SDformerFlow: Spiking Neural Network Transformer for Event-based Optical Flow

Yi Tian and Juan Andrade-Cetto

Institut de Robòtica e Informàtica Industrial, CSIC-UPC, Barcelona, Spain
`ytian@iri.upc.edu, cetto@iri.upc.edu`

Abstract. Event cameras produce asynchronous and sparse event streams capturing changes in light intensity. Overcoming limitations of conventional frame-based cameras, such as low dynamic range and data rate, event cameras prove advantageous, particularly in scenarios with fast motion or challenging illumination conditions. Leveraging similar asynchronous and sparse characteristics, Spiking Neural Networks (SNNs) emerge as natural counterparts for processing event camera data.
Recent advancements in Visual Transformer architectures have demonstrated enhanced performance in both Artificial Neural Networks (ANNs) and SNNs across various computer vision tasks. Motivated by the potential of transformers and spikeformers, we propose two solutions for fast and robust optical flow estimation: STTFlowNet and SDformerFlow. STTFlowNet adopts a U-shaped ANN architecture with spatiotemporal Swin transformer encoders, while SDformerFlow presents its full spike counterpart with spike-driven Swin transformer encoders.
Notably, our work marks the first utilization of spikeformer for dense optical flow estimation. We conduct end-to-end training for both models using supervised learning on the DSEC-flow Dataset. Our results indicate comparable performance with state-of-the-art SNNs and significant improvement in power consumption compared to the best-performing ANNs for the same task.
Our code will be open-sourced at https://github.com/yitian97/SDformerFlow.

Keywords: Spiking Neural Network · Event camera · Transformer · Optical flow.

1 Introduction

Event cameras have a higher temporal resolution compared to traditional cameras, capturing per-pixel intensity changes. The sparse and asynchronous event streams they produce can directly encode apparent motion patterns (optical flow), enabling accurate motion estimation in challenging scenarios like fast motion or low illumination. However, due to the fundamentally different data throughput of the two camera types, estimating event-based optical flow suggests different approaches than conventional computer vision methods. Recent research utilizing Artificial Neural Networks (ANNs) has demonstrated higher accuracy [10, 19] compared to model-based methods [28, 26] for event-based optical flow estimation.

ANN layers rely on floating-point calculations and may not fully exploit the sparse and asynchronous nature of event data. Spiking Neural Networks (SNNs) have emerged as a promising match for event data. In SNNs, neurons integrate input spike trains and generate a binary spike when the membrane potential reaches a threshold, resetting its value afterward. Neurons are active only when spikes arrive, just as individual event camera pixels are active only when intensity changes. Sharing this event-driven characteristic makes SNNs an energy-efficient option for processing event data. However, directly training deep SNNs is challenging due to the non-differentiability of the spike activity. The backpropagation through time with surrogate gradient method [25] has bridged neuromorphic computing with the deep learning community, enabling the training of deeper SNNs. Despite this advancement, the performance of SNNs still lags behind that of ANN methods.

The Visual Transformer (ViT) and its variant architectures have garnered increasing interest as potential replacements for convolution networks in various computer vision tasks in recent years. Traditional convolution-only models struggle to capture temporal correlation and efficiently represent global spatial dependencies due to their inherent locality [4, 11, 29, 15]. The integration of ViT, particularly with spatiotemporal attention, has also shown promising results in event-based vision tasks, such as monocular depth estimation [42] and action recognition [1, 8]. Combining SNNs with the ViT architecture for event cameras appears to be a natural choice, promising both performance and energy efficiency. Moreover, the self-attention mechanism in transformers also shares a biological background with SNNs [43, 40, 39, 34]. While early work proposing the Spikeformer architectures primarily validates their efficacy on event-based classification tasks [43, 39], their application in event-based regression tasks remains limited [47].

Solutions for event-based optical flow are primarily dominated by correlation-based methods [30, 10, 19], which require substantial computation and memory resources to compute the pixel-wise correlation volume. Integrating transformers into optical flow models has shown superior performance compared to non-transformer models in conventional computer vision, particularly excelling in scenarios involving large displacements due to their ability to capture global dependencies effectively [15, 36, 23]. While some works have utilized transformer architectures for event camera optical flow estimation tasks [18, 32], no one has proposed a pure SNN architecture, specifically utilizing spikeformer, for event-based optical flow estimation.

In this work, we introduce SDformerFlow, an SNN employing spiking spatiotemporal Swin transformers. Additionally, for better comparison, we propose STTFlowNet, an ANN counterpart to our SNN model. We conduct end-to-end training using supervised learning on the DSEC-flow Dataset. Our work marks the first instance of utilizing spikeformer for optical flow estimation, demonstrating comparable performance to state-of-the-art SNN optical flow estimation while significantly reducing energy consumption compared to the baseline model. Our contributions are threefold: Firstly, we introduce STTFlowNet, a Swin transformer-based model for event-based optical flow estimation, equipped with spatiotemporal self-attention to capture dependencies in both the time and space domains. Secondly, we present the spiking version of our architecture, SDformerFlow, marking the first known utilization of spikeformer for event-based optical flow estimation. Lastly, we conduct extensive experiments on datasets and compare them with baseline models, uncovering the potential of combining transformers with SNNs for regression tasks.

2 Related work

2.1 Learning-based methods for event-based optical flow estimation

Drawing inspiration from frame-based optical flow techniques, the estimation of event-based optical flow using deep learning has achieved state-of-the-art performance compared to model-based methods [10, 26, 28]. Early works predominantly employed a U-Net architecture [46, 17, 12, 3] to predict sparse flow and evaluated it using masks due to limited accuracy where no events are present. Inspired by RAFT flow [30], Gehrig et al. [10] proposed E-RAFT and contributed the Dense Stereo Event Camera (DSEC) Optical Flow Benchmark [9]. Since then, methods based on recurrent neural networks with correlation features and iterative refinement strategies have yielded state-of-the-art performance [10, 3, 18].

