VibES: Induced Vibration for Persistent Event-based Sensing

Vincenzo Polizzi[1] Stephen Yang[1] Quentin Clark[2]
Jonathan Kelly[1] Igor Gilitschenski[2] David B. Lindell[2]

[1]University of Toronto, Robotics Institute
[2]University of Toronto, Department of Computer Science
{vincenzo.polizzi, jonathan.kelly}@robotics.utias.utoronto.ca stephy.yang@mail.utoronto.ca
{qtcc, gilitschenski, lindell}@cs.toronto.edu

Abstract

Event cameras are a bio-inspired class of sensors that asynchronously measure per-pixel intensity changes. Under fixed illumination conditions in static or low-motion scenes, rigidly mounted event cameras are unable to generate any events, becoming unsuitable for most computer vision tasks. To address this limitation, recent work has investigated motion-induced event stimulation that often requires complex hardware or additional optical components. In contrast, we introduce a lightweight approach to sustain persistent event generation by employing a simple rotating unbalanced mass to induce periodic vibrational motion. This is combined with a motion-compensation pipeline that removes the injected motion and yields clean, motion-corrected events for downstream perception tasks. We demonstrate our approach with a hardware prototype and evaluate it on real-world captured datasets. Our method reliably recovers motion parameters and improves both image reconstruction and edge detection over event-based sensing without motion induction.

1. Introduction

Most computer vision algorithms rely on conventional image sensors — such as RGB or grayscale cameras — that capture synchronous image frames at fixed intervals. While widely adopted, these frame-based sensors suffer from well-known limitations, including motion blur, latency due to frame timing, and high power consumption [19]. These drawbacks become particularly problematic in robotics applications where fast dynamics, low latency, and energy efficiency are critical, such as in autonomous navigation [41], feature tracking [32], and real-time scene understanding [27].

Event cameras have emerged as a promising alternative. These bio-inspired sensors asynchronously detect per-pixel changes in log-intensity, producing a sparse stream of events with microsecond latency and high dynamic range [19]. Event cameras inherently avoid motion blur, operate at lower power [20], and enable low-latency perception, making them well suited for resource-constrained, high-speed robotics. Recent advances in event-based feature tracking and visual odometry have demonstrated strong performance in dynamic scenarios [32, 45], further highlighting their potential for robust autonomy.

However, a fundamental limitation persists: event generation depends on motion. In static scenes — or when edges align with the direction of the camera motion — events are not triggered. This results in event sparsity, loss of spatial information, and perceptual fading [7], which undermines long-term feature tracking, edge detection, and high-quality image reconstruction [24]. Consequently, the performance of event-based systems degrades significantly without sufficient camera or scene motion.

Interestingly, the human visual system faces a similar challenge. To prevent image fading during fixation, our eyes perform rapid, involuntary movements known as microsaccades or fixational eye movements (FEMs). These movements continuously stimulate photoreceptors, refreshing the visual input and maintaining high-resolution perception [28, 47]. This biological mechanism suggests that artificial motion could be exploited to sustain event generation.

In this work, we draw inspiration from microsaccades while adopting a deterministic, engineered approach. Instead of attempting to replicate their stochastic nature, we design a lightweight vibration mechanism that mechanically stimulates the event camera, ensuring continuous event generation in static or quasi-static scenes. Our approach, VIBES, uses a simple rotating unbalanced mass within a spring–damper system to induce harmonic motion. A motion-compensation pipeline then removes the injected vibration, producing clean, motion-corrected events

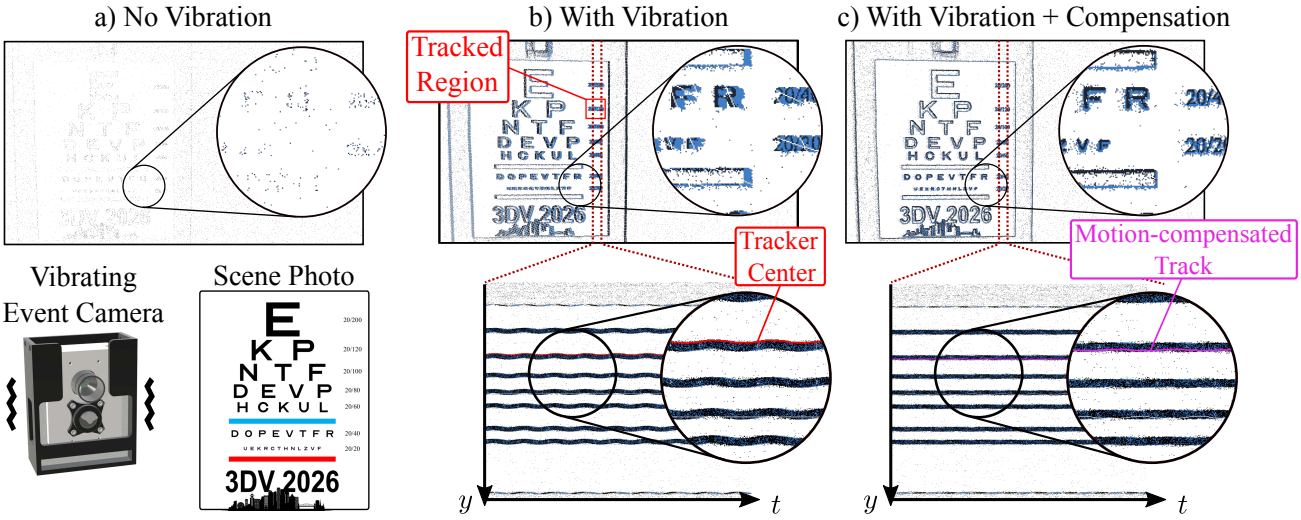


Figure 1. **Qualitative illustration of our method. No Vibration.** With a static event camera and no induced motion, the accumulated event image appears blurred and lacks sharp edges, while the y–t slice shows little temporal structure. **With Vibration.** Introducing controlled vibrations stimulates the sensor, increasing the number of events and producing sinusoidal traces in the y–t slice. We develop an Extended Kalman Filter (EKF) to track the vibrational motion in a region of the scene (red). **With Vibration + Compensation.** We use the EKF-tracked region to estimate and compensate for the sinusoidal motion across the entire scene. The motion-compensated accumulated image recovers sharp structures, and the y–t slice aligns with the stable motion-compensated track (magenta), revealing the underlying scene. An inset (bottom left) shows the reference scene displayed in front of the camera with induced motion.

for downstream tasks. Compared to existing microsaccade-inspired methods based on optical systems [24] or pan–tilt actuators [13, 34, 43, 44, 50], our design requires no additional optical components or position-encoding sensors, enables real-time motion compensation on the event stream, and extends to tracking motion frequencies in a target scene. An overview of the proposed method is shown in Fig. 1.

Our contributions are threefold:

- We design a vibrating event camera using a rotating unbalanced mass that enables event generation in static scenes, addressing a key limitation of event cameras.
- We introduce a real-time motion-compensation pipeline that estimates and removes induced vibrations online, without requiring calibration or prior knowledge of physical parameters.
- We validate our method on four real-world datasets, demonstrating improved event density and higher-quality results for edge detection and image reconstruction. We also show extensions to specialized applications such as scene frequency estimation and relative depth prediction.

2. Related Work

Event cameras are increasingly explored in robotics and computer vision for their ability to capture scene dynamics with high temporal precision. In this section, we review their role in perception, outline complementary hybrid sensing strategies, and discuss microsaccade-inspired methods for maintaining event generation in otherwise static scenes.

Event cameras in robotics. Event cameras have gained significant attention in robotics and computer vision communities due to their unique capabilities, including high dynamic range, low latency, and motion-blur-free imaging. Recent methods have exploited these properties for feature tracking in challenging conditions [21, 32, 51].

Common applications of event cameras in robotics include visual localization [46], edge detection [5], and motion compensation [18]. Due to their high temporal resolution, frequency estimation with event cameras has also been studied extensively [3, 37, 42]. Other notable applications include image reconstruction [23, 26, 45] and image deblur [25, 36]. We show that for motion compensation, edge detection and image reconstruction tasks, our method can provide a high-quality information stream, taking advantage from our induced motion.

Hybrid-modality sensing methods. Beyond microsaccades, other strategies for event generation include moving cameras at high speeds for 3D reconstruction [29], pairing event cameras with conventional frames for static localization [11], and employing visual tracking to follow objects [22]. Machine learning-based approaches have also been developed to infer object motion from sparse event data [6], with some relying on continuous-motion assumptions, such as in human choreography capture applications [49]. Our system does not rely on conventional image frames or on the assumption of fast motion in the scene. Instead, we propose a fully model-based event-centric ap-

proach that exploits the intrinsic properties of events and their sensitivity to motion, generating a continuous stream of information. This stream can be leveraged for reconstruction, edge detection, and the tasks discussed above, even in otherwise static scenes.

