + \sigma_i c (K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p}). \qquad (12)$$

In addition, it is necessary to update the timestamp state

$$t^{\boldsymbol{p}} = t_i. \qquad (13)$$

Equations (10), (12) and (13) together define an asynchronous distributed update that can be applied pixel-by-pixel to compute the filter state. The state could also be updated at any user-chosen time-instance (for example just before a read-out) with the time-instance stored in $t^{\boldsymbol{p}}$.

Multiple different filters can be run in parallel. For example, if gradient estimation is required, then two filter states $(\hat{G}_x, \hat{G}_y)$ can be run in parallel for the x and y components using an appropriate directional kernels (Sobel, central difference, etc).

IV. RESULTS

The experiments were performed using a DAVIS240C [2] with default biases provided in the jAER software, and sequences from [15], [36]. The internal filter state of the system is asynchronous and for visualisation we display instantaneous snap shots taken at sample times. There is only a single parameter α in the filter. We set $\alpha = 2\pi\,\text{rad/s}$ for all sequences unless otherwise stated. The complexity of our algorithm scales linearly with the number of (non-zero) elements in the kernel, and we find that a kernel size of 3×3 is usually sufficient. We fix the contrast threshold c constant.

A. Event Convolutions

Figure 1 displays a range of different filtered versions of an input sequence (`sun` and `night_drive` are taken from [15]). The identity can be seen as a reconstruction of log intensity. It can be observed that the direct event filtered outputs are equivalent to applying the filter to the

††Although we write this as a partial differential equation (the partial taken with respect to time) there is no coupling between pixel locations and the solution decouples into parallel pixel-by-pixel ODEs.