Recent studies have shifted their focus towards enhancing the temporal continuity of optical flow estimation, aiming to fully leverage the low latency characteristics of event cameras [35, 19, 27], or integrating richer simulated training datasets [18, 37] to improve accuracy. However, these recurrent refinement methods implicate calculating computationally expensive cost columns and an iterative update scheme that brings latency to the inference phase.

Another line of work based on SNNs emerges as a computationally efficient solution for event camera optical flow estimation. Early works trained SNNs using self-supervised learning, yielding sparse flow estimation [12]. More recent efforts involve training SNNs using supervised learning with U-Net architectures and trained with surrogate gradient on the DSEC dataset, resulting in dense flow estimation [2, 16]. To incorporate longer temporal correlations into the SNN model, some works utilize adaptive neural dynamics in comparison with event inputs containing richer temporal information [16], while others introduce external recurrence [27]. In [2], the authors employed 3D convolutions with stateless spiking neurons, neglecting the intrinsic temporal dynamics of the neurons. However, the performance of SNNs still falls behind that of ANNs. While some ANN methods incorporate transformer architectures in some of their stages [32, 18] and show performance improvements, to the best of our knowledge, this is the first time that SNNs are combined with a transformer architecture for optical flow estimation.

2.2 Spikeformer

Recently, the combination of SNNs and transformer architectures has garnered increasing interest for other tasks in the neuromorphic community [43, 40, 38]. Zhou et al. [43] initially proposed spiking self-attention, which eliminates the softmax function as the spike-formed query and key naturally maintains non-negativity. Building upon this, Yao et al. [39] introduced a fully spike-driven transformer with spike-driven self-attention, leveraging only mask and addition operations to facilitate hardware implementation. Recently, Yao et al. [38] extended their previous work [39] into a meta-architecture that achieves state-of-the-art results in SNN classification, detection, and segmentation tasks. While most spikeformers only apply spatial-wise attention in a single time step [43], some works also incorporate spatiotemporal attention [47, 33]. However, none of the previous works have utilized the Swin variant of the Spikeformer, nor for optical flow estimation.

3 Method

3.1 Event Input Representation

We divide the event stream into non-overlapping chunks. Each chunk, comprising N events within a fixed time window, is represented as $E = \{(x_i, y_i, t_i, p_i)\}_{i\in[N]}$. We preprocess each event chunk into an event discretized volume representation V using a set of bins B, following the methodology introduced in [46].

$$V(x,y,t) = \sum_i p_i \kappa(x - x_i)\kappa(y - y_i)\kappa(t - t_i) \tag{1}$$

Timestamps are normalized and scaled to the range $[0, B-1]$, $t_i = (B-1)(t_i - t_0)/(t_N - t_1)$; and $\kappa(a) = \max(0, 1 - |a|)$ is a bilinear sampling kernel. To enable the neural network to learn large temporal correlations, we encode spatiotemporal information into channels. For the ANN model, we take the previous and current chunks of event voxels, dividing the total temporal channels into n blocks. Each event input block comprises $2B/n \times H \times W$ bins. In our case, $n = 2$.

For the SNN model, to mitigate the computational burden associated with large time steps, we use only one event voxel chunk. Similarly, we partition the temporal channel, containing B bins, into n blocks along with their corresponding polarities p. This yields an event representation of size $T \times 2n \times H \times W$, with $T = B/n$ time steps. In our implementation, we set $B = 10$ and $n = 2$. This representation aligns with the spike representation outlined in [16, 17]. Each event chunk comprises $C = 4$ channels and $T = B/2$ time steps, as illustrated in Fig. 1.

3.2 Spiking Neuron

We utilize the Leaky Integrate-and-Fire (LIF) neuron model for all layers in our models. LIF is widely adopted in the literature due to its simplicity of implementation and low computational cost. For our implementation, we employ the Spikejelly library [6] and set $V_{th} = 0.1$ and $\tau_m = 2$.

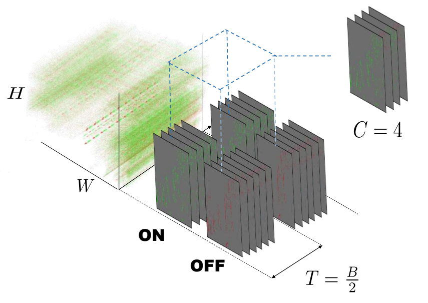


Fig. 1: Event input representation.

3.3 Network Architecture

The network architecture pipeline of our proposed methods STTFlowNet and SDformerFlow (Fig. 2) is similar. We adopt an encoder-decoder architecture, widely utilized in event-based optical flow literature [12, 46, 44, 2]. For STTFlowNet, the architecture of the Swin Transformer encoder resembles that of [22]. Each Swin block contains a Multi-Head Self-Attention (MSA) module, followed by a Feed-Forward Network (FFN) module consisting of two MLP layers. Layer Normalization (LN) is applied after each module, with residual connections incorporated. In the MSA module, unlike previous implementations [22], we utilize scaled cosine attention and logarithmic continuous relative position bias (CPB) from Swin Transformer V2 [20] to enhance the model's scaling capability. In the following sections, we focus on detailing the architecture of SDformerFlow. Further details are provided in the supplementary document.

For SDformerFlow, the primary architecture comprises three components: a) a Spike Feature Generator (SFG) embedding module, b) a Spatiotemporal Swin Spikeformer (STSF) encoder, and c) spike decoders and flow prediction. The event stream initially enters the SFG module, which outputs spatiotemporal embeddings for the STSF encoders. The STSF encoders then generate spatiotemporal features hierarchically. Subsequently, the output from each encoder is concatenated to the decoder at the same scale to predict the flow map. Two additional residual blocks exist between the encoder and decoder modules.