Microsaccade-inspired event cameras. Inspired by human fixational eye movements (FEMs), researchers have explored creating both saccadic and non-saccadic artificial FEMs to address the limitations of event cameras in static scenes. Proposed mechanisms include pan–tilt tables [13, 34, 43, 44, 50], mirrors [30], a rotating wedge-prism system [24], and a rotating quarter-wave plate in front of the camera [31]. However, these approaches require precisely constructed mechanical systems combined with per-system calibration. Our mechanical system is comparatively simple, and our processing pipeline can automatically adjust to different system parameters. This makes our approach more appealing for real-world use, where systems that require careful tuning are less favourable than those that adjust parameters online. Concurrent work has attempted to recover motion from unstructured camera jitter emulating spacecraft vibration [2], using a clustering-based algorithm. However, the absence of a prior on the camera's jitter reduces reconstruction accuracy, leading to errors of several pixels in image space.

Most similar to our approach is that of Botao et al. [24], which proposes a rotating wedge prism to generate events continuously and utilizes a position-encoding hardware to track and remove the induced motion, thereby stabilizing texture and enhancing information output. Our method introduces a mechanically simpler approach, requiring no custom optical elements or position encoders such as in [24]. Instead, we directly estimate the state of the sinusoidal motion from the event stream itself and perform parameter estimation online. Moreover, we demonstrate that our software stack is agnostic to the underlying hardware. On data generated by the rotating wedge-prism in [24], we show that our system performs equally well in terms of motion compensation quality and event stream consistency. Finally, our experiments show that VIBES enables the same downstream use cases proposed in [24] — particularly in image reconstruction and edges extraction — with simpler hardware and a more general motion-tracking framework.

3. Methodology

Our approach aims to stimulate the pixels of an event camera to produce a continuous stream of events, enabling high-quality image reconstruction and sharp edge detection from raw event data. We first describe the mechanical system responsible for generating vibrational motion and its effect on event generation in the camera (Sec. 3.1). We then present the motion compensation pipeline that allows us to recover the underlying scene structure (Sec. 3.2). Finally, we describe the physical prototype implementation of our system (Sec. 3.3).

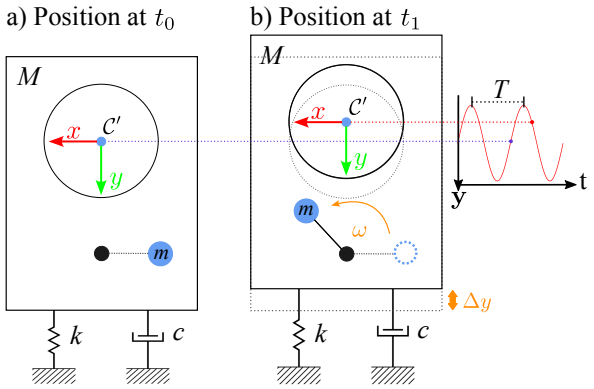


Figure 2. **Schematic of the mass–spring–damper model**. Camera setup at two time steps, t_0 a) and t_1 b). The rotation of an off-axis mass m with angular velocity ω induces planar displacements $\Delta x, \Delta y$ (only the vertical component is illustrated). The estimated oscillation frequency, $\frac{2\pi}{T}$, corresponds to the motion perceived by the camera. Note that this differs from the natural frequency of the off-axis mass ω, due to inertial effects.

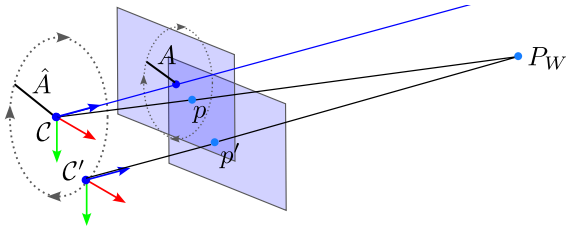


Figure 3. **Visual representation of the camera model**. The virtual camera $\mathcal{C}$ remains static, while the real camera $\mathcal{C}'$ translates along a circular path centered at $\mathcal{C}$ in the x–y image plane of $\mathcal{C}$. We denote $\mathbf{p}$ and $\mathbf{p}'$ as the projected point of $\mathbf{P}_W$ in the virtual and and real image planes respectively. $\hat{A}$ and A refer to the amplitudes of motion in the virtual and real frames respectively. The objective is to remove the resulting oscillatory motion and recover the projection of point P_W onto the image plane of $\mathcal{C}$.

3.1. Camera Motion Model

To characterize the motion induced by our vibration mechanism, we adopt a classical mass-spring-damper model, illustrated in Fig. 2. This model captures the essential dynamics of the periodic motion imparted to the camera.

We model our system as a forced damped harmonic oscillator, which can be described by a second-order differential equation. The closed-form steady-state solution is of the form

$$y(t) = \hat{A} \cdot \sin(\hat{\omega} t - \phi), \tag{1}$$

where $\hat{A} = \sqrt{\hat{A}_X^2 + \hat{A}_Y^2}$ is the oscillation amplitude, with $\hat{A}_X$ and $\hat{A}_Y$ as the components along $\mathbf{x}$- and $\mathbf{y}$-axis in cam-

era coordinates. As such, the motion of the camera when viewing a static scene can be expressed sinusoidally. Further details are provided in Supplementary Sec. S1.1.

The camera motion is described by introducing a *virtual camera frame*, denoted as $\mathcal{C}$, shown in Fig. 3. In our formulation, $\mathcal{C}$ remains static and acts as the reference frame, while the physical camera, $\mathcal{C}'$, moves along a controlled, circular trajectory in the x–y image plane of $\mathcal{C}$.

We adopt a standard pinhole projection model to describe how a 3D point $\mathbf{P}_W$ in the world frame $\mathcal{W}$ projects onto the image plane of $\mathcal{C}$. We denote $\mathbf{p} = (u, v, 1)$ as the projected point in the virtual frame, which is the point we would like to recover. The moving camera observes point $\mathbf{p}' = (u', v', 1)$. The transformation from $\mathcal{C}$ to the moving camera $\mathcal{C}'$ is defined as

$$\mathbf{T}(t)_{C'C} = \begin{bmatrix} & \hat{A}_X \cos(\hat{\omega}t + \phi_X) \\ \mathbf{I} & \hat{A}_Y \cos(\hat{\omega}t + \phi_Y) \\ & 0 \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{2}$$

where $\mathbf{I}$ is the 3×3 identity matrix, $\hat{\omega}$ is the angular frequency of the motion, and ϕ_X, ϕ_Y are the corresponding phase shifts. Note that $\hat{\omega}$ differs from ω in Fig. 3 due to inertial factors. The $\mathbf{z}$-coordinate remains constant, as the motion is planar. The projection of $\mathbf{P}_W$ onto the moving camera $\mathcal{C}'$ is then

$$\mathbf{p}' = \mathbf{K}\mathbf{T}(t)_{C'C}\mathbf{T}_{CW}\mathbf{P}_W. \tag{3}$$

Here, $\mathbf{K}$ is the intrinsic matrix and $\mathbf{T}_{CW}$ is the extrinsic transform that expresses $\mathbf{P}_W$ in $\mathcal{C}$. To illustrate this relation more concretely, consider the vertical coordinate v of the projected point $\mathbf{p}$. In the x–y image plane, the point $\mathbf{p}$ travels along a circular path with amplitude $A = \sqrt{A_x^2 + A_y^2}$. Then expanding Eq. 3 we obtain

$$v' = f\frac{Y}{Z} + f\frac{\hat{A}_Y \cos(\hat{\omega}t + \phi_Y)}{Z} + c_Y, \tag{4}$$

where the term, $f\frac{Y}{Z} + c_Y$, corresponds to the projection v of $\mathbf{P}_W$ in the static virtual frame $\mathcal{C}$, and the second term models the time-varying displacement due to camera motion. We rewrite this as

$$v' = v + A_y \cos(\hat{\omega}t + \phi_Y), \tag{5}$$

where $A_y = f\frac{Y_0}{Z}$ is the amplitude of the induced sinusoidal motion as in Fig. 3.

Thus from Eq. 5, we identify three key unknowns for motion compensation in one axis: the amplitude A_y (dependent on scene depth and oscillation magnitude), the phase ϕ_Y, and the coordinate v in the virtual frame. Accurate estimation of these parameters allows precise inversion of the apparent motion in the event stream, recovering the underlying scene as if the camera were stationary.

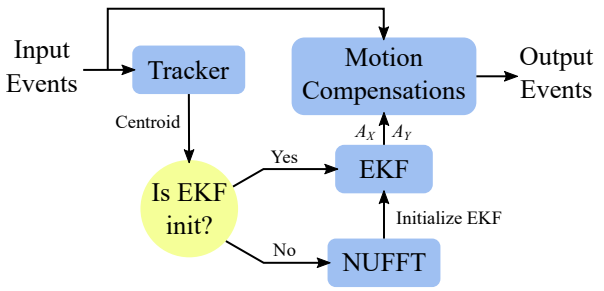


Figure 4. **Schematic representation of VIBES**. The input event stream is processed by a tracker that extracts trajectories used to estimate the dominant frequency, amplitude, and phase shift of the oscillatory motion. Once the sinusoidal motion is characterized, an Extended Kalman Filter (EKF) is initialized independently for each axis to track the induced motion. The EKF estimates are then used to compensate for the induced motion in the incoming event stream.