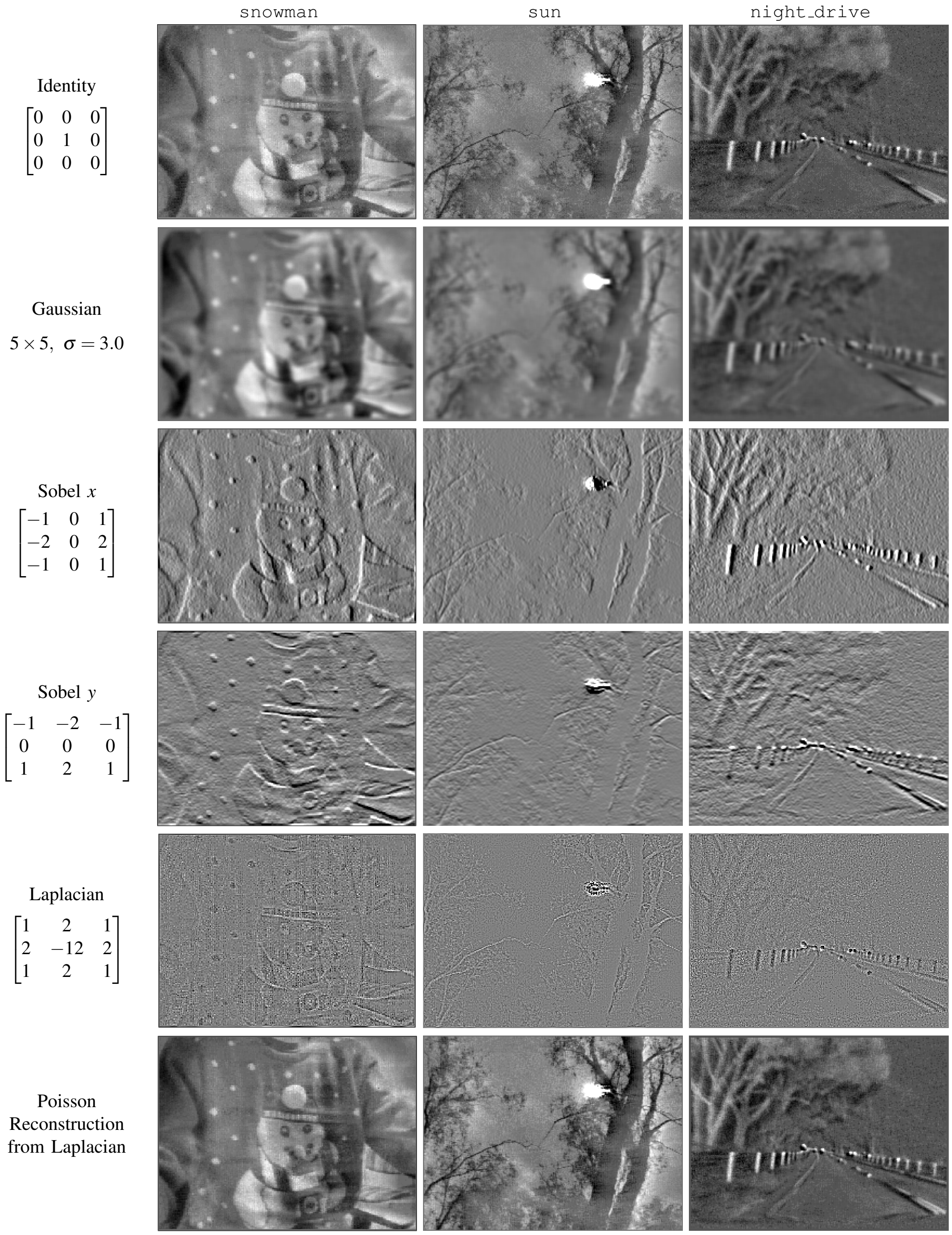

Fig. 1. Different kernels K applied to events using high-pass filter (9). `Sun` demonstrates robustness in extreme dynamic range scenarios and `night_drive` is captured in pitch black conditions, demonstrating excellent performance in low-light settings thanks to the event camera.

reconstruction. The key advantage is that direct event filtering does not incur latency or additional computation associated with reconstruction. The sequences in Fig 1 are:

- `Snowman` *(left)*: The first author wearing a knitted jumper with prominent snowman and snowflakes design (taken under normal office conditions).
- `Sun` *(centre)*: Looking directly at the sun through the trees. Exemplifies high dynamic range performance of the camera.
- `Night_drive` *(right)*: Country road at night with no street lights or ambient lighting, only headlights. The car is travelling at 80km/h causing considerable motion in the scene. Exemplifies performance in high-speed, low-light conditions.

The first row of Figure 1 shows the application of the identity kernel. This kernel returns a (temporal) high-pass filtered version of the original image.

Despite noise in the event stream, our approach reproduces a high-fidelity representation of the scene. It is particularly interesting to note the response for the two challenging sequences `sun` and `night_drive`. In both cases the image is clear and full of detail, despite the high dynamic range of

