

REACT: A Fully Spiking State-Space Model for Real-Time Event-Driven Temporal Perception

Geoffroy Keime
CerCo, CNRS UMR5549
Université de Toulouse
Toulouse, France
IPAL, CNRS IRL, Singapore
geoffroy.keime@cnrs.fr 

Nicolas Cuperlier
ETIS, CY Cergy Paris Université
CNRS, ENSEA
Cergy, France
IPAL, CNRS IRL, Singapore
nicolas.cuperlier@cyu.fr 

Benoit R. Cottureau
CerCo, CNRS UMR5549
Université de Toulouse
Toulouse, France
IPAL, CNRS IRL, Singapore
benoit.cottureau@cnrs.fr 

Abstract—Robotic systems operating in dynamic environments require visual perception that evolves continuously with the incoming sensory stream. Event cameras provide microsecond temporal resolution and asynchronous sensing, but most learning-based methods accumulate events into frames or temporal bins, introducing an integration delay that can limit fast reaction. Here we propose REACT, a fully spiking state-space model for event-driven temporal perception that processes raw events one by one, without temporal accumulation. REACT uses a complex-valued spiking neuron, C-SiLIF, whose continuous-time dynamics are driven by the physical inter-event interval, allowing its internal state to evolve at the temporal resolution of individual events. We evaluate REACT on gesture recognition and time-to-collision (TTC) estimation from full-field event streams, without a target bounding box or localization input. On EvTTC, REACT achieves a 9.59% relative TTC error with 4.6 ms end-to-end inference latency, within 0.15 percentage points of the best learned method while requiring no target prior. At the dataset’s mean approach speed, this latency corresponds to only 4 cm of vehicle motion, compared with 1 m for the fastest competing learned method. REACT further supports anytime TTC prediction, zero-shot transfer to a different driving sequence, and INT8 quantization, reducing the estimated energy consumption from 18.5 to 2.8 mJ per 32,768 events. These results show that event-driven spiking state-space dynamics can provide low-latency, continuously updated temporal perception for reactive robotic systems.

Index Terms—Event-based data, Spiking Neural Networks, State Space Models, Time-to-Contact, Real-time Processing

I. INTRODUCTION

Robotic systems operating in dynamic environments require visual perception that evolves continuously with the incoming sensory stream [1]. Event cameras provide microsecond temporal resolution and asynchronous sensing, making them particularly attractive for low-latency, asynchronous event-driven robotic perception. However, most learned event-based methods accumulate events into frames or temporal bins before inference, introducing a temporal integration delay and an artificial discretization of the sensor stream [1]–[3]. Such aggregation prevents the model from updating its state at the native event timescale and can limit timely perception in fast-moving environments.

Spiking neural networks (SNNs) naturally process event streams through sparse, asynchronous updates, while state-space models (SSMs) provide an efficient framework for modeling temporal dependencies through compact dynamical states. Recent works have combined these paradigms for event-based vision, including spiking formulations of SSMs and SSM-inspired spiking neurons, but they generally do not combine event-driven spiking computation with continuous-time state evolution directly at the irregular timestamps of incoming events. As a result, the temporal structure of the event stream remains partially constrained by discrete processing or temporal aggregation.

We propose REACT (Real-time Event-driven Asynchronous Computation for Temporal perception), a fully spiking, asynchronous event-driven state-space model that processes raw events sequentially as they arrive. REACT uses the complex-valued spiking neuron C-SiLIF, whose continuous-time dynamics are driven by the measured inter-event interval. The internal state is therefore updated event by event, with the elapsed time between consecutive events explicitly incorporated into the dynamics, without requiring a fixed temporal window or event accumulation. This enables REACT to preserve fine-grained temporal information while retaining the efficient linear recurrence of an SSM.

We evaluate REACT on two complementary temporal perception tasks: gesture recognition on DVS128 Gesture and time-to-collision (TTC) estimation on EvTTC. For TTC estimation, REACT processes the full field of view without a supplied target bounding box, requiring temporal collision-relevant information to emerge directly from the asynchronous event stream. This provides a direct test of asynchronous event-driven temporal perception in a dynamic driving scenario. We further investigate zero-shot transfer, anytime prediction, and INT8 quantization, assessing both the temporal flexibility and practical efficiency of the proposed approach.

The main contributions of this work are as follows:

- We introduce REACT, a new fully spiking state-space architecture for raw event streams, combining asynchronous event-by-event processing with continuous-time state evolution driven by the physical inter-event interval. This eliminates temporal accumulation while