3.2. Motion Compensation Setup

Once a model of the induced motion has been established, the next step is to eliminate motion from the incoming event stream. This requires estimating the dynamic motion parameters, which varies as the camera or the scene changes over time. To achieve this, we employ a series of software modules, illustrated in Fig. 4. Below, we describe each stage of the pipeline and its role in accurate motion compensation.

3.2.1 Event Tracker

We first extract N centroid trajectories $(u'_i, v'_i, t_i)_{i=1}^N$ using an event tracker. In this work, we use HASTE [1], a lightweight algorithm that operates directly on the raw event stream. HASTE tracks local spatiotemporal patterns without requiring additional image processing, providing robust estimates of centroid positions over time. These trajectories capture the periodic motion induced by the camera and are subsequently used to estimate the unknown parameters of Eq. 5. To initialize the HASTE trackers, we manually select regions with high texture or edge content.

3.2.2 Frequency Estimation with NUFFT

Since the camera motion is oscillatory, recovering its frequency is essential. However, events are sampled irregularly in time, which makes standard Fourier analysis unsuitable. To obtain an initial estimate of the angular velocity parameter in Eq. 5, we employ the non-uniform fast Fourier Transform (NUFFT) [4]. The NUFFT decomposes a signal into its constituent frequencies but is specifically designed to operate on irregularly sampled data — making it ideal for event-based inputs, which are inherently asynchronous and non-uniform in time. Before applying the

NUFFT, we remove the DC component from the tracker signal (Sec. 3.2.1) to isolate oscillatory motion from the static offset introduced by the patch location. Temporal samples t_i are then rescaled to $[-\pi, \pi]$, yielding normalized timestamps $\bar{t}_i$. This produces a zero-mean, time-normalized samples $(\bar{u'}_i, \bar{v'}_i, \bar{t}_i)_{i=1}^{N}$ which are passed to the NUFFT. The transform returns the top M dominant frequencies, from which we extract the angular frequency $\hat{\omega}$ of the camera motion. Estimation is performed independently for the X and Y axes. Using $\hat{\omega}$, we fit the tracked samples to a sinusoidal model using least-squares optimization, recovering the amplitudes and phase offsets of the motion. These parameters initialize the Extended Kalman Filter (EKF) described in Sec. 3.2.3.

3.2.3 Extended Kalman Filter Setup

The Extended Kalman Filter (EKF) refines the motion parameters over time by incorporating a dynamical model and accounting for observation noise. This yields temporally consistent parameter estimates, which are essential for reliable motion compensation. We model all motion parameters dynamically in the EKF, but they are updated differently. Oscillatory frequency and phase are assumed to be globally constant across the image plane, whereas the oscillation amplitude depends on the scene depth. We make the assumption that all objects in the same scene share the same depth plane, although our implementation allows for multiple trackers which may have regions with different depths. Thus, we vary the u', v' parameters with each specific tracker to allow for tracking at different depths.

The camera frame's angular position θ evolves as

$$\theta_t = \theta_{t-1} + \hat{\omega}\Delta t, \tag{6}$$

where Δt is the time elapsed since the last update. The projected event position in the Y direction is modeled as

$$v' = a_y \sin(\theta) + b_y \cos(\theta) + v \tag{7}$$

which is equivalent to Eq. 5 but avoids the numerical instability that arises from directly modeling the phase shift ϕ or explicit time dependence. This formulation also mitigates ambiguities caused by trigonometric wrapping. The parameters for the y-axis expression in Eq. 5 can be recovered as

$$A_Y = \sqrt{a_y^2 + b_y^2},\ \phi_Y = \arctan 2(b_y, a_y), \tag{8}$$

The EKF state vector is defined as

$$\mathbf{x} = \begin{bmatrix} \theta & \hat{\omega} & a_y & b_y & v \end{bmatrix}^\top. \tag{9}$$

This state is updated as new tracker measurements arrive, with the corresponding Jacobians provided in Supplementary Appendix S2. The filter can operate on the entire image plane or, more effectively, on localized patches centered around feature points. Our implementation allows multiple trackers to be instantiated in parallel, enabling flexible motion parameter estimation across the scene.

The EKF enables us to predict the expected oscillatory amplitude of the incoming event stream at any given time, and subsequently remove it to obtain a motion-compensated event stream, as illustrated in panel (c) of Fig. 1.

3.3. Hardware Prototype

For all real-world data collection and experiments, we use the Prophesee EVK3 event camera with IMX636 (1280x720 px) sensor. To induce motion in the Prophesee camera, we rigidly attach a DC motor [14] to the body of the event camera. Affixed to the DC motor shaft is a mass, which has a slight eccentricity. This mass was custom machined, although any off-axis weight with sufficient eccentricity and mass would suffice. A casing enclosing the event camera was 3D printed, and wrapped in foam. The foam acts as a passive-spring damper, enabling the motion model described in Sec. 3.1. Lastly, a 3D printed shell supporting the foam and event camera casing was used to ensure 2D planar motion for the event camera and to enable mounting for real-world use. We refer to Supplementary Sec. S2.3 for more details regarding our prototype and modeling assumptions.

4. Results

We evaluated VIBES on three real-world scenes captured with the Prophesee EVK3 (1280x720 px), demonstrating efficient motion-compensated event generation while preserving reconstruction quality and edge detail. Additional validation on the AMI-EV dataset [24], recorded with a DVXplorer (640×480 px), confirms VIBES effectiveness and hardware independence.

4.1. Data Collection

We record real-world scenes and produce output data for three settings. First, we capture the scene using a standard event camera (S-EV). Second, we capture the scene with vibration (V-EV). Finally, we apply our motion compensation algorithm to the acquired V-EV data (VIBES). Overall, we evaluated our method on four real-world scenes: *AMI-EV* [24], *Logo*, *Pattern Checkerboard*, and *Pattern*. Each scene consists of a textured pattern moving in front of the camera. The *Logo* pattern contains text and rich textures, moving slowly while performing partial rotations. The *Pattern Checkerboard* and *Pattern* sequences involve planar patterns moving at different speeds, with the *Pattern* exhibiting faster motions. Images of all patterns are provided in Supplementary Sec. S2.4. To ensure a fair comparison in the real-world setting, we employed a Franka robotic arm to repeatably move various patterns in front of the camera for the V-EV and S-EV settings. We mounted the camera

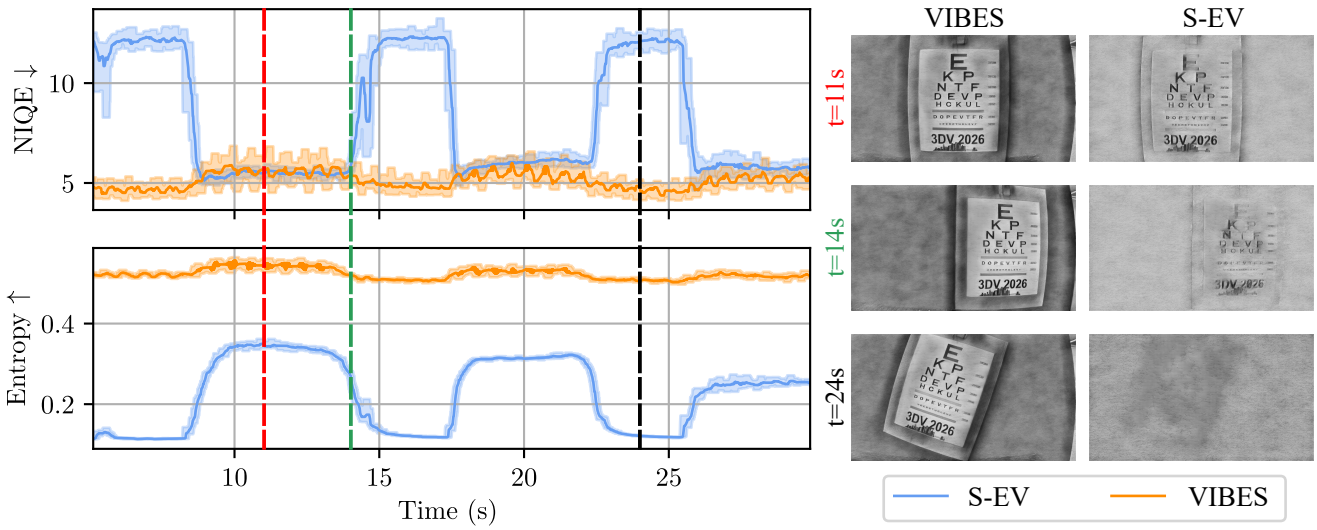


Figure 5. **Shannon entropy and NIQE scores** of reconstructed frames of binary accumulated event frames, computed over 10 ms time windows on the *Logo* real-world scene. The shaded range denotes the minimum and maximum values within every 10-frame window. Temporal screenshots of both the VIBES and S-EV reconstructions, generated using E2VID [38, 39], are shown on the right. The regions where the S-EV method produces low-quality images are those where the scene is completely static or there is slow motion.

on the physical prototype mass-spring-damper system described in Sec. 3.3.