We propose two variants of our fully spiking model: SpikeformerFlow and SDformerFlow. The key distinction lies in the residual connection. In SpikeformerFlow, all residual shortcuts utilize spike-element-wise shortcuts (SEW) [7]. Conversely, in SDformerFlow, we employ membrane-potential shortcut (MS) [14]. Figure 3(b) illustrates the main differences among the vanilla shortcut, SEW shortcuts, and MS shortcuts. The vanilla shortcut adds spikes into the memory potential values, which cannot achieve identity mapping and have degradation problem [14]. In SEW shortcuts, the residuals are applied after the spikes, which results in integrals. Meanwhile, in MS shortcuts, residuals are applied before the spikes to preserve the spike-driven property.

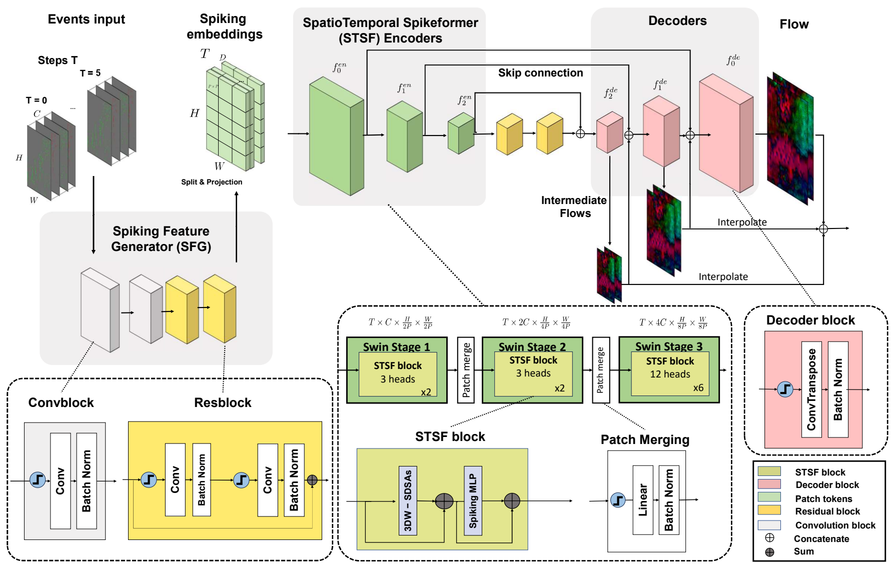


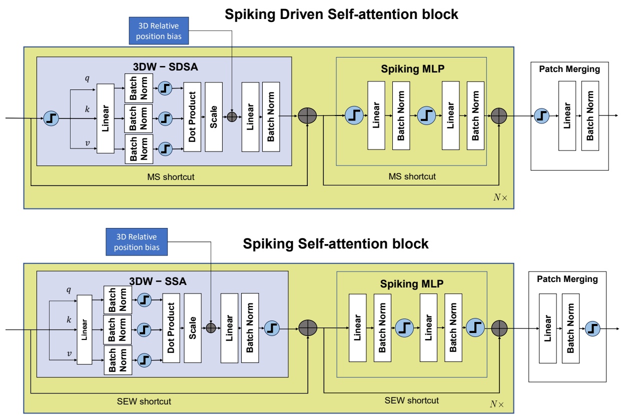
Fig. 2: SDformerFlowNet architecture.

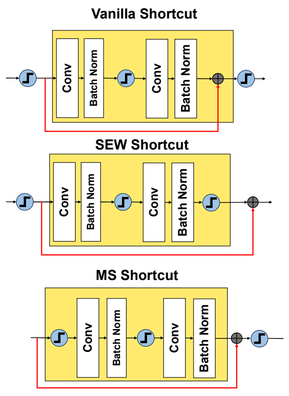
Spiking Feature Generator Embedding (SFG) The SFG module comprises two stages: generating spatiotemporal features and projecting them into token embeddings for the STSF encoder module. In the first stage, we process the event input through a spiking convolutional module followed by two residual blocks to downsample the resolution by half. This projection results in a feature map of shape $T \times C \times H/2 \times W/2$. In the second stage, we split the feature map into spatial patches of size $P \times P$, maintaining the time steps as the temporal dimension. This operation creates spatiotemporal tokens of size $1 \times P \times P$, projecting the spatial-temporal features into spike embeddings of shape $T \times C \times H/(2P) \times W/(2P)$. For STTFlowNet, both the former and latter chunks are fed into a shared *Resblock* module while retaining the spatial dimension.

Spatiotemporal Swin Spikeformer (STSF) Encoder The STSF module draws inspiration from the Video Swin Transformer [22] and Spikeformer [43]. Its detailed architecture is illustrated in Fig. 2.

We adopt three stages of Swin Transformer, with each stage comprising 2-2-6 numbers of STSF blocks successively, followed by a patch merging layer to reduce the dimension by half.

Each STSF block comprises a spiking multi-head Spike-driven Self-Attention (SDSA) block with a 3D shifted window (3DW), followed by a spiking MLP block (see Fig. 3(a) upper). Each spatiotemporal token of shape $T \times H \times W$ is partitioned into non-overlapping 3D windows of size $T_w \times H_w \times W_w$. We employ a window size of $2 \times 9 \times 9$ for cropped resolution and $2 \times 15 \times 15$ when fine-tuning





(a) Spatiotemporal Spikeformer (STSF) block architecture (b) Shortcuts in SNN

Fig. 3: In (a): The top diagram depicts the Spike-driven Self-Attention (SDSA) block utilized in SDFormerFlow, while the bottom one illustrates the Spiking Self-Attention (SSA) block with Spike-Element-Wise (SEW) shortcuts used in Spikeformerflow and in other architectures [43]. Fig.(b) shows the comparison of different residual shortcuts in SNN.

the model on a full resolution of 480×640. The SDSA is performed within the window. We utilize different numbers of attention heads $(3, 6, 12)$ for the STSF blocks in different stages. The details of our SDSA and spiking patch merge modules are explained as follows:

Spike-Driven Self-Attention (SDSA) In our SDSA block, the Query, Key, and Value tensors, denoted as Q_s, K_s, V_s, are spiking tensors. We use dot product attention, and since the attention maps are naturally non-negative, softmax is unnecessary [43]. We additionally apply a scale factor s for normalization to prevent gradient vanishing. The single-head SDSA can be formalized as follows:

$$SDSA(Q_s, K_s, V_s) = BN(Linear(SN(Q_s K_s^T V_s * s))) \tag{2}$$

Spiking Patch Merge The patch merge layer comprises a Linear layer followed by a batch normalization layer to downsample the feature map in the spatial domain.