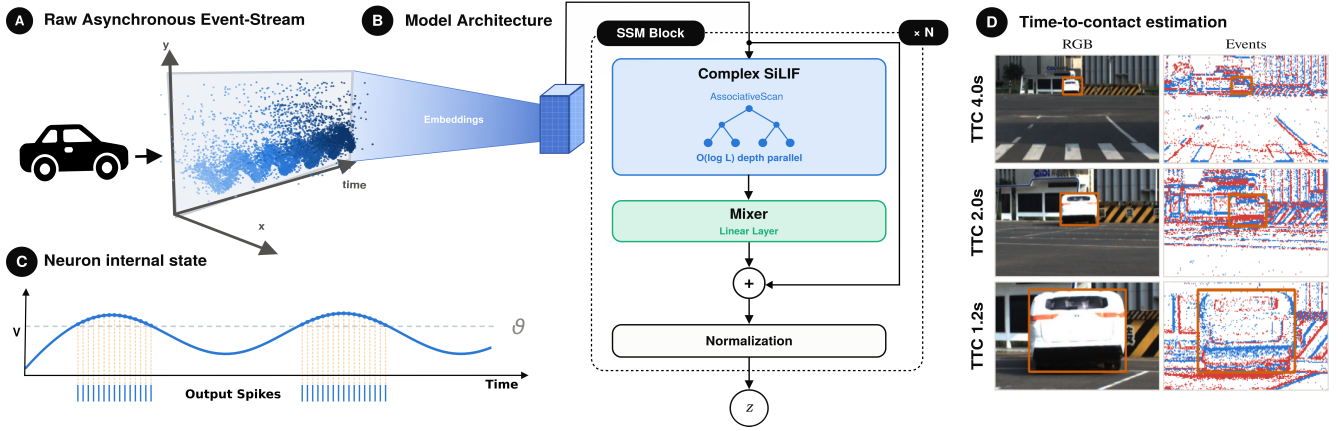


Fig. 1. **Overview of REACT.** (A) Vehicle motion generates an asynchronous stream of events. (B) Each event is embedded into a fixed-dimensional vector and processed sequentially by spiking state-space blocks, whose temporal dynamics are driven by the physical inter-event interval. (C) The C-SiLIF neurons emit binary spikes when their membrane potential crosses the threshold, without resetting the underlying state, thereby preserving continuous-time state dynamics. (D) Examples of event-based visual inputs at 4, 2, and 1.2 s before collision. Red and blue spikes correspond to *On* and *Off* events, respectively. The associated RGB frames are shown for illustration only and were not used by the model.

preserving continuous-time temporal dynamics at the native timescale of the sensor.

- We demonstrate target-free, anytime time-to-contact (TTC) estimation directly from full-field event streams, without temporal accumulation or a supplied target bounding box. REACT continuously updates its prediction as events arrive and achieves 4.6 ms end-to-end latency with a 9.59% relative TTC error on EvTTC.
- We show that the learned temporal representation supports both anytime prediction and zero-shot transfer across datasets, demonstrating its ability to provide useful temporal estimates under varying observation horizons and sensing conditions.
- We quantify the efficiency potential of the fully spiking formulation for neuromorphic deployment. REACT maintains competitive gesture-recognition accuracy with a substantially smaller state-space component, while INT8 quantization reduces the estimated TTC energy from 18.5 to 2.8 mJ per 32,768 events.

Code will be made publicly available upon publication.

II. RELATED WORKS

A. State Space Models and Spiking Neural Networks

Most current event-based vision models rely on an intermediate event representation, aggregating events into pseudo-frames through hand-designed or learned strategies. While effective, these representations discard part of the fine-grained timing provided by the sensor. Processing events directly preserves this temporal resolution, but requires architectures capable of efficiently modeling long asynchronous sequences. State space models (SSMs) have emerged as efficient sequence models, scaling linearly with sequence length while maintaining bounded computational and memory costs [4]–[6]. The linear ordinary differential equations underlying continuous-time SSMs are structurally analogous to the sub-threshold dynamics of leaky integrate-and-fire (LIF) neurons. This correspondence has motivated both spiking formulations

of SSMs [7], [8] and the derivation of neuron models from SSM parameterizations [9], including SSM-inspired adaptive LIF [10], [11] and resonate-and-fire neurons [12], [13]. SSMs have also been applied directly to raw event streams, avoiding frame-based representations [14]–[16]. These approaches efficiently extract spatio-temporal features from asynchronous data and offer pathways toward compute-in-memory implementations [16]. However, their computation remains fundamentally dense, requiring state updates across the model as events are processed. They therefore do not fully exploit the sparse event-driven activity that underpins the energy efficiency of neuromorphic computing. This limitation is particularly relevant for temporal prediction tasks in which low latency and fine-grained timing are critical, such as time-to-contact estimation.

B. Time-to-contact Estimation

Time-to-contact (TTC) denotes the remaining time before an object reaches the observer under its current relative motion. Unlike metric depth, TTC does not require recovering absolute scale or metric scene structure, making it a compact cue for reactive behaviors such as braking, landing, and obstacle avoidance [17]. Because informative visual changes become increasingly concentrated near the collision endpoint, latency is critical for accurate real-time TTC prediction [18]. Event cameras are therefore particularly well suited to TTC estimation through their asynchronous response, high temporal resolution, and high dynamic range. Event-based TTC estimation has traditionally relied on model-based approaches that fit explicit geometric or kinematic models to event streams [17], [19]–[23]. These methods typically recover the focus of expansion, motion parameters, or scene depth before deriving TTC. A smaller body of work preserves sensor asynchrony by updating a filter event by event [24], but still relies on a hand-specified observation model. Their performance is consequently constrained by assumptions such as prescribed motion patterns or planar

and rigid-translation models, as well as by iterative inference [25]. Learning-based TTC estimation remains comparatively underexplored, with only a few recent approaches [25], [26]. Garl-TTC [25] is currently the only dedicated object-level framework. To the best of our knowledge, the combination of fully spiking, event-by-event processing and explicit state-space temporal dynamics remains largely unexplored for TTC estimation.

III. METHODS

A. Event stream and input encoding

An event camera outputs an asynchronous stream of L tuples $\mathcal{E} = \{(t_k, x_k, y_k, p_k)\}_{k=1}^L$, where t_k is the timestamp, (x_k, y_k) the pixel coordinate on an $H \times W$ sensor, and $p_k \in \{0, 1\}$ the polarity of the brightness change. Most learning-based pipelines first accumulate this stream into frames, voxel grids, or fixed time bins. REACT consumes it directly: no event representation is constructed at any stage, and the model state is updated once per event. Each event is reduced to a spatial channel index:

$$j_k = p_k \cdot HW + y_k \cdot W + x_k \in \{0, \dots, 2HW - 1\}, \quad (1)$$

and a physical inter-event interval $\Delta t_k = t_k - t_{k-1}$, with $\Delta t_1 = 0$ by convention, intervals are expressed in milliseconds. The stream is thus a pair of aligned sequences $(j_k, \Delta t_k)$: j_k carries *where* the event occurred and Δt_k carries *when*, the latter entering the network as a continuous quantity rather than as a bin index. The channel index is mapped to a D -dimensional embedding $\mathbf{z}_k = \text{Embed}(j_k) \in \mathbb{R}^D$ followed by layer normalisation, through a learned lookup table $\mathbf{E} \in \mathbb{R}^{2HW \times D}$ that can be read as a per-pixel, per-polarity synaptic fan-in. Variable-length streams are right-padded and masked for batched training; padded positions contribute to neither the loss nor the reported firing rates. Fig. 1 summarises the resulting pipeline.