4.2. Metrics

We evaluated the performance of our method for image quality, edges, and texture detection using the following metrics. We provide mathematical definitions of all metrics in Supplementary Sec. S2.5.

- **Shannon entropy** [40]**.** The entropy of binary images accumulated over a 10 ms window indicates the amount of information captured by the sensor. Higher values correspond to richer input data [24].
- **Natural Image Quality Evaluator (NIQE)** [33]**.** We compute NIQE on reconstructed images obtained using E2VID [38, 39] with a 10 ms event stream window. Lower values indicate reconstructions closer to the statistical properties of natural images and higher perceptual quality.
- **Variance of image pixel values.** We count per-pixel events over a 10 ms window and compute the variance of the resulting image pixel values. High variance indicates well-aligned events forming sharp edges, while low variance reflects blurred or dispersed events [18].
- **Gradient magnitude of image pixel values.** We count per-pixel events over a 10 ms window and calculate the gradient magnitude of the resulting image pixel values. High values indicate strong edges and texture.
- **Edges continuity and fragmentation.** We count per-pixel events over a 33 ms window, smooth the resulting image with a Gaussian blur, binarize, and thin using the Zhang–Suen algorithm [52]. From the resulting edge maps, connected components are extracted [15, 48], and their number reflects fragmentation (many small components imply broken or noisy edges, fewer longer ones imply better continuity). We additionally compute the average edge length [35] and count edge junctions, which together serve as indicators of edge coherence and texture reconstruction quality.

Scene	Status	Entropy ↑
AMI-EV [24]	S-EV	0.08 ± 0.05
	VIBES	$\mathbf{0.24 \pm 0.04}$
Logo	S-EV	0.21 ± 0.09
	VIBES	$\mathbf{0.52 \pm 0.01}$
Pattern Checkerboard	S-EV	0.29 ± 0.13
	VIBES	$\mathbf{0.59 \pm 0.04}$
Pattern	S-EV	0.30 ± 0.12
	VIBES	$\mathbf{0.39 \pm 0.08}$

Table 1. **Shannon entropy** values (mean, standard deviation) computed on binary accumulated event frames over 10 ms windows. Higher entropy indicates a richer and more informative event stream.

4.3. Image Quality Evaluation

To evaluate the consistency of the output, we compute the entropy of accumulated event frames. The plot in Fig. 5 reports the entropy for the *Logo* scene and results for all scenes are shown in Tab. 1. Our results show that our induced motion, although small (less than a millimeter in the x- and y-directions), is sufficient to yield a higher information gain compared to the static event camera. It is worth noting that the variance of the entropy information is low when a vibratory motion is induced, meaning that we have

Scene	Status	NIQE ↓
AMI-EV [24]	S-EV	10.4 ± 5.0
	VIBES	$\mathbf{9.1} \pm 1.0$
Logo	S-EV	8.6 ± 9.4
	VIBES	$\mathbf{5.1} \pm 0.3$
Pattern Checkerboard	S-EV	8.2 ± 9.0
	VIBES	$\mathbf{6.3} \pm 0.2$
Pattern	S-EV	9.4 ± 5.0
	VIBES	$\mathbf{8.2} \pm 0.6$

Table 2. **Natural Image Quality Evaluator (NIQE)** average values (mean, standard deviation) computed using E2VID [38] at 100 Hz.

Scene	Status	Variance ↑	Grad. Mag. ↑
AMI-EV [24]	S-EV	0.06 ± 0.05	0.04 ± 0.02
	V-EV	0.32 ± 0.06	0.09 ± 0.02
	VIBES	$\mathbf{0.44 \pm 0.10}$	$\mathbf{0.12 \pm 0.02}$
Logo	S-EV	0.11 ± 0.08	0.18 ± 0.07
	V-EV	2.34 ± 0.13	0.48 ± 0.03
	VIBES	$\mathbf{3.86 \pm 0.34}$	$\mathbf{0.56 \pm 0.02}$
Pattern Checkerboard	S-EV	0.16 ± 0.11	0.22 ± 0.11
	V-EV	1.9 ± 0.15	0.57 ± 0.04
	VIBES	$\mathbf{3.16 \pm 0.33}$	$\mathbf{0.60 \pm 0.04}$
Pattern	S-EV	0.22 ± 0.18	0.19 ± 0.09
	V-EV	0.76 ± 0.16	$\mathbf{0.30 \pm 0.06}$
	VIBES	$\mathbf{1.11 \pm 0.25}$	$\mathbf{0.30 \pm 0.06}$

Table 3. **Quantitative evaluation of motion compensation.** We report image variance and average gradient magnitude, expressed as mean and standard deviation across frames. Higher values indicate sharper images and better-preserved scene structure.

a consistent and stable information input that does not depend on scene dynamics (Tab. 1).

Fig. 5 shows the NIQE and entropy results for the *Logo* scene, while per-scene NIQE values are provided in Tab. 2. The results demonstrate that induced motion consistently leads to improved reconstructions compared to a static event camera. A clear correlation emerges between entropy and image quality: higher entropy yields better NIQE scores in the S-EV setup. Conversely, when entropy is low — such as in static or low-motion scenes — our reconstructions remain stable and do not fade, ensuring consistent information output over time.

4.4. Edge Extraction Evaluation

We evaluate the sharpness of the accumulated frames by computing the variance and the gradient magnitude for the three setups: S-EV, V-EV, and VIBES. Tab. 3 reports the average and variance of these metrics across all scenes.

The results indicate that our method effectively compensates for induced motion, achieving higher variance and gradient magnitude values, which correspond to sharper edges and stronger structural cues. We evaluate edge quality using both continuity and fragmentation metrics. In Tab. 4, we report statistics for the described metrics. The evaluation shows that VIBES produces clean, well-connected edge maps, making it well-suited for preserving object boundaries and overall shape structure. In contrast, the standard event camera without induced vibration tends to produce noisier and more fragmented edges.

4.5. Runtime Evaluation

Our software stack, implemented in C++ with the Metavision API by Prophesee, is evaluated by running the compensation script ten times per dataset. The average processing time is 15.28 ± 3.7 ns per event ($\sim$ 65 Mev/s), including undistortion and motion compensation. The initial parameters estimation Sec. 3.2.2, requires 104.8 ± 0.03 ms, but remains non-blocking thanks to a multithreaded design that tracks incoming events while buffering and processing compensation. A video demonstration is provided in the supplementary material. Although runtime is fast, real-time performance depends on sensor resolution and scene dynamics; in our experiments, event rates peaked at 45 Mev/s.

5. Applications

In this section, we demonstrate two additional applications of our pipeline beyond its primary goal of showing that induced motion can generate a stable event stream suitable for high-quality image reconstruction and edge extraction.

5.1. Frequency Estimation

While our software stack is designed to detect the frequency of the camera's oscillatory motion, it can also be applied to estimate the vibration frequency of an object in the scene with a static camera.

To evaluate this, we displayed a triangle moving along a small circular path on a computer monitor, pointed the event camera at the screen, and applied VIBES to estimate the motion frequency. Given the monitor's refresh rate of 75 Hz, we tested a range of frequencies from 7.5 Hz to 22.5 Hz, noting that higher frequencies were affected by aliasing due to temporal sampling limitations. For each target frequency, we repeated the estimation process 10 times, reporting the mean estimated frequency and the corresponding 3σ error.

The results in Tab. 5 demonstrate that VIBES achieves accurate frequency estimation with low bias and variance. We note, however, a slight increase in bias at higher frequencies, with deviations of up to 0.1Hz between the ground truth and the predicted values. This effect is likely due to aliasing, as we are able to estimate higher frequencies both in the simulated Sec. 5.2 and in real-world data.

Scene	Status	Avg. Num. Components ↓	Avg. Contour Length ↑	Avg. Junction Count
AMI-EV [24]	S-EV	91.1 ± 17.2	4.2 ± 2.4	63.3 ± 54.6
	V-EV	$\mathbf{82.9 \pm 14.1}$	$\mathbf{8.3 \pm 1.6}$	147.9 ± 31.8
Logo	S-EV	265.8 ± 221.4	3.6 ± 3.1	136.5 ± 165.8
	V-EV	$\mathbf{175.0 \pm 52.5}$	$\mathbf{32.1 \pm 10.8}$	979.3 ± 349.2
Pattern Checkerboard	S-EV	300.3 ± 232.9	6.7 ± 5.3	193.2 ± 176.3
	V-EV	$\mathbf{140.7 \pm 40.2}$	$\mathbf{49.9 \pm 17.3}$	691.3 ± 63.2
Pattern	S-EV	138.7 ± 102.3	5.6 ± 4.2	226.9 ± 197.5
	V-EV	$\mathbf{116.3 \pm 35.1}$	$\mathbf{15.6 \pm 4.2}$	304.4 ± 127.6

Table 4. **Edge extraction evaluation on captured scenes.** We report the number of connected components (lower is better), the average contour length in pixels (higher is better), and the number of junctions, expressed as mean ± standard deviation across frames. Across most scenes, VIBES reduces edge fragmentation and increases contour continuity compared to S-EV, indicating cleaner, more coherent edge maps.