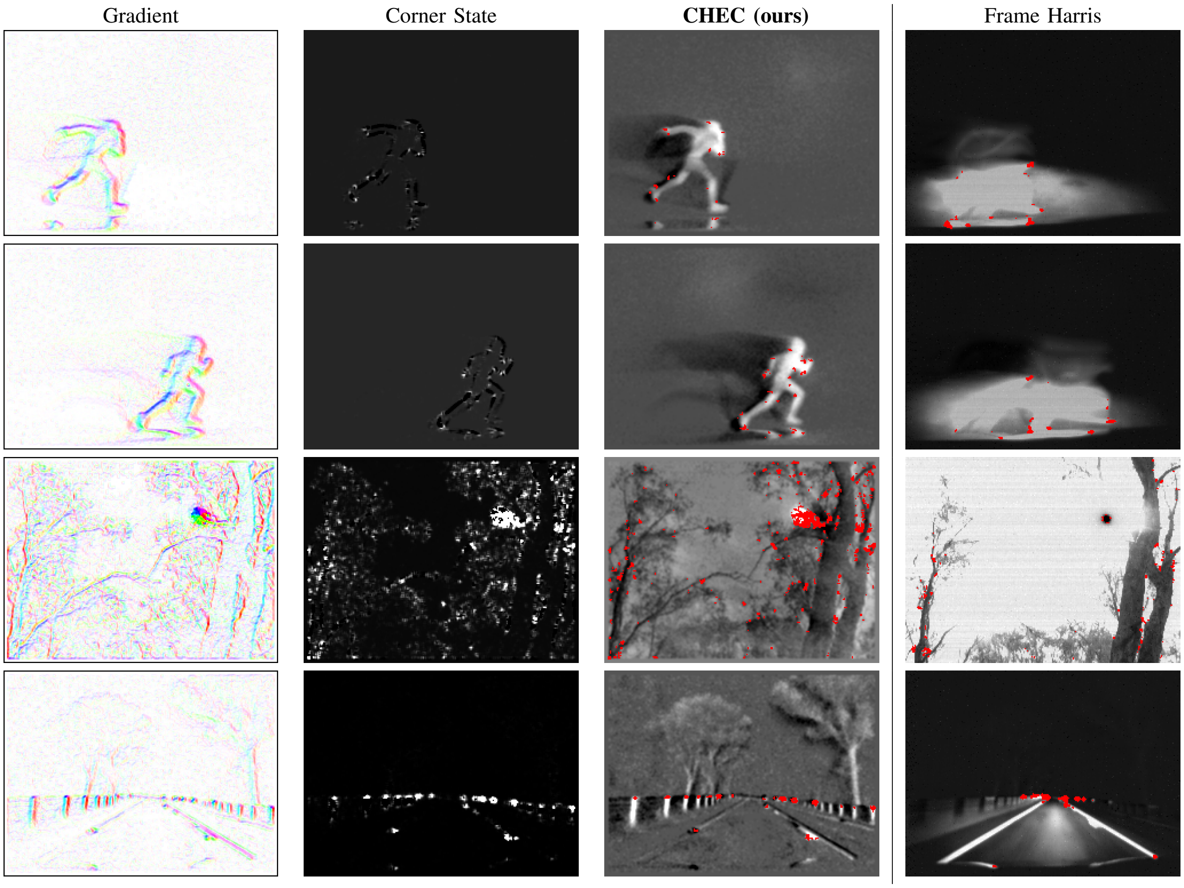

Fig. 2. Harris corner-response: events only vs. conventional camera. Gradient shows a snap shot of the internal gradient state, obtained by applying Sobel x and y kernels directly to events and high-pass filtering (9). Corner State shows a snap shot of the full Harris corner-response state (15) computed from only the gradient state. CHEC (ours) shows the corner-response thresholded at a suitable value and superimposed onto a log intensity image, obtained via Poisson reconstruction [20], [21] of Gradient. Note: log intensity is displayed purely for visualisation and is not used to compute corners. Frame Harris shows the Harris detector [37] applied to raw image frames for comparison.

the scene.

The second row computes a (spatial) low pass Gaussian filter of the sequences. The low pass nature of the response is clear in the image. The authors note that if it was desired to compute an image pyramid then it is a straightforward generalisation of the filter equations to reduce the state dimension at a particular level of the image pyramid by linear combination of pixel values. The resulting filter would still be linear and the same filter equations would apply.

The third and fourth rows display the internal filter state for the Sobel kernels in both vertical and horizontal directions. The results show that the derivative filter state is operating effectively even in very low light and high dynamic range conditions.

Rows five and six display the Laplacian of the image (the sum of second derivatives of the image) and a Poisson reconstruction built from the Laplacian image. The Laplacian kernel computes an approximation of the divergence of the gradient vector field. It can be used for edge detection: zero crossings in the Laplacian response correspond to inflections in the gradient and denote edge pixels. It is also possible to reconstruct an original (log) intensity image from a Laplacian image using Poisson solvers [20], [21]. In this case, we present the Poisson reconstruction of the Laplacian image (row six) primarily to verify the quality of the filter response.

It is important to recall that the internal state of the filter is computed directly from the event stream in all these cases. Thus, if only the Laplacian is required then the filter will not compute a grey scale image or gradient image.

B. Continuous Harris Event Corners (CHEC)

To demonstrate a practical application of our filter, we compute a continuous-time 2D state that encodes the Harris-corner-response [37] of pixels. We call our method *Continuous Harris Event Corners* (CHEC). This can be used in a feature detection and tracking algorithm on a robotic system. In Figs. 2, 3, it can be seen that CHEC corresponds to real corners in the image. In difficult conditions, our approach clearly outperforms the conventional camera, which is subject to significant under/overexposure and blur.