Spike Decoder Block The decoder consists of three Transposed convolutional layers, each increasing the spatial resolution by a factor of two. A skip connection from each STSF encoder is concatenated to the prediction output from the corresponding decoder of the same scale. Flow prediction is generated at each scale and concatenated to the decoders. Loss is applied to the flow prediction upsampled to the full resolution.

3.4 Loss Function

We train our model with supervised learning using the mean absolute error between the estimated optical flow $\mathbf{u}_i^{pred} = (u_i^{pred}, v_i^{pred})$ and the ground-truth flow $\mathbf{u}_i^{gt} = (u_i^{gt}, v_i^{gt})$. Our loss function can be formulated as:

$$L = \frac{1}{n} \sum_{i=1}^{n} |\mathbf{u}_i^{pred} - \mathbf{u}_i^{gt}| \tag{3}$$

where n is the number of valid ground truth pixels. For SNN, we employ surrogate gradient (SG) [25] with backpropagation through time (BPTT) to train the network. We use the inverse tangent as the surrogate function with a width of 2.

4 Experiments

4.1 Dataset and training details

We utilize the DSEC dataset [9] for both training and evaluation purposes. The DSEC dataset is a comprehensive outdoor stereo event camera dataset featuring a resolution of 640×480. Ground-truth optical flow annotations are provided at a rate of 10Hz for some of the sequences. To address the lack of ground truth in the test set, we adopt a similar data split strategy to [2], dividing the training sequences into training and validation sets. Notably, we exclusively utilize rectified event data from the left camera. During training and validation, we perform data augmentation techniques, including random horizontal and vertical flips, as well as random crops on a 288×384 resolution. We train the models on three NVIDIA GeForce RTX 2080 Ti GPUs and employ the AdamW optimizer for a total of 80 epochs, ensuring convergence. The initial learning rate is set to 0.001 with a weight decay of 0.01. Additionally, we implement a multistep scheduler that halves the learning rate every 10 epochs. To mitigate performance degradation when scaling up to the full resolution, we conduct fine-tuning on the full-resolution data for an additional 30 epochs before testing. Given the constraints of GPU memory, training at full resolution necessitates a reduced batch size (1 or 2). During evaluation, we disable the tracking of running states for batch normalization layers to optimize memory usage.
Additionally, we trained our models on the MVSEC dataset [45]. Since the MVSEC dataset and DSEC dataset share different spatial resolutions and ground truth rates. We trained our model using MDR training dataset [24] with cropped resolution 256×256 and reported our evaluation results for the sparse optical flow to compare with other models.

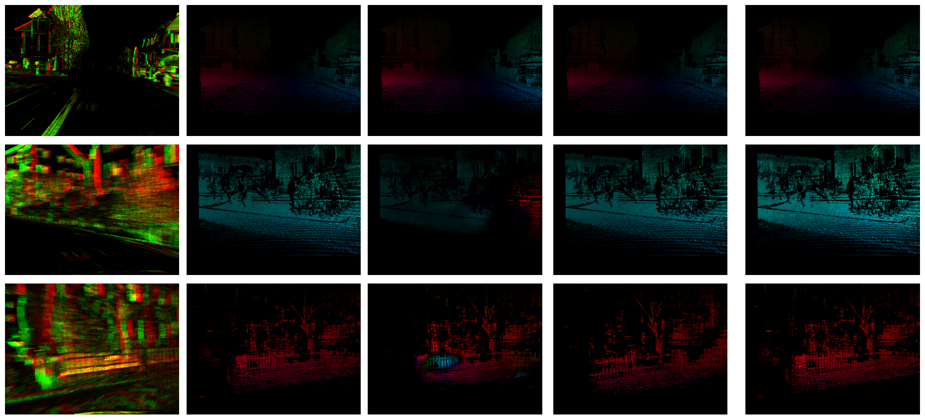


Fig. 4: Qualitative results for optical flow are evaluated on the DSEC validation dataset. The first column displays the event input, while the second column depicts the ground truth dense optical flow from our split validation dataset. During evaluation, we mask the estimated flow where ground truth flows are available. (Best viewed in color).