Target (Hz)	Est. (Hz)	Avg. Abs. Error ↓	3σ ↓
7.5	7.486	0.0617	0.202
10.0	10.074	0.0754	0.177
12.6	12.684	0.0923	0.234
15.2	15.255	0.0764	0.264
17.4	17.476	0.0829	0.225
20.0	20.100	0.1000	0.189
22.6	22.644	0.0517	0.147

Table 5. **Scene frequency estimation**. VIBES performs precise estimation of frequency, although performance degrades as the frequency increases. Reported results are based on 10 samples.

5.2. Relative Depth Awareness

We explore how parallax induced by the camera vibration can encode information about scene depth.

Note that this type of depth estimation is not possible in setups like the wedge-prism solution proposed by [24], because the camera center of projection does not change.

We tested three synthetic scenes containing two staggered patterns at different depth planes, as in Fig. 6. To create synthetic data, we generated RGB frames using Blender [8] at 1000 FPS and converted them to events using V2E [9]. Camera motion was simulated according to the sinusoidal motion model described in Sec. 3.1. We instantiated two trackers per pattern. Then we stored the amplitudes estimated by the EKF for each of the patterns and averaged them across the entire scene duration. Finally, we computed the ratio between the average amplitudes to get the estimation of the relative depth. We repeated this process ten times per scene. Our simulated results show that our system reliably predicts the relative depth of the two patterns: for ground-truth relative depth ratios of 0.33, 0.50, and 0.66 the predicted ratios are 0.41 ± 0.01, 0.59 ± 0.02, and 0.80 ± 0.03. Please see Supplementary Sec. S2.6 for additional details.

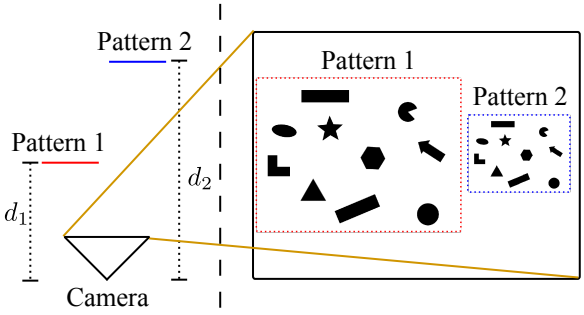


Figure 6. **Illustration of depth prediction setup**. The two patterns are placed at different distances from the camera and we are capable of estimating the ratio $\frac{d_1}{d_2}$ (or vice versa) by observing the amplitudes of the perceived motion of the camera measured by the tracker (Sec. 3.2.1).

6. Conclusion

In this work, we presented VIBES, a novel bio-inspired method that enhances event cameras by actively inducing motion. Our approach produces persistent event streams for static scenes, overcoming a common obstacle in event-based vision.

We see several promising avenues for future work. For example, the vibration system could be made adaptive — i.e., disabled in highly dynamic scenes with dense event streams, and reactivated when the event density drops. Such an approach would preserve the low-power benefits of event cameras while ensuring a continuous visual signal when needed. Additionally, although our method already handles high event rates, downsampling or other event filtering techniques may be considered to further improve the real time performance, given the high event bandwidth generated by the induced motion. Our approach could also be combined with other event-based algorithms to improve performance on downstream tasks.

S1. Supplementary Material

S1.1. Motion Model Details

Without loss of generality, we focus on one axis, y, noting that the motion along the x axis is equivalent up to a phase shift. The displacement of the camera as a function of time follows the steady-state solution of a damped harmonic oscillator, governed by the second-order differential equation

$$M\ddot{y}(t) + c\dot{y}(t) + ky(t) = me\omega^2 \cos(\omega t), \tag{S1}$$

where M is the total system mass, m is the eccentric mass, e is the eccentricity, c is the damping coefficient, and k is the spring constant. The external force driving the system has amplitude $F_0 = me\omega^2$ and angular frequency ω. The exact angular frequency generated by the motor is modeled in Sec. S1.1.1. $y(t)$, $\dot{y}(t)$ and $\ddot{y}(t)$ denotes the displacement, velocity and acceleration, of the camera respectively. After transient dynamics have decayed, the system reaches a steady-state oscillation described by

$$\hat{A}_Y = \frac{F_0}{\sqrt{(k - m\omega^2)^2 + (c\omega)^2}}, \tag{S2}$$

$$\phi = \tan^{-1}\left(\frac{c\omega}{k - m\omega^2}\right), \tag{S3}$$

$$x_s(t) = \hat{A}_Y \sin(\omega t - \phi). \tag{S4}$$

This solution describes a smooth, predictable sinusoidal motion that ensures events are continuously generated, even when viewing a static scene. Since the motion is mechanically induced and follows a known trajectory, it provides a strong prior for both data association and noise filtering in subsequent stages of our pipeline.

S1.1.1 Motor Modeling

By varying the applied voltage to the motor, we can control the speed and, consequently, the amplitude and frequency of the induced harmonic motion. To relate the applied voltage V_A to the motor's angular speed ω_m, we use a classical DC motor model, depicted in Figure S1:

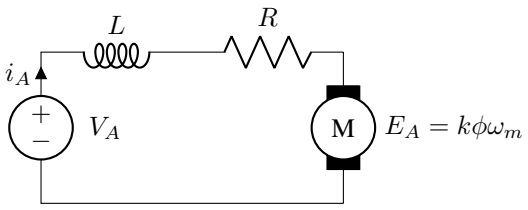


Figure S1. **Simple DC motor model.**

where i_A is the armature current, k_ϕ is the motor constant, and R is the armature resistance. The back electromotive force (EMF) E_a is proportional to the rotational speed ω_m. Neglecting inductance (which has minimal impact in steady-state operation), the motor equations are

$$V_A = i_A R + E_a = i_A R + k_\phi \omega_m \,, \tag{S5}$$

$$\omega_m = \frac{V_A}{k_\phi} - \frac{R}{(k_\phi)^2} T_q \,, \tag{S6}$$

We estimate the motor constant k_ϕ using manufacturer specifications. Tab. S1 shows a full list of motor parameters. Specifically, we calculate k_ϕ using

$$\omega_n^N = \frac{V_A^N}{k_\phi},$$

where values superscripted with N indicate nominal or rated values, as reported by the manufacturer, and are shown in Tab. S1. Here, T_q is the load torque at the motor shaft, determined by the weight and eccentricity of the eccentric mass. We approximate T_q using the following

$$T_q = e \cdot m \cdot g.$$

S1.1.2 Matlab Modeling

Vibrations were induced by applying a voltage of 2.0 V to the DC motor setup, producing angular velocities in the range $(250, 350)$ rad/s. More details are provided in Sec. S2.3 of our supplementary material.

For frequency estimation, the NUFFT initialization parameters were set to $\omega_{\text{min}} = 30$ rad/s and $\omega_{\text{max}} = 500$ rad/s, matching the expected frequency range of the mechanical system.

S2. Extended Kalman Filter Details

For the EKF, we require a compact parametrization of the oscillatory motion that is numerically stable and avoids explicit dependence on absolute time t. We model the vertical image coordinate $v'(t_k)$ (an analogous formulation applies to the horizontal coordinate) in discrete time as

$$v'_k = A_{y,k} \cos(\hat{\omega}_k + \phi_Y) + v_k \tag{S7}$$

$$= A_{y,k} \big[\cos(\hat{\omega}_k) \cos\phi_Y - \sin(\hat{\omega} t_k) \sin\phi_Y \big] + v_k \tag{S8}$$

$$= \underbrace{(A_{y,k} \cos\phi_Y)}_{b_{y,k}} \cos(\hat{\omega}_k) + \underbrace{(-A_{y,k} \sin\phi_Y)}_{a_{y,k}} \sin(\hat{\omega}_k) + v_k \tag{S9}$$

$$= a_{y,k} \sin(\theta_k) + b_y \cos(\theta_k) + v_k, \tag{S10}$$

where we augment the state with θ_k defined as

$$\theta_k = \theta_{k-1} + \hat{\omega}_k \Delta t, \tag{S11}$$

$\mathbf{k}\phi$	$\mathbf{R}$ [Ω]	$\mathbf{V_A^N}$ [V]	$\mathbf{i_A^N}$ [mA]	$\omega_\mathbf{m}^\mathbf{N}$ [RPM]	$\mathbf{m_{mass}}$ [g]	$\mathbf{e}$ [mm]	$\mathbf{T_q}$ [N · mm]
0.00374	3.75	3	110	6600	7.1	3.1	0.216

Table S1. **Motor and Vibration Setup Parameters.** Experimental values of the motor and vibration system, including motor constants, operating parameters, and physical properties.

with $\Delta t = t_k - t_{k-1}$ the elapsed time between steps.