The Harris corner detector [37] depends on image gradients that we compute using the filter architecture proposed in equations (10), (12) and (13) for Sobel kernels K_x and K_y. A key advantage of our approach is that we are able to update the corner-response state asynchronously with each event, rather than having to synchronously update the entire state. The corner-response depends on a spatial region of

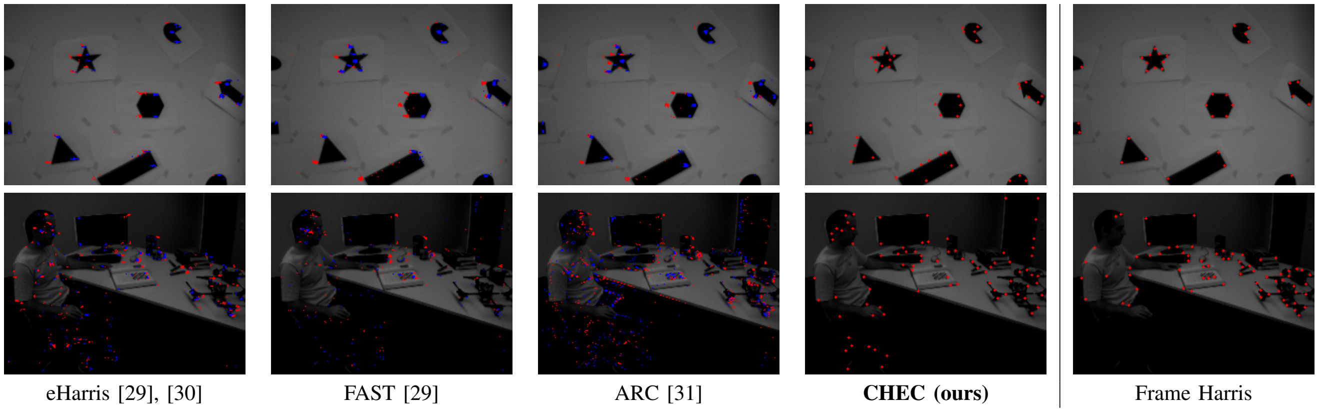

Fig. 3. Top: `shapes`, bottom: `dynamic`. We plot the raw camera frame for visualisation, but it is not used in any event-based corner detection (eHarris, FAST, ARC, CHEC). Corners appear shifted due to the low temporal resolution of the raw frame. Since eHarris, FAST and ARC provide asynchronous positive (blue) and negative (red) corner events, we accumulate the last 30ms for visualisation. Frame Harris computes corners from the raw frames, which are subject to low temporal resolution and limited dynamic range.

TABLE I
FRACTION OF CORNER MATCHES BETWEEN METHODS

`shapes`	Frame Harris [37]	eHarris [29], [30]	CHEC (ours)
F. Harris	1	0.36	**0.46**
eHarris	0.36	1	0.30
CHEC	0.46	0.30	1
`dynamic`	F. Harris	eHarris	CHEC
F. Harris	1	0.16	**0.22**
eHarris	0.16	1	0.15
CHEC	0.22	0.15	1

image gradient, so for each event we update the state in a local patch to incorporate all changes.

Let $\hat{G}(\boldsymbol{p},t) = \begin{bmatrix}\hat{G}_x(\boldsymbol{p},t) & \hat{G}_y(\boldsymbol{p},t)\end{bmatrix}^\top$ denote an internal gradient state (9).

The Harris matrix is

$$M(\boldsymbol{p},t) := W * \hat{G}(\boldsymbol{p},t)\hat{G}(\boldsymbol{p},t)^T. \tag{14}$$

where W is a smoothing kernel, e.g. box or Gaussian.

The Harris corner-response [37] is

$$R(\boldsymbol{p},t) := \det(M(\boldsymbol{p},t)) - \gamma \operatorname{trace}(M(\boldsymbol{p},t))^2, \tag{15}$$

where γ is an empirically determined constant, in this case $\gamma = 0.04$.

We emphasise, that the Harris response $R(\boldsymbol{p},t)$ is updated locally whenever an event updates a pixel. A significant saving in computational cost, since it is only where events are occurring that corners occur. Corners can be extracted from the Harris response using thresholding or non-maximum suppression.