B. The C-SiLIF neuron

The core computational unit of our SNN is the complex-valued C-SiLIF neuron introduced by [9], whose subthreshold dynamics form a diagonal linear time-invariant state-space model with one complex state per channel, a log-parameterized decay, a learnable timestep, and S4D-Lin initialization [27]. Fabre et al. evaluate it on fixed-bin audio streams. We retain their formulation and drive the discretisation with the physical inter-event interval Δt_k rather than a fixed step, which is what makes event-by-event operation possible. Each of the D channels carries one independent state $u \in \mathbb{C}$, evolving as:

$$\frac{du}{dt} = \Lambda u + b x(t), \quad \Lambda = -e^{\lambda_{\text{re}}} + i \lambda_{\text{im}}, \quad b = b_{\text{re}} + i b_{\text{im}}, \quad (2)$$

where λ_{re} is a log-parametrized decay rate, λ_{im} an angular frequency, and b a complex input gain, all learned per neuron.

Discretisation rescales the measured inter-event interval by a per-neuron learnable factor e^ρ , giving an effective step

$\delta_k = e^\rho \Delta t_k$, so each neuron measures time in units of its own, and event-driven update.

$$\alpha_k = \exp((-e^{\lambda_{\text{re}}} + i \lambda_{\text{im}}) \delta_k), \quad (3)$$

$$u_k = \alpha_k u_{k-1} + b \mathbf{z}_k, \quad (4)$$

$$s_k = \Theta(2 \text{Re}(u_k) - \vartheta), \quad (5)$$

with Θ the Heaviside step function, ϑ a learnable firing threshold, and $s_k \in \{0, 1\}$ the emitted spike. Events enter as Dirac impulses at their timestamps, so the physical inter-event interval enters the recurrence only through the transition coefficient α_k . The factor of two follows from the real-valued observation of the conjugate-symmetric state pair, since $u + \bar{u} = 2 \text{Re}(u)$.

Importantly, no reset is applied to u_k after a spike. The state recurrence therefore remains linear, while the spiking nonlinearity acts only as a readout. Because the state transition coefficient α_k depends on the physical inter-event interval, the temporal dynamics are driven directly by the timing of incoming events. The dynamics are also stable by construction: $e^{\lambda_{\text{re}}} > 0$ implies $|\alpha_k| < 1$ for any positive interval. Equivalently, each neuron acts as a damped harmonic resonator with impulse response $|b| e^{-e^{\lambda_{\text{re}}} t} \cos(\lambda_{\text{im}} t + \varphi)$. The phase φ therefore controls the temporal phase of the neuron's oscillatory impulse response, enabling different neurons to preferentially respond to different temporal patterns without introducing an explicit delay parameter. Gradients are propagated through (5) with an arctangent surrogate of width 2.

Following the S4D-Lin initialisation adopted for C-SiLIF in [9], we set $e^{\lambda_{\text{re}}} = 0.5$ and $\lambda_{\text{im}} = \pi/s$ as shared initial values, sample e^ρ log-uniformly over 10^{-3} – 10^{-1} , and initialise the input gain as $b_{\text{re}} \sim \mathcal{U}(0, 1)$ and $b_{\text{im}} = 0$. The parameters λ_{re} and λ_{im} define the initial pole location, while the heterogeneous timestep ρ provides each neuron with a distinct temporal scale, and therefore a distinct effective decay and oscillation frequency. This diversity avoids starting all neurons from a homogeneous temporal configuration and provides a broad set of initial temporal responses for optimisation.

C. Sequential inference, parallel training

Equation (4) is a first-order linear recurrence $u_k = \alpha_k u_{k-1} + b_k$ whose transitions compose associatively,

$$(A_j, B_j) \circ (A_i, B_i) = (A_j A_i, A_j B_i + B_j), \quad (6)$$

so the state sequence $\{u_k\}_{k=1}^L$ can be obtained by an inclusive Hillis–Steele prefix scan in $O(\log L)$ sequential depth and $O(L)$ total work, parallelised across batch elements and channels. We use this scan during training and batched offline evaluation only. At deployment the model runs strictly sequentially, one state update per incoming event, which is the regime assumed by the latency and energy figures of Secs. IV-B and IV-F.

D. Network architecture

The network body is a stack of $N = 2$ residual blocks with $D = 128$ channels, operating on the token sequence produced by the embedding. Each block applies C-SiLIF independently along the temporal dimension of every channel, updating its state according to the inter-event intervals and emitting a binary spike train, and a channel-mixing linear stage then integrates information across features. The block output is combined with its input through a residual connection, which keeps optimisation stable as blocks are stacked. Every signal crossing a block boundary is therefore a spike train, and each C-SiLIF layer is driven by spikes throughout the network, so all synaptic operations reduce to accumulations.

Because sequence length grows with the number of incoming events, consecutive tokens are merged between blocks by masked-mean pooling, with their inter-event intervals summed so that the compressed sequence still represents the correct elapsed physical time. We pool by a factor of 32 on DVS128-Gesture and apply no pooling on EvTTC, which is the source of the higher per-event energy reported in Sec. IV-F.