This reparameterization introduces the coefficients (a_y, b_y) instead of $(A_{y,k}, \phi_Y)$, which has several advantages: (i) it avoids the explicit trigonometric wrapping of the phase ϕ_Y, (ii) it ensures smooth Jacobians for the EKF update step, and (iii) it makes the model linear in (a_y, b_y, v), with only θ evolving nonlinearly.

The same formulation applies to the horizontal coordinate $u'(t_k)$ with parameters (a_x, b_x). Without loss of generality, and to simplify notation, we drop the x/y subscripts and use a generic c to indicate the DC component of the sinusoid. The complete EKF state vector for the sinusoid is then

$$\mathbf{x}_k = \begin{bmatrix} \theta_k & \omega_k & a_k & b_k & c_k \end{bmatrix}^T, \tag{S12}$$

where c_k denotes the motion-compensated, stable image coordinate in the virtual frame. The state covariance is $\mathbf{P}_k \in \mathbb{R}^{5\times5}$.

S2.1. Propagation Step

The discrete-time process model is

$$\mathbf{x}_{k|k-1} = \begin{bmatrix} \theta_{k-1} + \omega_{k-1}\Delta t \\ \omega_{k-1} \\ a_{k-1} \\ b_{k-1} \\ c_{k-1} \end{bmatrix}. \tag{S13}$$

Given the propagation Eq. S11, the state transition Jacobian is

$$\mathbf{F}_k = \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{S14}$$

The predicted covariance is updated as

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \, \mathbf{P}_{k-1} \, \mathbf{F}_k^\top + \mathbf{Q}, \tag{S15}$$

where $\mathbf{Q} \in \mathbb{R}^{5\times5}$ is the process noise covariance.

S2.2. Measurement Update

The observation model at time t_k is given by Eq. S10, with measurement noise $n_k \sim \mathcal{N}(0, \sigma_r^2)$:

$$h_k = a_k \sin(\theta_k) + b_k \cos(\theta_k) + c_k + n_k. \tag{S16}$$

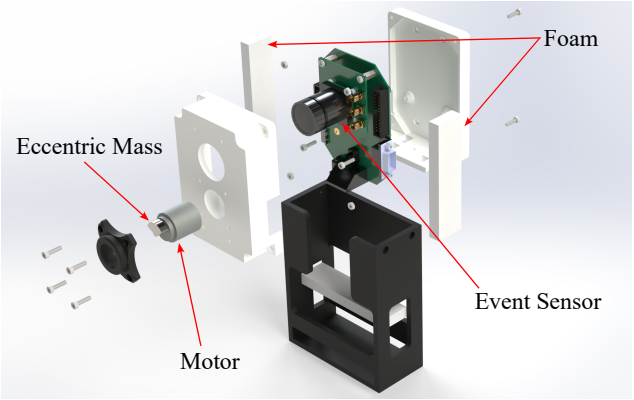


Figure S2. **Exploded view of the camera assembly**, with eccentric rotating mass for induced vibration.

The measurement Jacobian is

$$\mathbf{H}_k = \frac{\partial h_k}{\partial \mathbf{x}} \tag{S17}$$

$$= \begin{bmatrix} a_k \cos(\theta_k) - b_k \sin(\theta_k) & 0 & \sin(\theta_k) & \cos(\theta_k) & 1 \end{bmatrix}. \tag{S18}$$

The innovation is

$$\tilde{y}_k = y_k - h(\mathbf{x}_{k|k-1}), \tag{S19}$$

with innovation covariance

$$S_k = \mathbf{H}_k \, \mathbf{P}_{k|k-1} \, \mathbf{H}_k^\top + \mathbf{R}. \tag{S20}$$

where $\mathbf{R}$ is the measurement noise covariance.

The Kalman gain is

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \, \mathbf{H}_k^\top \, S_k^{-1}. \tag{S21}$$

The state and covariance are updated using the Joseph form

$$\mathbf{x}_k = \mathbf{x}_{k|k-1} + \mathbf{K}_k \tilde{y}_k, \tag{S22}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \, \mathbf{P}_{k|k-1} \, (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^\top + \mathbf{K}_k \, \sigma_r^2 \, \mathbf{K}_k^\top. \tag{S23}$$

This form preserves numerical stability and guarantees that $\mathbf{P}_k$ remains positive semi-definite.

S2.3. Physical Prototype Details

To induce motion in the camera, we rigidly attach a DC motor to the body of the event camera. The motor drives an

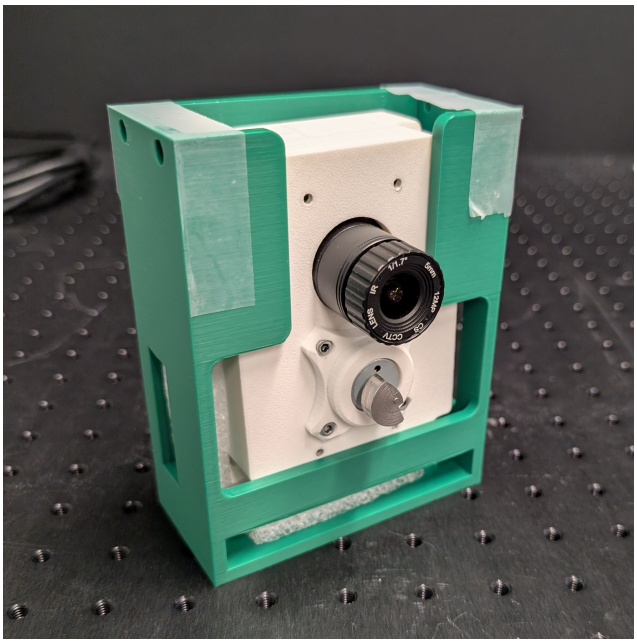


Figure S3. **A picture of the real prototype**

eccentric (off-axis) mass, acting as a forced rotating unbalance. This mass was custom-machined out of 1" diameter tool steel stock. The parameters of the mass are provided in Tab. S1. More detail about the modelling of the motor is provided in Sec. S1.1.1.

An exploded view of the entire assembly is shown in Fig. S2. All assembly components except for the event camera, and motor, are designed in SolidWorks and secured with standard fasteners. The white casing and black shell, shown in Fig. S2, are 3D printed using Polylactic Acid (PLA).

The entire camera-motor assembly is mounted to an outer shell cushioned with foam, which acts as a passive spring-damper system. This shell also limits movement of the camera in the Z direction, constraining the resulting motion predominantly to the X-Y image plane which ensures predictable, repeatable oscillations. These small oscillations stimulate the event sensor, triggering events even in otherwise static scenes and revealing fine details that would not be detectable without motion.

S2.4. Dataset

Here we provide more detailed information on the real-world dataset and synthetic datasets.

S2.4.1 Real World Data

Our experiments were performed in a scene containing our physical VIBES prototype mounted to a table, along with a Franka Emika arm. We printed three patterns, shown in Figure Fig. S5, and taped them to a sheet of cardboard, which

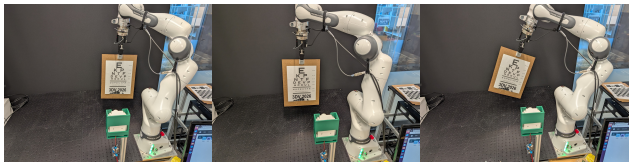
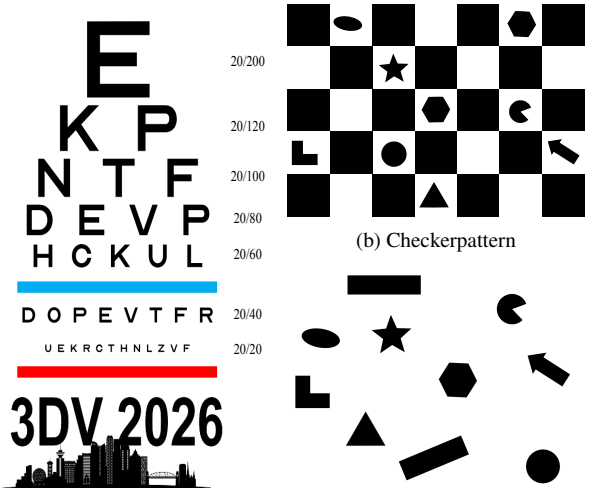


Figure S4. **Data collection setup.** The Franka arm moves the board with the pattern in front of the prototype VIBES hardware.



(a) Logo (b) Checkerpattern (c) Pattern

Figure S5. **The patterns used in the real-world dataset.**

Scene Ratio	Pattern 1 Distance	Pattern 2 Distance
0.33	21.0 cm	64.1 cm
0.5	32.1 cm	64.1 cm
0.66	40.6 cm	64.1 cm

Table S2. **Distances used in the synthetic depth estimation data.**

was mounted in the gripper of the Franka arm. We then moved the patterns in front of the event camera by creating motion plans in the Franka Desk software. Figure Fig. S4 shows pictures of our data collection scene.