4.2 Results

Training	dt = 1 frame	D	outdoor_day1		indoor_flying1		indoor_flying2		indoor_flying3		Avg	
			AEE	% Outlier	AEE	% Outlier	AEE	% Outlier	AEE	% Outlier	AEE	% Outlier
	EV-FlowNet [44]	M	0.49	0.20	1.03	2.20	1.72	15.10	1.53	11.90	1.19	7.35
	EV-FlowNet2 [46]	M	**0.32**	0.00	0.58	0.00	1.02	4.00	0.87	3.00	0.69	1.75
	GRU-EV-FlowNet [12]	FPV	0.47	0.25	0.60	0.51	1.17	8.06	0.93	5.64	0.79	3.62
A	STE-FlowNet [3]	M	0.42	0	0.57	0.1	0.79	1.6	1.72	1.3	0.62	0.75
	ET-FlowNet [32]	FPV	0.39	0.12	0.57	0.53	1.2	8.48	0.95	5.73	0.78	3.72
	ADM-Flow [24]	MDR	0.41	0.00	**0.52**	0.14	**0.68**	1.18	**0.52**	0.04	**0.53**	0.34
	STT-FlowNet (ours)	MDR	0.66	0.29	0.57	0.33	0.88	4.47	0.73	1.58	0.71	1.67
	Spike-FlowNet [17]	M	0.49		0.84		1.28		1.11		0.93	
	XLIF-EV-FlowNet [12]	FPV	0.45	0.16	0.73	0.92	1.45	12.18	1.17	8.35	0.95	5.40
S	Adaptive-SpikeNet [16]	FPV	<u>0.44</u>		0.79		1.37		1.11		0.93	
	SNN3DNet [2]	M	0.85		<u>0.58</u>		<u>0.72</u>		<u>0.67</u>		<u>0.71</u>	
	SDformerFlow (Ours)	MDR	0.69	0.21	0.61	0.60	0.83	3.41	0.76	1.45	0.72	1.42

Table 2: Quantitative results for optical flow evaluated for MVSEC dataset. D shows the training dataset: MVSEC dataset, FPV or MDR dataset. We highlight the best-performing results and underline the best results for the SNN model in each tested sequence.

The quantitative results for our models, STTFlowNet and SDformerFlow, evaluated on the DSEC benchmark, are presented in Table 1. Additionally, Figures 4 and 5 illustrate the qualitative results obtained from the validation and test dataset, respectively.

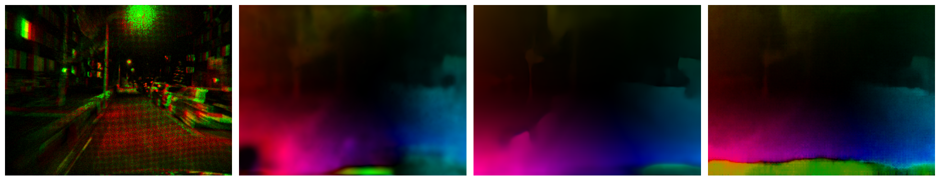


Fig. 5: Qualitative results for optical flow are evaluated on the official DSEC test dataset. The first column presents the event input, while the other columns show the corresponding estimated optical flow for the baseline method EVFlow and our methods STTFlowNet and SDformerFlow. (Best viewed in color).

Training		EPE	Outlier %	AAE
A	E-RAFT [10]	0.779	2.684	2.838
	EV-FlowNet_retrained [10]	2.32	18.60	-
	IDNet [35]	0.719	2.036	2.723
	TMA [19]	0.743	2.301	2.684
	E-Flowformer [18]	0.759	2.446	2.676
	TamingCM[26]	2.33	17.771	10.56
	STTFlowNet-en3 (Ours)	0.997	4.588	3.235
S	SNN_3DNet[2]	1.707	10.308	6.338
	SDFormerFlow-en3 (Ours)	2.142	14.021	5.941
M	MultiCM [28]	3.472	30.855	13.983

Table 1: Quantitative results for optical flow estimation of the DSEC optical flow benchmarks for all the test sequences. The first column shows the methods, A stands for ANN, S stands for SNN, while M stands for model-based method.

Figure 4 shows both our STTFlowNet and SDformerFlow models, trained with cropped resolution on our split training dataset and tested on the validation dataset. We use bicubic interpolation to remap the relative positional bias for full resolution, as described in [21]. Notably, when the vehicle moves forward in steady motion, all models achieve accurate flow estimation. However, in scenarios involving sharp turns or large, abrupt motions (third row in the figure), the baseline EVFlowNet struggles to estimate the correct direction. In contrast, both our STTFlowNet and our fully spiking model effectively handle such scenarios, thanks to their utilization of spatiotemporal attention mechanisms.

Figure 5 showcases the improved estimation performance of our models on the DSEC optical flow benchmark[1] test set compared to the baseline. Notably, SDformerFlow achieves superior estimation, although it encounters challenges

[1] Full benchmark statistics are available at https://dsec.ifi.uzh.ch/uzh/dsec-flow-optical-flow-benchmark/

in areas where the sensor hits the car hood for which ground truth data is unavailable. This limitation could be attributed to the additional convolutional layers added in the early stages to downsample and reduce memory consumption.

Regarding the quantitative evaluation presented in Table 1, our ANN model outperforms the baseline model [46] and other self-supervised trained models [26]. However, it still trails behind correlation-volume-based models [10, 18]. The only SNN model included in the benchmark [2] uses stateless neurons and is trained at full resolution, whereas most other SNN approaches are trained and validated on cropped resolution [27, 16], with limited representation in the benchmark. Notably, our fully spiking model, SDformerFlow, exhibits superior performance compared to the ANN baseline [46].

The quantitative evaluation tested on MVSEC dataset is presented in Table. 2. Both our ANN and SNN models yield competitive results. Our ANN model performs better than another transformer-based U-Net architecture [32]. Our SDformerFlow ranked second for the average AEE for all the sequences among all the SNN methods. However, the best performing model [2] reports their results for the indoor sequences separately trained on the subsets of the same dataset, which may have overfitted to the test dataset.