After the last block, tokens are masked-mean-pooled over valid positions and mapped by a single linear layer to the task output. On neuromorphic substrates this readout corresponds to a non-firing integrator, whose membrane potential is read out directly.

E. Task formulation and metrics

For gesture recognition the readout produces a distribution over the 11 classes, trained with cross-entropy and label smoothing.

For time-to-contact we regress the inverse contact time $\eta = 1/\text{TTC}$ rather than the time itself, since TTC diverges under slow approach while η remains bounded. The readout emits a mean μ_i and a log-variance s_i in $\log \eta$ space, trained with a Gaussian negative log-likelihood,

$$\mathcal{L}_{\text{NLL}} = \frac{1}{N} \sum_{i=1}^N \left[\frac{1}{2} e^{-s_i} (\mu_i - \log \eta_i)^2 + \frac{1}{2} s_i \right], \quad (7)$$

so that $\hat{\eta} = e^\mu$ and $\widehat{\text{TTC}} = e^{-\mu}$, with s_i providing a per-estimate uncertainty alongside the prediction.

We report the EvTTC benchmark metric [28],

$$e_{\text{TTC}} = \frac{|\widehat{\text{TTC}} - \text{TTC}|}{\text{TTC}} \times 100\%, \quad (8)$$

averaged per sequence over its full range, so that every row of Table II is scored identically. Because (8) normalises by ground-truth time, a fixed absolute error counts for little when the target is far and heavily when it is close, and a sequence average is dominated by its far portion. We therefore also report a criticality-weighted log-ratio error, MiD_w , which averages $|\log \hat{\eta} - \log \eta|$ within ground-truth TTC bands and reweights them towards imminent contact, with weights 0.5, 0.3 and 0.1 on $[0, 3)$, $[3, 6)$ and $[6, 10)$ s following [25]. We exclude their negative-scenario term,

since the sequences evaluated here contain no non-collision cases. This is the quantity on which models are selected, and it scores the part of the sequence that matters for collision avoidance rather than the part that dominates (8).

F. Datasets and training

DVS128-Gesture [29] contains 11 hand and arm gestures from 29 subjects under three lighting conditions, recorded at 128×128 , giving an embedding table of $2 \times 128 \times 128$ rows. We use the official split. Training windows are contiguous slices of 131,072 events. Augmentation follows the set used by [14]: time skew, per-event temporal and spatial jitter, and random event dropping, together with the geometric augmentations of [30] (roll, rotation, scaling) and event CutMix [31]. Horizontal flipping is excluded, since mirroring exchanges gesture classes.

EvTTC [28] contains collision scenarios with cars and pedestrian dummies recorded with Prophesee cameras at 1280×720 , with ground-truth TTC derived from synchronised GNSS and LiDAR. Unlike prior work on this benchmark, we supply no target bounding box: REACT predicts from the full field of view. Event coordinates are downsampled by a factor of 8 onto a 160×90 grid, with a 10 ms refractory period per coarse pixel and polarity to suppress duplicates; the embedding table therefore contains $2 \times 90 \times 160$ rows, giving 3.7 M parameters in total. Training windows contain 32,768 tokens, obtained by decimating a contiguous span by a factor of 16, roughly 0.8 s of stream. Augmentation comprises horizontal flipping, per-event spatial jitter, random event dropping, injected clutter events, and a time skew that rescales the stream, which simulates a different approach speed. We additionally evaluate zero-shot transfer on FCWD [20] without fine-tuning.

Models are implemented in SpikingJelly [32] and trained with AdamW, with weight decay disabled on the state-space parameters. We report three seeds on DVS128-Gesture and five on EvTTC.

IV. EXPERIMENTS

A. Gesture recognition

We first evaluate the architecture on a standard neuromorphic classification benchmark, before turning to the continuous temporal prediction task of Sec. IV-B. DVS128-Gesture is the benchmark on which Event-SSM and its compute-in-memory counterpart both report results, so it measures whether a fully spiking SSM reaches the accuracy of comparable event-by-event models at a much smaller cost. Table I compares the three. Event-SSM [14] and the compute-in-memory implementation [16] achieve accuracies of 97.7% and 97.3%, respectively, but propagate real-valued activations between layers, requiring non-spiking computation and therefore limiting their direct deployment on spike-based neuromorphic hardware. REACT achieves a comparable accuracy of 97.6% while using 26% fewer parameters for the SSM blocks and maintaining fully spiking computation throughout the network.

TABLE I

DVS128-GESTURE. PARAMETER COUNTS ARE REPORTED AS EMBEDDING + STATE-SPACE, FOLLOWING [14]; THE EMBEDDING IS FIXED BY THE SENSOR AND IDENTICAL ACROSS ALL THREE MODELS. ACTIVITY DENOTES THE FRACTION OF UNITS ACTIVE PER UPDATE. *ENERGY IS ESTIMATED USING THE PROXY DESCRIBED IN SEC. IV-F, APPLIED TO SIX DENSE REAL-VALUED SSM LAYERS. BEST RESULTS ARE IN **BOLD**; SECOND-BEST RESULTS ARE UNDERLINED.