S2.4.2 Synthetic Data

We used synthetic data generated in Blender [8] to test the depth estimation capabilities of VIBES. Our virtual camera had an angular FOV of $51.7°$ and our VIBES motion radius was set to 3 mm. VIBES in our synthetic scene can detect differences in depth up to **148** cm (see Sec. S2.6). Given this, we placed objects in the virtual scene well within this range to ensure depth estimation was possible. Table Tab. S2 reports the distances of the objects in the three synthetic scenes.

S2.5. Details on Metrics Equations

Here we give the full mathematical definitions of the metrics used in Section Sec. 4.

Shannon Entropy [40]: We calculate Shannon Entropy over binary accumulated event images, where the binary pixels are treated as drawn from the same distribution. This is calculated as follows:

$$H(X) := -\sum_{x \in X} p(x) \log_2 p(x), \tag{S24}$$

where x is each pixel in an image $\mathcal{X}$.

Natural Image Quality Evaluator (NIQE) [33]:

We used NIQE as introduced in Mittal et al. [33]. For brevity we do not include the full algorithm. Roughly, NIQE estimates natural image quality without requiring access to distorted images by calculating certain image statistics from a large dataset of natural images, and then comparing a multivariate gaussian model of those natural image statistics to the same statistics calculated on a potentially distorted image. We specifically use the MATLAB implementation which contains a pre-trained model.

Variance of image pixel values: Following [18] which found that high variance in accumulated event frames corresponds to sharper edges. We generated accumulated frames over a 10 ms window, and then calculated the variance of the per-pixel event counts:

$$V(X) := \frac{1}{N_p} \sum_{i,j} (h_{ij} - \mu_X)^2, \tag{S25}$$

where N_p is the number of pixels of image X and μ_X is the mean of X

Gradient magnitude of image pixel values: We perform the same event accumulation procedure as above, but calculate per-pixel gradients and take an average of the magnitude over the image. We approximate per-pixel gradients using forward differences which can be calculated with convolutions:

$$G_x = \begin{bmatrix} -1 & 1 \end{bmatrix} * X, \tag{S26}$$

$$G_y = \begin{bmatrix} 1 \\ -1 \end{bmatrix} * X, \tag{S27}$$

$$M(X) = \sqrt{G_x^2 + G_y^2}. \tag{S28}$$

Edges continuity and fragmentation:

Once the raw edge maps are smoothed and thinned, we calculate three metrics.

The **average number of components** measures the number of components in an edge map, with a greater number showing less fragmented and noisy edges. We specifically consider 8-connectivity, which considers pixels to be neighbors if another pixel is in any of the 8 surrounding pixels (practically, this means regions remain connected across diagonals).

The **average contour length** measures the average lengths of the edges in the edge map. Given that in our edge maps the edges are roughly ~ 1 pixel wide, we simply calculate the area of each edge by counting the total pixels to calculate the edge length.

The **average junction count** measures the number of "junctions" in the edge map, which is defined as the number of pixels with more than 2 neighbors in their 8-connectivity area (meaning in the 8 pixels surrounding a pixel, at least 2 are occupied by other positive edge pixels).

S2.6. Details on Camera Distance and Correction

Given the pinhole camera and the motion model, we can compute the minimum maximum distance required to generate events that cover more than one pixel, that is, after a certain distance, the induced motion can generate a stream of events with positive and negative polarity in the same pixel location.

Assume the axis of the camera's rotation is pointing at a point we wish to observe as a feature. Define the distance from the camera to the point as d, and the diameter of the camera's rotation as $\hat{A}_y$. We can observe that the angle from the camera's axis of rotation is given by

$$\theta = \frac{\pi}{2} - \arctan\left(\frac{d}{r}\right). \tag{S29}$$

The formula for the change in pixel space of this object is given by

$$\Delta p = \tan(\theta) * \frac{\text{Resolution}/2}{\tan\left(\frac{\text{FOV}}{2}\right)}. \tag{S30}$$

We used the Prophesee EVK3 event camera with the IMX636 sensor has a resolution of (1280 x 720) and a 5 mm lens focal length. Using Prophesee's optics calculator [1] we found our camera has a vertical angular FOV of 39°. Our rotation radius with our gimbal setup was estimated based on a spring-dampener ODE model to be 1mm. Using our equation, we can find the minimum value for d such that Δp is less than or equal to 1, which represents a change of less than a pixel of the feature given the camera motion. Solving for d with an inverse solver gave a distance of $\mathbf{47.3}$ **cm** beyond which the camera cannot detect event changes through the camera motion. Due to this relatively shallow depth, we use synthetic data for our depth estimation experiment where we can easily increase the magnitude of motion to allow for deeper depth perception.

Methodology	Motor Type	Power Consumption ↓	Measurement/Evaluation Basis
VIBES	DC Hobby Motor [14]	**0.282W**	Full Power
Orchard et al. [34]	Dynamixel MX-28T [12]	2.4W	Standby
Testa et al. [43]	PTU-D46-17 [16]	1W	Holding Power Off
		6W	Low Power
		13W	Full Power
Testa et al. [44]	PTU-E46 [17]	1W	Holding Power Off
		6W	Low Power
		13W	Full Power
AMI-EV [24]	DJI M2006 BLDC [10]	14.4W	No-load

Table S3. **Power Consumption**. VIBES facilitates persistent event generation, while using significantly less power than other methods.

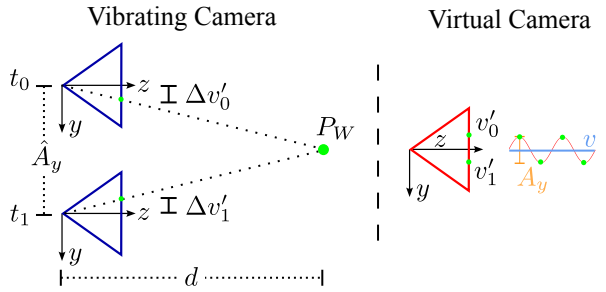


Figure S6. **Schematic representation of the induced stereo system.** The induced motion produces an effect analogous to a stereo system. The blue cameras are the vibrating camera at two different times, t_0 and t_1. The point P_W is projected onto the camera plane, and at the two different timestamps has a different position v'_0 and v'_1 respectively. On the virtual camera, we want to recover the static omponent of the vibratonal motion indicated as v. While the baseline $\hat{A}_y$ is unknown, the amplitude of the sinusoidal motion A_y can be estimated from the incoming event stream. This amplitude provides a strong cue for relative depth: if multiple tracks are initialized in the scene, the ratio between their amplitudes corresponds to their relative depth, as described in Eq. S32.

S2.7. Depth Amplitude Relation

The oscillation amplitudes estimated from the event stream inherently encode information about scene depth. As illustrated in Fig. S6, the induced motion can be interpreted as forming a pseudo-stereo-camera model. At the motion apexes along the y axis (analogously for x), the effective stereo baseline corresponds to the physical amplitude $\hat{A}_y$ imposed by the mass–spring–damper system.

From stereo geometry, given a baseline $\hat{A}_y$ and disparity $\Delta v'_0 - \Delta v'_1$ — which in our setting is observed as the oscillation amplitude A_y — the depth d can be expressed as

$$d = f \cdot \frac{A_y}{\hat{A}_y}. \tag{S31}$$

[1]See here: https://docs.prophesee.ai/stable/hw/manuals/optics_calculator.html

In practice, however, the true physical motion $\hat{A}_y$ of the camera is unknown and cannot be measured directly. Our system can only recover the apparent amplitudes A_y from the incoming event stream. While this prevents absolute depth estimation, the relative scaling of amplitudes across different tracked objects still provides a strong cue to relative depth. For instance, by initializing multiple independent EKFs at different image locations and compensating events locally, one can compare their oscillation amplitudes to infer depth ordering in the scene.

Specifically, following from Eq. 5 and considering only the oscillatory component, for two points p_1 and p_2 located at depths Z^1 and Z^2, we obtain

$$\frac{A_y^1}{A_y^2} = \frac{f \cdot \frac{Y_0}{Z^1}}{f \cdot \frac{Y_0}{Z^2}} = \frac{Z^2}{Z^1}, \tag{S32}$$

where Y_0 denotes the (unknown but constant) physical baseline motion along the y axis. This ratio directly encodes the relative distances of points from the camera.

S2.8. Power Consumption

We report the power consumption of our rotating unbalanced mass system and compare its performance against alternative approaches. VIBES employs a DC hobby motor, with parameters listed in Sec. S1.1.1. From our experiments, we observe that the motor draws an average current of 0.141A, while the applied voltage was set at 2V. As most trials were conducted at 2V, we report the typical power consumption of our DC motor setup as

$$2V \cdot 0.141A = 0.282W$$

In these experiments, the DC motor was powered via a DC/DC buck converter. We do not account for the converter's efficiency in our calculations for two reasons. First, switched-mode DC/DC converters generally achieve high efficiency within this voltage range. Second, comparable

data on power supply efficiency from other methodologies are not readily available.

Power consumption values are summarized in Tab. S3. For each methodology, we report both the estimated power consumption and the basis of evaluation. When specific hardware is identified, we reference the corresponding manufacturer's datasheets. For example, the FLIR PTU units [16, 17] list distinct power requirements for holding (power off), low-power, and full-power modes. For VIBES, power consumption was measured directly during online experiments. For other approaches, expected values are reported, defaulting to no-load or standby power consumption when both online measurements and datasheet values are unavailable.