Figure 2 shows the continuous-time Harris corner-response state (15) computed from the gradient state estimate (9), on real sequences from [15]. We emphasise that the image intensity was not required and not computed. Log intensity (obtained via post processing of the stored gradient state and Poisson reconstruction) is displayed purely for visualisation purposes. `Night_run` (top row Fig 2) is captured in pitch black conditions as someone runs in front of headlights of a car. The conventional camera suffers extreme motion blur because of the high exposure-time required in low-light conditions. Our approach leverages the high-dynamic-range, high-temporal-resolution event camera, yielding crisp edges and corners. `Sun` (third row) displays artifacts in the corner state around the sun because of extreme brightness gradients caused by the high-dynamic range of the sun on a cloudless day, where the event camera is pushed to the limit. Nevertheless, we still get clear corners around the branches and leaves of the trees, whereas the conventional camera frame is largely over-saturated. `Night_drive` (last row) demonstrates performance under challenging high-speed, low-light conditions. Our approach clearly detects corners on roadside poles and road-markings. The conventional camera frame is highly blurred and unable to detect corners in much of the image.

Figure 3 compares state-of-the-art event-based corner detectors [29]–[31], as well as frame-based Harris detector (Frame Harris) [37], against our proposed method (CHEC), on real sequences from the event camera dataset [36] (`shapes_translation` and `dynamic_6dof`). eHarris was first developed by [30], and later improved by [29]. We use improved eHarris code implementation of [29]. We also compare against FAST event-based corner detector [29] and ARC (asynchronous event-based corner detection) [31]. For state-of-the-art we use default parameters provided in the open-source code. For CHEC, we increase the filter gain to $\alpha = 10\,\text{rad/s}$ to reduce low-temporal-frequency noise. To extract corners from the Harris response of both Frame Harris and CHEC, we first threshold, then apply non-maxmimum suppression. In simple low-texture environments (such as `shapes`) each method performs well, whereas in high-texture environments (`dynamic`), state-of-the-art event-based detectors find many spurious corners.

Table I displays quantitative evaluation and comparison of frame- and event-based Harris detectors. We compute the fraction of matching corners between Frame Harris, eHarris

[29], [30] and CHEC (ours). To make the comparison as fair as possible, we select frames with reasonable (slow) camera motions, and use eHarris corner-events occurring within the last 30ms of each camera frame. Since there are still many corner-events per frame, we accumulate corner-events into a binary image and extract the centroids of each blob. We impose the condition that corners within three pixels of each other count as a match. The highest levels of agreement are between Frame Harris and our proposed method CHEC.

V. CONCLUSION

We have introduced a method to compute spatial convolutions for contrast event cameras. A key feature is the continuous-time internal state that encodes convolved image information and allows asynchronous, event-driven, incremental updates. We extend the concept of an internal state to a Harris corner-response state, and demonstrate corner detection (CHEC) without requiring intensity. We believe there are many exciting possibilities in this direction, including alternative feature states, continuous-time optical flow state, and application of event-based convolutions to convolutional neural networks.

REFERENCES

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

[2] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240x180 130dB 3us latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[3] 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.

[4] J. Huang, M. Guo, and S. Chen, "A dynamic vision sensor with direct logarithmic output and full-frame picture-on-demand," in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.

[5] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza, "EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real-time," *IEEE Robot. Autom. Lett.*, vol. 2, pp. 593–600, 2017.

[6] E. Mueggler, G. Gallego, and D. Scaramuzza, "Continuous-time trajectory estimation for event-based vision sensors," in *Robotics: Science and Systems (RSS)*, 2015.

[7] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-DOF pose tracking for high-speed maneuvers," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2014, pp. 2761–2768.

[8] A. Censi and D. Scaramuzza, "Low-latency event-based visual odometry," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 703–710.

[9] A. Rosinol Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? combining events, images, and IMU for robust visual SLAM in HDR and high speed scenarios," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 994–1001, Apr. 2018.

[10] 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, 2014.

[11] C. Reinbacher, G. Graber, and T. Pock, "Real-time intensity-image reconstruction for event cameras using manifold regularisation," in *British Machine Vis. Conf. (BMVC)*, 2016.