Method	Spiking	Params (M)	Prec.	Memory (MB)	Energy (mJ)	Activity (%)	Acc. (%)
Event-SSM [14]	✗	4.19 + 0.80	FP32	20.00	158.8*	100	97.7
CIM-SSM [16]	✗	4.19 + 0.81	INT8	<u>5.00</u>	<u>23.7*</u>	100	97.3
REACT (ours)	✓	4.19 + 0.04	FP32	16.14	50.6	<u>31.1</u>	<u>97.6</u>
REACT (ours)	✓	4.19 + 0.04	INT8	4.04	7.7	31.1	97.2

Energy is calculated from 131,072 events for REACT

B. Time-to-contact

The asynchronous nature of REACT is particularly relevant for tasks that require continuous feedback from a rapidly evolving sensory stream. We therefore apply the model to event-based time-to-contact (TTC) estimation, which provides a direct test of its ability to continuously update a temporal prediction from incoming events. Most existing approaches to event-based TTC estimation first aggregate events into pseudo-frames or voxel grids before performing the prediction. While these representations can achieve good estimation accuracy, they introduce an additional temporal latency: events must first be accumulated over a predefined temporal interval before the network can process them. Table II compares REACT with published results on EvTTC. Every prior entry receives the target as a tracked or ground-truth bounding box, and Garl-TTC [25] additionally uses frames alongside events; REACT predicts from the raw stream over the full field of view. Under that handicap it reaches $9.59 \pm 0.74\%$, within 0.15 percentage points of the best learned method. Parameter counts are not directly comparable across rows: REACT’s are dominated by the per-pixel embedding table, which replaces a convolutional front-end rather than supplementing it. The frame-based focus-of-expansion baseline [33] reaches a lower error still, but operates offline on rectified frames with the target given and reports no inference time, so it does not constrain a real-time budget.

Accumulation latency becomes critical when the platform moves rapidly. Conventional approaches must integrate events over a finite temporal interval before producing a prediction, during which the vehicle continues to move. This creates an unavoidable gap between when visual information is acquired and when the corresponding estimate becomes available. Event-by-event processing avoids this integration delay: REACT updates its internal state and TTC estimate as events arrive, so the sensing-to-prediction latency is determined primarily by computation. Figure 3 quantifies this difference. For frame- and window-based methods, overall latency is dominated by the integration window rather than network inference, resulting in the totals reported in Table II: 113 ms for Garl-TTC [25], 115 ms for ETTCM [23], and at least 3644 ms for CMax [19]. In contrast, REACT produces

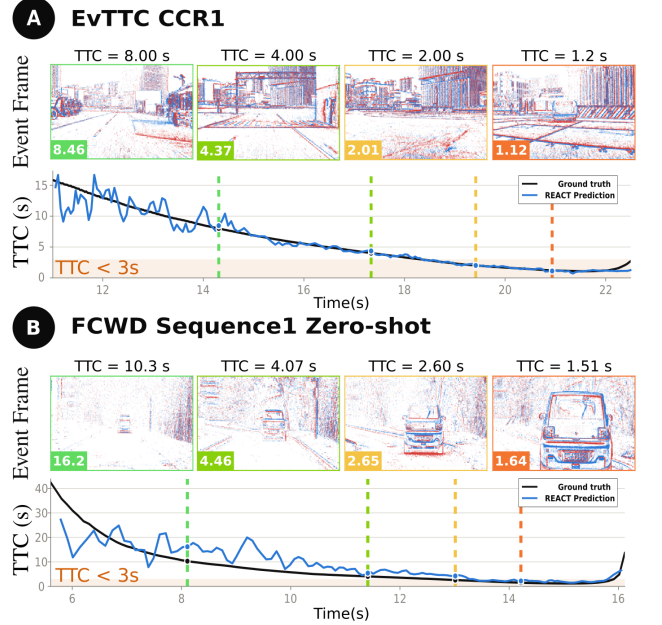


Fig. 2. **Qualitative TTC predictions over complete sequences.** (A) In-domain evaluation on the CCR1 split of the medium-overlap-50 set of EvTTC [28]. (B) Zero-shot evaluation on Sequence 1 of FCWD [20]. For each sequence, four event frames are shown at decreasing time to contact. The values above each frame indicate the ground-truth TTCs, while the colored labels in the lower-left corners show the corresponding REACT predictions (in seconds). Predictions progressively converge toward the ground truth as the time to contact decreases, with reliable estimates obtained within approximately 3 s of impact (see the critical zone in orange). REACT exhibits similar temporal prediction behavior on the unseen FCWD sequence under the domain shift, without fine-tuning. In this cases, the average vehicle speed is 9.22 m/s.

an updated estimate with an end-to-end latency of only 4.6 ms. At the mean approach speed of 9.22 m/s measured near the target in EvTTC, the vehicle travels only 4 cm during this interval, compared with more than 1 m for the fastest of the window-based alternatives.

1) *Zero-shot transfer:* To test whether the learned temporal representation is specific to EvTTC, we evaluate the same trained model on FCWD [20] without fine-tuning. FCWD was recorded with a different sensor and under different driving conditions, so the transfer is a test of the representation rather than of the training distribution.

TABLE II

EVTTTC TIME-TO-COLLISION. ALL PRIOR METHODS RECEIVE THE TARGET AS A TRACKED OR GROUND-TRUTH BOUNDING BOX, WHEREAS REACT PREDICTS DIRECTLY FROM THE RAW EVENT STREAM WITHOUT A TARGET PRIOR, USING THE IN-DOMAIN VALIDATION SPLIT OF [28]. INFERENCE TIME DENOTES THE END-TO-END LATENCY REPORTED IN FIG. 3; $\geq$ INDICATES METHODS THAT DO NOT REPORT AN ACCUMULATION WINDOW. *ENERGY IS ESTIMATED USING THE PROXY DESCRIBED IN SEC. IV-F, APPLIED TO GARL-TTC’S RESNET-50 BACKBONE; NO PRIOR METHOD REPORTS ENERGY. BEST RESULTS PER COLUMN ARE IN **BOLD**.