References

[1] I. Alzugaray and M. Chli. HASTE: multi-hypothesis asynchronous speeded-up tracking of events. In *Brit. Mach. Vis. Conf. (BMVC)*, 2020. 4

[2] Samya Bagchi, Peter Anastasiou, Matthew Tetlow, Tat-Jun Chin, and Yasir Latif. Event-based star tracking under spacecraft jitter: the e-sturt dataset. *arXiv e-prints*, 2025. 3

[3] Dwijay Bane, Anurag Gupta, and Manan Suri. Non-invasive qualitative vibration analysis using event camera. *arXiv e-prints*, 2024. 2

[4] Alexander H Barnett, Jeremy Magland, and Ludvig af Klinteberg. A parallel nonuniform fast fourier transform library based on an "exponential of semicircle" kernel. *SIAM J. Sci. Comput.*, 2019. 4

[5] Christian Brändli, Jonas Strubel, Susanne Keller, Davide Scaramuzza, and Tobi Delbruck. Elised — an event-based line segment detector. In *Int. Conf. Event-Based Control, Commun. Signal Proc. (EBCCSP)*, 2016. 2

[6] Zhiwen Chen, Jinjian Wu, Junhui Hou, Leida Li, Weisheng Dong, and Guangming Shi. Ecsnet: Spatio-temporal feature learning for event camera. *IEEE TCSVT*, 2022. 2

[7] FJJ Clarke. A study of troxler's effect. *Opt. Acta: Int. J. Opt.*, 1960. 1

[8] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 8, 11

[9] Tobi Delbruck, Yuhuang Hu, and Zhe He. V2e: From video frames to realistic dvs event camera streams. *arXiv e-prints*, 2020. 8

[10] DJI. Dji robomaster m2006 p36 brushless dc gear motor, 2025. 13

[11] Yan Dong and Tao Zhang. Standard and event cameras fusion for feature tracking. In *Proc. Int. Conf. Mach. Vis. and Appl. (ICMVA)*, 2021. 2

[12] Dynamixel. Dynamixel mx-28t. http://emanual.robotis.com/docs/en/dxl/mx/mx-28t/. 13

[13] Giulia D'Angelo, Victoria Clerico, Chiara Bartolozzi, Matej Hoffmann, P Michael Furlong, and Alexander Hadjiivanov. Wandering around: A bioinspired approach to visual attention through object motion sensitivity. *Neuromorphic Comput. Eng.*, 2025. 2, 3

[14] SparkFun Electronics. Geared hobby motor. https://www.sparkfun.com/hobby-motor-gear.html, 2025. 5, 13

[15] Christophe Fiorio and Jens Gustedt. Two linear time union-find strategies for image processing. *Theor. Comput. Sci.*, 1996. 6

[16] FLIR. Ptu d46. https://www.sustainable-robotics.com/reference/PTU/ptu_d46_datasheet.pdf. 13, 14

[17] FLIR. Ptu e46. https://www.sustainable-robotics.com/reference/PTU/PTU-manual-E46-1.00-SMALL.pdf, 2014. 13, 14

[18] Guillermo Gallego, Henri Rebecq, and Davide 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.*, 2018. 2, 6, 12

[19] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022. 1

[20] Daniel Gehrig and Davide Scaramuzza. Low-latency automotive vision with event cameras. *Nature*, 2024. 1

[21] Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. Eklt: Asynchronous photometric feature tracking using events and frames. *Int. J. Comput. Vis.*, 2020. 2

[22] Arren Glover and Chiara Bartolozzi. Robust visual tracking with a freely-moving event camera. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. IEEE, 2017. 2

[23] Guangsha Guo, Yang Feng, Hengyi Lv, Yuchen Zhao, Hailong Liu, and Guoling Bi. Event-guided image super-resolution reconstruction. *Sensors*, 2023. 2

[24] Botao He, Ze Wang, Yuan Zhou, Jingxi Chen, Chahat Deep Singh, Haojia Li, Yuman Gao, Shaojie Shen, Kaiwei Wang, Yanjun Cao, Chao Xu, Yiannis Aloimonos, Fei Gao, and Cornelia Fermüller. Microsaccade-inspired event camera for robotics. *Sci. Robot.*, 2024. 1, 2, 3, 5, 6, 7, 8, 13

[25] Zhe Jiang, Yu Zhang, Dongqing Zou, Jimmy Ren, Jiancheng Lv, and Yebin Liu. Learning event-based motion deblurring. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[26] Yongcheng Jing, Yiding Yang, Xinchao Wang, Mingli Song, and Dacheng Tao. Turning frequency to resolution: Video super-resolution via event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 2

[27] Lingdong Kong, Youquan Liu, Lai Xing Ng, Benoit R Cottereau, and Wei Tsang Ooi. Openess: Event-based semantic scene understanding with open vocabularies. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024. 1

[28] Bart Krekelberg. Microsaccades. *Curr. Biol.*, 2011. 1

[29] Siqi Li, Zhikuan Zhou, Zhou Xue, Yipeng Li, Shaoyi Du, and Yue Gao. 3d feature tracking via event camera. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024. 2

[30] Maximilian PR Löhr and Heiko Neumann. Contrast detection in event-streams from dynamic vision sensors with fixational eye movements. In *IEEE Int. Symp. Circuits Syst. (ISCAS)*. IEEE, 2018. 3

[31] Ryota Maeda, Yunseong Moon, and Seung-Hwan Baek. Event ellipsometer: Event-based mueller-matrix video imaging. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2025. 3

[32] Nico Messikommer, Carter Fang, Mathias Gehrig, and Davide Scaramuzza. Data-driven feature tracking for event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023. 1, 2

[33] A. Mittal, R. Soundararajan, and A. C. Bovik. Making a "completely blind" image quality analyzer. *IEEE Sign. Process. Letters*, 2013. 6, 12

[34] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.*, 2015. 2, 3, 13

[35] W Pabst and EJIP Gregorova. Characterization of particles and particle systems. *ICT Prague*, 2007. 6

[36] Liyuan Pan, Cedric Scheerlinck, Xin Yu, Richard Hartley, Miaomiao Liu, and Yuchao Dai. Bringing a blurry frame alive at high frame-rate with an event camera. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 2

[37] Bernd Pfrommer. Frequency cam: Imaging periodic signals in real-time. *arXiv e-prints*, 2022. 2

[38] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. Events-to-video: Bringing modern computer vision to event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 6, 7

[39] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. High speed and high dynamic range video with an event camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019. 6

[40] C. E. Shannon. A mathematical theory of communication. *Bell Sys. Tech. J*, 1948. 6, 12

[41] Waseem Shariff, Mehdi Sefidgar Dilmaghani, Paul Kielty, Mohamed Moustafa, Joe Lemley, and Peter Corcoran. Event cameras in automotive sensing: A review. *IEEE Access*, 2024. 1

[42] Radim Spetlik, Tereza Uhrová, and Jiří Matas. Efficient real-time quadcopter propeller detection and attribute estimation with high-resolution event camera. In *Scandinavian Conf. on Im. Analysis (SCIA)*. Springer, 2025. 2

[43] Simone Testa, Giacomo Indiveri, and Silvio P Sabatini. Dynamic detectors of oriented spatial contrast from isotropic fixational eye movements. SciTePress, 2020. 2, 3, 13

[44] Simone Testa, Silvio P Sabatini, and Andrea Canessa. Active fixation as an efficient coding strategy for neuromorphic vision. *Sci. Rep.*, 2023. 2, 3, 13

[45] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high speed scenarios. In *IEEE Robot. Autom. Lett.*, 2018. 1, 2

[46] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robot. Autom. Lett.*, 2018. 2

[47] Eric G Wu, Nora Brackbill, Colleen Rhoades, Alexandra Kling, Alex R Gogliettino, Nishal P Shah, Alexander Sher, Alan M Litke, Eero P Simoncelli, and EJ Chichilnisky. Fixational eye movements enhance the precision of visual information transmitted by the primate retina. *bioRxiv*, 2023. 1

[48] Kesheng Wu, Ekow Otoo, and Arie Shoshani. Optimizing connected component labeling algorithms. In *Im. Proc.* SPIE, 2005. 6

[49] Lan Xu, Weipeng Xu, Vladislav Golyanik, Marc Habermann, Lu Fang, and Christian Theobalt. Eventcap: Monocular 3d capture of high-speed human motions using an event camera. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[50] Amirreza Yousefzadeh, Garrick Orchard, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. Active perception with dynamic vision sensors. minimum saccades with optimum recognition. *IEEE Trans. Biomed. Circuits Syst.*, 2018. 2, 3

[51] Jiqing Zhang, Bo Dong, Haiwei Zhang, Jianchuan Ding, Felix Heide, Baocai Yin, and Xin Yang. Spiking transformers for event-based single object tracking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 2

[52] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, 1984. 6