4.3 Ablation studies

Model	EPE	Outlier %	AEE	I	Training res.	Param. (M)
EVFlowNet_retrained	1.63	10.01	5.84	count	288,384	14.14
EVFlowNet_retrained	1.57	9.918	6.09	voxel	288,384	14.14
STTFlowNet-en3-b2-p4-w5	1.67	12.61	8.22	count	240,320	20.30
STTFlowNet-en3-b2-p2-w10	1.34	8.29	5.98	count	240,320	20.30
STTFlowNet-en3-b4-p4-w10	1.37	8.21	6.77	count	240,320	20.29
STTFlowNet-en3-b4-p2-w10	1.43	9.44	5.54	count	240,320	20.29
STTFlowNet-en3-b4-p2-w10	1.05	4.97	5.34	voxel	240,320	20.29
STTFlowNet-en3-b2-p2-w10	0.94	3.97	4.78	voxel	240,320	20.30
STTFlowNet-en3-b2-p4-w10	0.83	2.61	4.36	voxel	480,640	20.29
STTFlowNet-en4-b2-p4-w10	**0.81**	**2.50**	**4.33**	voxel	480,640	57.51

Table 3: Ablation study for STTFlowNet. Column I stands for the event input type. For the variant of STTFlowNet, en means number of encoders, b stands for number of input blocks, p means spatial patch size, and w stands for swin spatial window size. Best-performing results are highlighted.

The ablation study was conducted on the validation DSEC set. For the ANN models, we retrained EVFlowNet [46] on the DSEC training set as our base model for 60 epochs while randomly cropping to size 288×384. Our ANN model shares the same U-Net architecture with EVFlowNet, with the key difference being the use of spatiotemporal swin transformer encoders instead of convolutional layers. Our models were trained at either half or full resolution of 480×640, and validated in full resolution. We analyzed the effects of: a) the input representation:

Model	EPE		Outlier %		AEE		I	Training res.	Param. (M)
test res: cropped (C) or full (F)	C	F	C	F	C	F			
LIF-EV-FlowNet-en4-s5	3.08	3.47	19.67	23.70	17.90	14.41	voxel10	288,384	14.13
SpikeformerFlowNet-SEW-en3-s8-c4	1.60	3.21	11.90	32.30	12.51	14.77	voxel15*	240,320	19.80
SpikeformerFlowNet-SEW-en3-s4-c8	1.76	3.54	13.43	41.18	14.01	27.81	voxel15*	240,320	19.81
SpikeformerFlowNet-SEW-en3-s5-c4	1.51	2.52	9.85	22.75	10.68	11.10	voxel10	288,384	19.83
SpikeformerFlowNet-MS-en3-s5-c4	1.28	2.01	6.91	15.55	9.01	8.99	voxel10	288,384	19.83
SpikeformerFlowNet-MS-en4-s5-c4	**1.25**	**1.98**	**6.69**	**15.06**	**8.48**	**8.81**	voxel10	288,384	56.48
SpikeCAformerFlow-MS-en4-s5-c4	1.66	2.97	10.65	27.87	12.05	22.55	voxel10	288,384	15.73

*The SEW variant with input voxel size of 15 was trained with a resolution of 240×320 due to GPU memory limitations. The rest of the Spikeformer models were trained at 288×384 resolution.

Table 4: Ablation study for SDformerFlow. For the SNN model variants, en denotes number of encoders, s stands for number of steps, and c stands for number of channels. Best performing results are highlighted in bold.

event voxel or count; b) the number of temporal partitioning blocks: 2 (b2) or 4 (b4); c) the spatial patch size: $p = 2$ or $p = 4$, and swin spatial window size w; and d) the training resolution.

Results are summarized in Table. 3. Using the event voxel representation retained more temporal information and notably improved results. Introducing swin transformer layers instead of convolutions also led to significant performance gains. For the variants of STTFlowNet, the window size influenced the range of the area to pay attention to, with smaller window sizes making it difficult for the network to learn larger displacements. Adjusting the patch size between 2 and 4 according to the window size and resolution was found to be effective. Partitioning the temporal domain into 2 chunks yielded better results than 4, potentially due to the total number of channels. Further improvements may be achieved by incorporating a local-global chunking approach as described in [42]. To maintain equivalence between our ANN and SNN models, we utilized local temporal blocks exclusively. Given the performance degradation experienced by the swin transformer at higher resolutions, we opted to train the model directly at full resolution using a patch size of 4. This approach ensured that the resolution within the swin encoders remained consistent with training the model at half resolution with a patch size of 2. Notably, this strategy resulted in remarkable improvements in performance.

For our SNN model, we trained the fully spiking version of EVFlowNet with LIF neurons using the same input representations as our base model for comparison. We studied: a) the number of time steps/channels; b) shortcut variants: SEW or MS shortcuts; and c) the number of encoders.

Results are presented in Table. 4. The spikeformer encoders significantly improved performance compared to the baseline model, albeit with reduced robustness when directly tested on scaled-up resolutions. Increasing the number of time steps helped capture temporal information at the expense of increased memory consumption. Opting for 5 time steps and 4 channels struck a balance between performance and memory consumption. The MS shortcut variant notably improved results compared to SEW shortcut. One possible explanation is that the MS shortcut provides an information flow path between the states of the neurons before the spike function and is not regulated by their firing status. Increasing the number of encoders from three to four further enhanced performance at the cost of increased parameters. Finally, incorporating convolution-based modules as CAformer in [38, 41] in the first two swin encoders yielded a lightweight model but with slightly reduced performance.

4.4 Energy consumption analysis

In our energy consumption analysis, we follow established methodologies from prior research [38, 27, 16]. For ANN models, we estimate energy consumption based on the number of floating-point operations (FLOPS) required, as all operations in ANN layers are multiply-accumulate (MAC) operations. Therefore, the energy consumption for ANN models is calculated as $FLOPS \times E_{MAC}$. Conversely, SNN models convert multiplication operations into addition operations due to their binary nature. Thus, for SNN models, we estimate energy consumption by multiplying the FLOPS with the spiking rate R_s and the number of time steps T, resulting in $FLOPS \times R_s \times T \times E_{AC}$. Here, E_{MAC} represents the energy required for MAC operations, and E_{AC} represents the energy required for addition operations. For 32-bit floating-point computation, these energy values are typically $E_{MAC} = 4.6pJ$ and $E_{AC} = 0.9pJ$, respectively, based on a 45 nm technology [13]. We estimate the average spiking rates among all time steps for each layer to calculate energy consumption, ignoring the negligible contribution of batch normalization layers (around 0.01%). The energy consumption for each model during the inference phase, with an image input size of 288×384, is presented in Table 5. Our results demonstrate that the energy consumption of our SNN model is nearly one-tenth that of its ANN counterpart and one-third that of the baseline EVFlowNet model.