Method	Category	Modality	Params	Prec.	e_{TTC} (%)	Target-free	Async.	Time (ms)	Energy (mJ)
Image’s FoE [33]	Geometric	Frame	-	-	3.51	$\times$	$\times$	-	-
CMax [19]	Geometric	Event	-	-	9.11	$\times$	$\times$	≥ 3644	-
Garl-TTC [25]	Learned	Event + Frame	23.5M*	FP32	9.44	$\times$	$\times$	113	197*
STRITC [20]	Geometric	Event	-	-	11.22	$\times$	$\times$	≥ 25	-
FAITH [22]	Geometric	Event	-	-	40.97	$\times$	$\times$	≥ 155	-
ETTCM [23]	Geometric	Event	-	-	50.38	$\times$	$\times$	115	-
REACT (ours)	Learned	Event	3, 7M	FP32	9.59 ± 0.74	$\checkmark$	$\checkmark$	4.6	18.5
REACT (ours)	Learned	Event	3, 7M	INT8	10.26 ± 0.81	$\checkmark$	$\checkmark$	4.6	2.8

Energy is calculated from 32,768 events for REACT.

*Estimated parameter count with a ResNet-50.

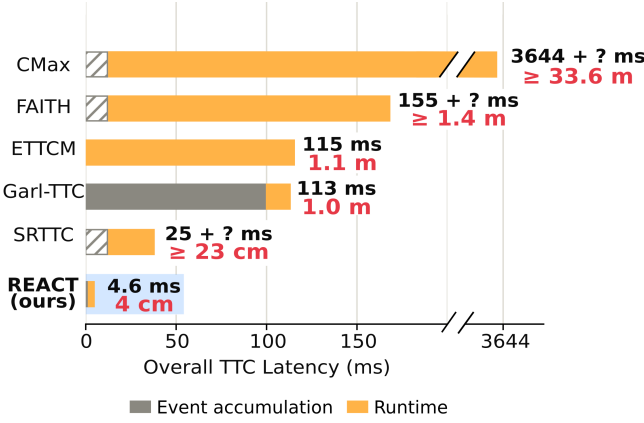


Fig. 3. **Latency breakdown of time-to-contact (TTC) methods**, including event accumulation (grey) and runtime (orange). Diagonal hatching indicates methods for which the event acquisition time is not reported; in these cases, only a lower bound on the overall latency can be provided. Red annotations indicate the corresponding distance travelled during this latency, computed using the mean vehicle speed near the target in the EvTTC dataset [28] (9.22 m/s).

Table III reports the resulting errors alongside the in-domain figures, and Fig. 2 shows complete predicted trajectories in both settings. The behaviour is consistent across the domain shift: predictions are unreliable while the target is distant, where the expansion signal in the event stream is weak, and converge as the collision approaches. The error therefore concentrates in the region that (8) weights most heavily and that matters least operationally, which is what motivates MiD_w (Sec. III-E) as the selection criterion.

C. Anytime accuracy

Because no accumulation window is required, REACT can be queried at any instant, including instants at which it has seen very little of the stream. To measure how much stream it needs, we empty the state of a trained checkpoint δ ms before a query, let the model consume only the events of that interval, and record the median error at the query.

TABLE III

ZERO-SHOT TRANSFER TO FCWD [20]. THE SAME MODEL TRAINED ON EVTTTC IS EVALUATED ON FCWD WITHOUT FINE-TUNING OR DOMAIN ADAPTATION.

Evaluation	$e_{TTC} \downarrow$ (%)	$MiD_w \downarrow$
EvTTC (in-domain)	9.322	0.0986
FCWD (zero-shot)	25.7	0.246

An estimate is usable long before a frame-based pipeline could assemble its input: 27% median error from 100 ms of stream, which is the earliest instant at which a two-frame method such as Garl-TTC [25] has its second frame, and 20% from 200 ms. Accuracy peaks at 12.9% around 400 ms and then degrades to 21%. By not having any form of reset in the internal state and the readout of the model, its predictions are refined as times goes on, making REACT suitable for instant prediction and high-horizon predictions (see Figure 2 to see model stabilization of prediction in function of the time)

D. Network quantization

To implement SNNs on dedicated neuromorphic hardware, the network needs to be quantized (usually on from full-precision 32-bits (FP32) to integer 8 bits (INT8). Following a similar methodology as in [16], we use quantization-aware training. From a converged FP32 model we fine-tune the model for 150 epochs by converting only the synaptic weights and the neuron parameters to INT8 as the activation functions are already binary. By clamping the weights in INT8 and making the update and the full precision weights. This methods allows the model to fine tune progressively using full-precision copy for weight update.

Over three different seeds, this gives $97.2 \pm 0.4\%$ against $97.6 \pm 0.3\%$ in FP32 on DVS-Gesture dataset.

E. Ablation Study

1) *TTC Full-Frame:* Every published method on EvTTC receives the target as a bounding box, either provided by

TABLE IV
NEURON MODEL ON EVTTC. ALL VARIANTS SHARE THE
ARCHITECTURE, EVENT BUDGET AND TRAINING SCHEDULE OF
SEC. III-D; ONLY THE NEURON DIFFERS. BEST IN **BOLD**, SECOND
UNDERLINED.

Neuron	State	e_{TTC} (%) ↓	MiD _w ↓
LIF	real, fixed decay, reset	116.9	0.681
LIF, no reset	real, fixed decay	84.1	0.52
SiLIF [9]	real, learned decay	<u>11.42</u>	<u>0.12</u>
C-SiLIF [9]	complex, oscillatory	9.3	0.09

a detector and tracker or obtained from ground truth. This removes distractors from the field of view and reduces TTC estimation to the temporal evolution of a known target region [25]. While this setup is standard and effective, it introduces a separate localization stage before TTC estimation and therefore does not provide a fully asynchronous end-to-end pipeline. To disentangle the contributions of target localization and temporal estimation, we evaluate the same REACT configuration both on the full field of view and on target crops. On the held-out sequences, the TTC error decreases from 34.2% with the full frame to 28.3% with target crops, indicating that a substantial part of the gap between REACT and published results stems from target localization rather than temporal estimation. This also suggests that incorporating an ROI front-end could improve performance without modifying the underlying spiking temporal model.