[12] S. Barua, Y. Miyatani, and A. Veeraraghavan, "Direct face detection and video reconstruction from event cameras," in *IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, 2016, pp. 1–9.

[13] P. Bardow, A. J. Davison, and S. Leutenegger, "Simultaneous optical flow and intensity estimation from an event camera," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2016, pp. 884–892.

[14] C. Brandli, L. Muller, and T. Delbruck, "Real-time, high-speed video decompression using a frame- and event-based DAVIS sensor," in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, June 2014, pp. 686–689.

[15] C. Scheerlinck, N. Barnes, and R. Mahony, "Continuous-time intensity estimation using event cameras," in *Asian Conf. Comput. Vis. (ACCV)*, 2018.

[16] L. Pan, C. Scheerlinck, X. Yu, R. Hartley, M. Liu, and Y. Dai, "Bringing a blurry frame alive at high frame-rate with an event camera," *arXiv preprint arXiv:1811.10180*, 2018.

[17] M. Cook, L. Gugelmann, F. Jug, C. Krautz, and A. Steger, "Interacting maps for fast visual interpretation," in *Int. Joint Conf. Neural Netw. (IJCNN)*, 2011, pp. 770–776.

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

[19] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," in *Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 349–364.

[20] A. Agrawal, R. Chellappa, and R. Raskar, "An algebraic approach to surface reconstruction from gradient fields," in *Int. Conf. Comput. Vis. (ICCV)*, 2005, pp. 174–181.

[21] A. Agrawal, R. Raskar, and R. Chellappa, "What is the range of surface reconstructions from a gradient field?" in *Eur. Conf. Comput. Vis. (ECCV)*, 2006, pp. 578–591.

[22] X. Lagorce, G. Orchard, F. Gallupi, B. E. Shi, and R. Benosman, "HOTS: A hierarchy of event-based time-surfaces for pattern recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 39, no. 7, pp. 1346–1359, July 2017.

[23] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "HATS: Histograms of averaged time surfaces for robust event-based object classification," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018, pp. 1731–1740.

[24] T. Stoffregen and L. Kleeman, "Simultaneous optical flow and segmentation (SOFAS) using dynamic vision sensor," in *Australasian Conf. Robot. Autom. (ACRA)*, 2017.

[25] G. Gallego, H. Rebecq, and D. Scaramuzza, "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018, pp. 3867–3876.

[26] G. Gallego and D. Scaramuzza, "Accurate angular velocity estimation with an event camera," *IEEE Robot. Autom. Lett.*, vol. 2, pp. 632–639, 2017.

[27] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017, pp. 4465–4470.

[28] X. Clady, S.-H. Ieng, and R. Benosman, "Asynchronous event-based corner detection and matching," *Neural Netw.*, vol. 66, pp. 91–106, 2015.

[29] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection," in *British Machine Vis. Conf. (BMVC)*, 2017.

[30] V. Vasco, A. Glover, and C. Bartolozzi, "Fast event-based Harris corner detection exploiting the advantages of event-driven cameras," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2016.

[31] I. Alzugaray and M. Chli, "Asynchronous corner detection and tracking for event cameras in real time," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3177–3184, Oct. 2018.

[32] A. Glover and C. Bartolozzi, "Robust visual tracking with a freely-moving event camera," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Sept. 2017, pp. 3769–3776.

[33] S.-H. Ieng, C. Posch, and R. Benosman, "Asynchronous neuromorphic event-driven image filtering," *Proc. IEEE*, vol. 102, no. 10, pp. 1485–1499, Oct. 2014.

[34] J. Huang, M. Guo, S. Wang, and S. Chen, "A motion sensor with on-chip pixel rendering module for optical flow gradient extraction," in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[35] T. Serrano-Gotarredona, A. G. Andreou, and B. Linares-Barranco, "Aer image filtering architecture for vision-processing systems," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 46, no. 9, pp. 1064–1071, 1999.

[36] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Robot. Research*, vol. 36, pp. 142–149, 2017.

[37] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Fourth Alvey Vision Conf.*, vol. 15, 1988, pp. 147–151.