Model	EPE	Type	Param (M)	FLOPS(G)	Avg. spiking rate	Power(mJ)
EVFlowNet retrained	1.57	ANN	14.14	22.38	-	102.95
LIF-EVFlowNet	3.08	SNN	14.13	22.38	0.29	29.21
STTFlowNet-en3	0.72	ANN	20.30	86.88	-	399.65
SDFlowNet-en3	1.28	SNN	19.83	34.80	0.27	37.64
SDFlowNet-en4	1.25	SNN	56.48	39.10	0.27	41.08

Table 5: Energy consumption for ANN and SNN models

5 Conclusions and future work

We introduced STTFlowNet and SDformerFlow, two novel architectures for event-based optical flow estimation that leverage spatiotemporal swin transformer encoders in ANN and SNN frameworks, respectively. Our work marks the first application of a spikeformer for event-based optical flow estimation. Despite not using correlation volumes and facing scalability challenges inherent to transformer architectures, our results highlight the potential in the use of spikeformers in regression tasks. Our ablation studies shed light on the importance of key components such as input representation, temporal partitioning, and spatial window size, providing insights for future research directions. Our SNN version is the first fully spikeformer implementation and is comparable to the other SNN implementations reported in the benchmark. Notably, our SNN model achieved remarkable energy savings compared to its ANN counterpart and also outperformed the baseline EVFlowNet model. We believe by introducing spatiotemporal attention, we strengthen our model's capability to map global context for the spatial feature maps while capturing spatiotemporal correlations, which improves the performance of our model compared to other CNN-based methods.

However, one limitation of our work is that, by feeding the entire chunk of data into the spatiotemporal attention modules, we were not able to fully exploit the asynchronous ability of the event camera and SNNs. This can be improved by introducing temporal delay, as proposed in [33], in future work. Secondly, transformer-based models suffer from constrained scalability across different resolutions. Recent work proposes methods to address this issue by incorporating multi-resolution training [31] or dynamic resolution adjustment modules [5]. Thirdly, our model is based on the encoder-decoder architecture to prevent calculating the correlation features and iterative process since it obeys the motivation to use SNN as a computational and energy-efficient solution, thus the performance still falls behind some state-of-the-art methods. For improvement, we propose to train with more diverse datasets and exploit learned neural dynamics parameters. Finally, much work remains to be done related to hardware implementation to fully exploit the advantage of energy efficiency of SNNs.

In conclusion, our work highlights the efficacy of integrating transformer architectures with spiking neural networks for efficient and robust optical flow estimation, paving the way for advancements in neuromorphic vision systems.

Acknowledgments

This work received support from projects EBCON (PID2020-119244GB-I00) and RAADICAL (PLEC2021-007817) funded by MCIN/ AEI/ 10.13039/ 501100011033 and the "European Union NextGenerationEU/PRTR"; the Consolidated Research Group RAIG (2021 SGR 00510) of the Departament de Recerca i Universitats de la Generalitat de Catalunya; and by an FI AGAUR PhD grant to Yi Tian.

Bibliography

[1] de Blegiers, T., Dave, I.R., et al.: Eventtransact: A video transformer-based framework for event-camera based action recognition. In: IEEE/RSJ Int. Conf. Intell. Robots Syst. pp. 1261–1267 (2023)

[2] Cuadrado, J., Rançon, U., et al.: Optical flow estimation from event-based cameras and spiking neural networks. Front. Neurosci. **17**, 1160034 (2023)

[3] Ding, Z., Zhao, R., et al.: Spatio-temporal recurrent networks for event-based optical flow estimation. In: AAAI Conf. Artif. Intell. vol. 36, pp. 525–533 (2021)

[4] Dosovitskiy, A., Beyer, L., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: Int. Conf. Learn. Represent. (2020)

[5] Fan, Q., You, Q., et al.: Vitar: Vision transformer with any resolution. arXiv preprint arXiv:2403.18361 (2024)

[6] Fang, W., Chen, Y., et al.: Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. Sci. Adv. **9**(40), eadi1480 (2023)

[7] Fang, W., Yu, Z., et al.: Deep residual learning in spiking neural networks. In: Conf. Neural Inf. Process. Syst. vol. 34, pp. 21056–21069 (2021)

[8] Gao, Y., Lu, J., et al.: Action recognition and benchmark using event cameras. IEEE Trans. Pattern Anal. Mach. Intell. **45**(12), 14081–14097 (2023)

[9] Gehrig, M., Aarents, W., et al.: DSEC: A stereo event camera dataset for driving scenarios. IEEE Robotics Autom. Lett. **6**(3), 4947–4954 (2021)

[10] Gehrig, M., Millhausler, M., et al.: E-RAFT: Dense optical flow from event cameras. In: Int. Conf. 3D Vis. pp. 197–206 (2021)

[11] Guizilini, V., Ambrus, R., et al.: Multi-frame self-supervised depth with transformers. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 160–170 (2022)

[12] Hagenaars, J., Paredes-Vallés, F., de Croon, G.: Self-supervised learning of event-based optical flow with spiking neural networks. In: Conf. Neural Inf. Process. Syst. vol. 34 (2021)

[13] Horowitz, M.: 1.1 computing's energy problem (and what we can do about it). In: IEEE Int. Solid-State Circuits Conf. pp. 10–14 (2014)

[14] Hu, Y., Deng, L., et al.: Advancing spiking neural networks toward deep residual learning. IEEE Trans. Neural Networks Learn. Syst. (2024), early access