2) *Neuron model*: The temporal dynamics of REACT strongly rely on its neuron model, making neuron replacement the natural ablation while keeping the rest of the architecture fixed. We compare C-SiLIF against SiLIF [9], which retains the log-reparameterized decay and learnable timestep but remains real-valued and therefore lacks an oscillatory mode, and against a standard LIF neuron, whose dynamics are governed by a single fixed decay constant. All variants are trained on EvTTC with the same two residual blocks, $D = 128$ channels, event budget, and training schedule. We evaluate LIF both in its standard form, with a spike-triggered reset, and with the reset removed as described in Sec. III-B. This isolates the contribution of the linear recurrence from that of the neuron dynamics themselves.

The results are reported in Table IV. Removing the reset alone reduces the e_{TTC} from 116.9% to 84.1%, showing that preserving the state across spikes is already critical for temporal prediction. With a spike-triggered reset, the neuron repeatedly discards its accumulated temporal state, severely limiting its ability to represent long-range temporal dependencies. Both LIF variants nevertheless remain far from usable in our experimental setting, highlighting the importance of the SSM-like temporal dynamics. Replacing LIF with SiLIF reduces the error to 11.42%, while C-SiLIF further reduces it to the reported 9.3% on EvTTC. The distance traveled during the latency, and thus the effective blind distance, increases linearly with vehicle speed. At 100 km/h, these latencies correspond to only 12.8 cm of blind

travel for REACT, compared with 3.14 m for Garl-TTC and 3.19 m for ETTCM, highlighting the practical advantage of our approach.

F. Energy consumption

We estimate inference energy for a neuromorphic deployment using the 45 nm energy proxy proposed in [34]. For each layer, the estimated energy is computed as $(\text{reads} + \text{writes})E_{\text{SRAM}} + \text{acc } E_{\text{add}}$, where the number of synaptic operations is determined by the number of input spikes, θ_{in} , measured on real data. We additionally account for two costs specific to our architecture. First, the event embedding requires D word reads per input event. Second, the C-SiLIF core reads five per-channel parameters $(\lambda_{\text{re}}, \lambda_{\text{im}}, \rho, b_{\text{re}}, b_{\text{im}})$ at each update, compared with a single decay parameter for a conventional LIF neuron, and requires 12 multiplications and 5 additions per event.

For energy estimation, we consider the sequential deployment of the model, since the parallel associative scan described in Sec. III is used only for training and batched sequence processing.

Firing rates are measured independently for each block. On DVS128-Gesture, the overall firing rate is $31.1 \pm 1.3\%$ across three seeds, with $31.9 \pm 1.3\%$ and $6.3 \pm 0.4\%$ in blocks 0 and 1, respectively. On EvTTC, the overall firing rate is $23.2 \pm 1.2\%$ across five seeds. INT8 quantization does not alter the binary spike activity and therefore leaves the firing rates unchanged. We consequently use the same measured rates for the FP32 and INT8 energy estimates, with the difference in energy arising solely from the reduced per-operation cost.

The resulting estimated energy per input window is reported in Tables I and II. Channel-mixing linear layers account for approximately 90% of the total estimated energy, while the C-SiLIF state-space core represents the lowest-cost major component. The estimated energy per event is $2.7\times$ higher for EvTTC than for DVS-Gesture, primarily because event pooling is not used for EvTTC. For EvTTC, where the input windows are bounded in time, the estimated energy scales linearly with the number of events at a fixed firing rate, yielding an estimated cost of 563 nJ per event.

V. DISCUSSION

We introduced REACT, a new fully spiking state-space model that processes raw event streams event by event, without frames, voxel grids, or fixed temporal bins. By combining binary neural activity with continuous-time state-space dynamics driven by physical inter-event intervals, REACT retains temporal information while avoiding explicit temporal discretization. On DVS128-Gesture, it achieves competitive accuracy with event-by-event non-spiking SSMs while using a substantially smaller state-space component, demonstrating that rich temporal dynamics can be implemented with a compact spiking formulation.

The TTC experiments illustrate the relevance of this approach for robotic perception. REACT achieves $9.59 \pm 0.74\%$ error on EvTTC, comparable to the best reported

learning-based method, while requiring no target bounding box as input. More importantly, processing events individually yields a 4.6 ms end-to-end inference latency, avoiding the accumulation delays introduced by frame- or window-based approaches. The zero-shot experiment further indicates that the learned temporal representation can transfer across datasets, although performance degrades for distant targets.

The energy analysis also provides insight into the architecture. INT8 quantization causes only a limited accuracy degradation and reduces the estimated EvTTC energy to 2.8 mJ per 32,768 events. However, approximately 90% of the estimated energy is spent in channel-mixing layers rather than in the C-SiLIF state-space core. This suggests that further sparsification of these operations may offer greater efficiency gains than optimizing the temporal dynamics themselves. These values remain analytical estimates and should therefore be validated on neuromorphic hardware.

Finally, the embedding remains a major source of parameters and scales with sensor resolution. Factorized or coordinate-based embeddings, along with training on more diverse motion scenarios, are promising directions for improving scalability and robustness. Evaluation on physical neuromorphic hardware and robotic platforms will be an important step toward deployment.

VI. CONCLUSION

REACT places the dynamics of a state-space model inside the spiking neuron. Processing events asynchronously brings learned models closer to anytime prediction, where an estimate is available whenever it is needed rather than once a window has closed. Nothing in the architecture is specific to time-to-contact: any task that maps a continuous event stream to a continuously updated estimate, such as steering-angle prediction, ego-motion estimation or place recognition, would be a natural fit for future ongoing works. Ultimately, such event-driven anytime perception could provide the low-latency feedback required to connect asynchronous vision directly to closed-loop robotic control.

ACKNOWLEDGMENTS

We acknowledge the use of ChatGPT and Claude for assistance with language editing, clarity, and phrasing throughout the manuscript. Claude was also used for coding assistance, including Triton implementations for accelerated GPU training, factorization, and plot generation.

REFERENCES

- [1] G. Gallego *et al.*, “Event-based vision: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022.
- [2] D. Gehrig *et al.*, “End-to-end learning of representations for asynchronous event-based data,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 5632–5642.
- [3] X. Lagorce *et al.*, “Hots: A hierarchy of event-based time-surfaces for pattern recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1346–1359, 2017.
- [4] A. Gu, K. Goel, and C. Ré, “Efficiently Modeling Long Sequences with Structured State Spaces,” Aug. 2022.
- [5] J. T. H. Smith, A. Warrington, and S. W. Linderman, “Simplified State Space Layers for Sequence Modeling,” Mar. 2023.
- [6] A. Gu and T. Dao, “Mamba: Linear-Time Sequence Modeling with Selective State Spaces,” 2024.
- [7] M. Bal and A. Sengupta, “Rethinking Spiking Neural Networks as State Space Models,” Jun. 2024.
- [8] M. I. Stan and O. Rhodes, “Learning Long Sequences in Spiking Neural Networks,” Dec. 2023.
- [9] M. Fabre, L. Dudchenko, and E. Neftci, “SiLIF: Structured State Space Model Dynamics and Parametrization for Spiking Neural Networks,” Feb. 2026.
- [10] A. Bittar and P. N. Garner, “A surrogate gradient spiking baseline for speech command recognition,” *Frontiers in Neuroscience*, vol. 16, Aug. 2022.
- [11] M. Baronig *et al.*, “Advancing spatio-temporal processing in spiking neural networks through adaptation,” *arXiv preprint arXiv:2408.07517*, 2024.
- [12] E. M. Izhikevich, “Resonate-and-fire neurons,” *Neural Networks*, vol. 14, no. 6, pp. 883–894, 2001.
- [13] S. Higuchi *et al.*, “Balanced resonate-and-fire neurons,” *arXiv preprint arXiv:2408.14603*, 2024.
- [14] M. Schöne *et al.*, “Scalable event-by-event processing of neuromorphic sensory signals with deep state-space models,” in *2024 International conference on neuromorphic systems (ICONS)*. IEEE, 2024, pp. 124–131.
- [15] N. Zubic, M. Gehrig, and D. Scaramuzza, “State space models for event cameras,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 5819–5828.
- [16] X. Zhang *et al.*, “Compute-in-memory implementation of state space models for event sequence processing,” *Nature Communications*, vol. 17, no. 1, p. 1513, Jan. 2026.
- [17] X. Clady *et al.*, “Asynchronous visual event-based time-to-contact,” *Frontiers in Neuroscience*, vol. 8, 2014.
- [18] D. Falanga, K. Kleber, and D. Scaramuzza, “Dynamic obstacle avoidance for quadrotors with event cameras,” *Science Robotics*, vol. 5, no. 40, p. eaaz9712, 2020.
- [19] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [20] J. Li *et al.*, “Event-aided time-to-collision estimation for autonomous driving,” in *Computer Vision – ECCV 2024*, A. Leonardis *et al.*, Eds. Cham: Springer Nature Switzerland, 2025, pp. 57–73.
- [21] S. McLeod *et al.*, “Globally optimal event-based divergence estimation for ventral landing,” in *European Conference on Computer Vision*. Springer, 2022, pp. 3–20.
- [22] R. Dinaux *et al.*, “Faith: Fast iterative half-plane focus of expansion estimation using optic flow,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7627–7634, 2021.
- [23] U. M. Nunes, L. U. Perrinet, and S.-H. Ieng, “Time-to-contact map by joint estimation of up-to-scale inverse depth and global motion using a single event camera,” in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 23 596–23 606.
- [24] Z. Wang *et al.*, “Asynchronous blob tracker for event cameras,” *IEEE Transactions on Robotics*, vol. 40, pp. 4750–4767, 2024.
- [25] J. Li *et al.*, “Toward deep representation learning for event-enhanced visual autonomous perception: The eap dataset,” *IEEE Transactions on Robotics*, vol. 42, pp. 1643–1661, 2026.
- [26] A. Bisulco, V. Kumar, and K. Daniilidis, “Ev-ttc: Event-based time to collision under low light conditions,” *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 6151–6158, 2025.
- [27] A. Gu *et al.*, “On the parameterization and initialization of diagonal state space models,” *Advances in neural information processing systems*, vol. 35, pp. 35 971–35 983, 2022.
- [28] K. Sun *et al.*, “EvTTC: An event camera dataset for time-to-collision estimation,” *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 6191–6198, 2025.
- [29] A. Amir *et al.*, “A low power, fully event-based gesture recognition system,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7243–7252.
- [30] Y. Li *et al.*, “Neuromorphic data augmentation for training spiking neural networks,” in *Computer Vision – ECCV 2022*, S. Avidan *et al.*, Eds. Cham: Springer Nature Switzerland, 2022, pp. 631–649.
- [31] S. Yun *et al.*, “Cutmix: Regularization strategy to train strong classifiers with localizable features,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6022–6031.

- [32] W. Fang *et al.*, “SpikingJelly: An open-source machine learning infrastructure platform for spike-based intelligence,” *Science Advances*, vol. 9, no. 40, 2023.
- [33] S. Stabinger, A. Rodríguez-Sánchez, and J. Piater, “Monocular obstacle avoidance for blind people using probabilistic focus of expansion estimation,” in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.
- [34] E. Lemaire *et al.*, “An analytical estimation of spiking neural networks energy efficiency,” in *Neural Information Processing*, M. Tanveer *et al.*, Eds. Cham: Springer International Publishing, 2023, pp. 574–587.