[15] Huang, Z., Shi, X., et al.: Flowformer: A transformer architecture for optical flow. In: Eur. Conf. Comput. Vis. pp. 668–685 (2022)

[16] Kosta, A.K., Roy, K.: Adaptive-spikenet: Event-based optical flow estimation using spiking neural networks with learnable neuronal dynamics. In: IEEE Int. Conf. Robotics Autom. pp. 6021–6027 (2023)

[17] Lee, C., Kosta, A., et al.: Spike-flownet: Event-based optical flow estimation with energy-efficient hybrid neural networks. In: Eur. Conf. Comput. Vis. pp. 366–382 (2020)

[18] Li, Y., Huang, Z., et al.: Blinkflow: A dataset to push the limits of event-based optical flow estimation. In: IEEE/RSJ Int. Conf. Intell. Robots Syst. pp. 3881–3888 (2023)

[19] Liu, H., Chen, G., et al.: TMA: Temporal motion aggregation for event-based optical flow. In: IEEE Int. Conf. Comput. Vis. pp. 9651–9660 (2023)

[20] Liu, Z., Hu, H., et al.: Swin transformer v2: Scaling up capacity and resolution. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 12009–12019 (2022)

[21] Liu, Z., Lin, Y., et al.: Swin transformer: Hierarchical vision transformer using shifted windows. In: IEEE Int. Conf. Comput. Vis. pp. 10012–10022 (2021)
[22] Liu, Z., Ning, J., et al.: Video swin transformer. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 3202–3211 (2022)
[23] Lu, Y., Wang, Q., et al.: Transflow: Transformer as flow learner. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 18063–18073 (2023)
[24] Luo, X., Luo, K., et al.: Learning optical flow from event camera with rendered dataset. arXiv preprint arXiv:2303.11011 (2023)
[25] Neftci, E.O., Mostafa, H., Zenke, F.: Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. IEEE Signal Process. Mag. **36**(6), 51–63 (2019)
[26] Paredes-Vallés, F., Scheper, K.Y.W., et al.: Taming contrast maximization for learning sequential, low-latency, event-based optical flow. In: IEEE Int. Conf. Comput. Vis. pp. 9695–9705 (2023)
[27] Ponghiran, W., Liyanagedera, C.M., Roy, K.: Event-based temporally dense optical flow estimation with sequential learning. In: IEEE Int. Conf. Comput. Vis. pp. 9827–9836 (2023)
[28] Shiba, S., Aoki, Y., Gallego, G.: Secrets of event-based optical flow. In: Eur. Conf. Comput. Vis. pp. 628–645 (2022)
[29] Sui, X., Li, S., et al.: CRAFT: Cross-attentional flow transformer for robust optical flow. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 17581–1790 (2022)
[30] Teed, Z., Deng, J.: RAFT: Recurrent all-pairs field transforms for optical flow. In: Eur. Conf. Comput. Vis. pp. 402–419 (2020)
[31] Tian, R., Wu, Z., et al.: Resformer: Scaling vits with multi-resolution training. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 22721–22731 (2023)
[32] Tian, Y., Andrade-Cetto, J.: Event transformer flownet for optical flow estimation. In: British Mach. Vis. Conf. (2022)
[33] Wang, Y., Shi, K., et al.: Spatial-temporal self-attention for asynchronous spiking neural networks. In: Int. Joint Conf. Artif. Intell. pp. 3085–3093 (2023)
[34] Wang, Z., Fang, Y., et al.: Masked spiking transformer. In: IEEE Int. Conf. Comput. Vis. pp. 1761–1771 (2023)
[35] Wu, Y., Paredes-Vallés, F., de Croon, G.C.H.E.: Rethinking event-based optical flow: Iterative deblurring as an alternative to correlation volumes. arXiv preprint arXiv:2211.13726 (2023)
[36] Xu, H., Zhang, J., et al.: Gmflow: Learning optical flow via global matching. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 8121–8130 (2022)
[37] Yang, Y., Pan, L., Liu, L.: Event camera data pre-training. arXiv preprint arXiv:2301.01928 (2023)
[38] Yao, M., Hu, J., et al.: Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. In: Int. Conf. Learn. Represent. (2024)
[39] Yao, M., Hu, J., et al.: Spike-driven transformer. In: Conf. Neural Inf. Process. Syst. (2023)
[40] Yao, M., Zhao, G., et al.: Attention spiking neural networks. IEEE Trans. Pattern Anal. Mach. Intell. **45**(8), 9393–9410 (2023)
[41] Yu, W., Si, C., et al.: Metaformer baselines for vision. IEEE Trans. Pattern Anal. Mach. Intell. **46**(2), 896–912 (2024)
[42] Zhang, J., Tang, L., et al.: Spike transformer: Monocular depth estimation for spiking camera. In: Eur. Conf. Comput. Vis. pp. 34–52 (2022)
[43] Zhou, Z., Zhu, Y., et al.: Spikformer: When spiking neural network meets transformer. In: Int. Conf. Learn. Represent. (2023)

[44] Zhu, A., Yuan, L., et al.: EV-FlowNet: Self-supervised optical flow estimation for event-based cameras. In: Robotics Sci. Syst. Conf. (2018)
[45] Zhu, A.Z., Thakur, D., et al.: The multivehicle stereo event camera dataset: An event camera dataset for 3D perception. IEEE Robotics and Automation Letters **3**(3), 2032–2039 (2018)
[46] Zhu, A.Z., Yuan, L., et al.: Unsupervised event-based learning of optical flow, depth, and egomotion. In: IEEE/CVF Conf. Comput. Vis. Pattern Recognit. pp. 989–997 (2019)
[47] Zou, S., Mu, Y., et al.: Event-based human pose tracking by spiking spatiotemporal transformer. arXiv preprint arXiv:2303.09681 (2023)