Supplementary materials: Discrete time convolution for fast event-based stereo

Kaixuan Zhang[1,3*] Kaiwei Che[2,3] Jianguo Zhang[1,4]
Jie Cheng[3] Ziyang Zhang[3] Qinghai Guo[3] Luziwei Leng[3*†]
[1] Department of Computer Science and Engineering, Southern University of Science and Technology, China
[2] Department of Electrical and Electronic Engineering, Southern University of Science and Technology, China
[3] ACS Lab, Huawei Technologies, Shenzhen, China [4] Peng Cheng Lab, Shenzhen, China

1. Numerical solution of continuous time convolution

To avoid numerical instabilities, a semi-implicit Euler method was applied to solve the ordinary differential equation (ODE) of fully connected LTC in [3]. We use a similar approach. The ODE of convolution LTC is described as:

$$\begin{aligned}\frac{dx_{cij}(t)}{dt} =& -\left[\frac{1}{\tau_{\mathrm{m},c}}+\frac{I_{cij}(t)}{C_{\mathrm{m},c}}\right]x_{cij}(t)\\ &+\frac{I_{cij}(t)}{C_{\mathrm{m},c}}E_{\mathrm{rev},c}+\frac{E_{\mathrm{leak},c}}{\tau_{\mathrm{m},c}}\end{aligned} \tag{1}$$

$$I_{cij}(t)=\sum_h\sum_k w_{chk}P^t_{h+i,k+j} \tag{2}$$

where $I_{cij}(t)$ represents the convolution input on channel c at location i, j from the event frame pre-processed by SBT. The basic Euler method [5] is formulated as:

$$x(t+\Delta)=x(t)+\Delta f(x(t+\tau),u(t+1)) \tag{3}$$

where Δ denotes a fixed step size, u denotes inputs to the neuron. By setting $\tau=\Delta$, it realizes the implicit Euler method, which we apply to the case of convolution LTC and get:

$$\begin{aligned}x_{cij}(t+\Delta) =& x_{cij}(t)+\Delta\left\{\frac{I_{cij}(t+1)}{C_{\mathrm{m},c}}E_{\mathrm{rev},c}+\frac{E_{\mathrm{leak},c}}{\tau_{\mathrm{m},c}}\right.\\ &\left.-\left[\frac{1}{\tau_{\mathrm{m},c}}+\frac{I_{cij}(t+1)}{C_{\mathrm{m},c}}\right]x_{cij}(t+\Delta)\right\}\end{aligned} \tag{4}$$

After readjusting the equation, we get:

$$x_{cij}(t+\Delta)=\frac{x_{cij}(t)\frac{C_{\mathrm{m},c}}{\Delta}+E_{\mathrm{rev},c}I_{cij}(t+1)+g_{\mathrm{l},c}E_{\mathrm{leak},c}}{\frac{C_{\mathrm{m},c}}{\Delta}+g_{\mathrm{l},c}+I_{cij}(t+1)} \tag{5}$$

where $g_{\mathrm{l},c}=C_{\mathrm{m},c}/\tau_{\mathrm{m},c}$. Following [3], to ensure precision, we evolve the LTC neuron with a frequency six times higher ($\Delta=1/6$) than the input rate.

Readjusting the convolution LTC equation 1, we get:

$$\begin{aligned}\frac{dx_{cij}(t)}{dt} =& \frac{E_{\mathrm{leak},c}-x_{cij}(t)}{\tau_{\mathrm{m},c}}\\ &+\frac{I_{cij}(t)(E_{rev,c}-x_{cij}(t))}{C_{\mathrm{m},c}}\end{aligned} \tag{6}$$

When neglecting the influence of $E_{\mathrm{rev},c}$ on the membrane potential, $E_{rev,c}-x_{cij}(t)$ can be viewed as 1. Then we obtain convolution LTC without reversal potential:

$$\frac{dx_{cij}(t)}{dt}=\frac{E_{\mathrm{leak},c}-x_{cij}(t)}{\tau_{\mathrm{m},c}}+\frac{I_{cij}(t)}{C_{\mathrm{m},c}} \tag{7}$$

Applying a similar method as before, its numerically solution is formulated as:

$$x_{cij}(t+\Delta)=\frac{x_{cij}(t)\frac{C_{\mathrm{m},c}}{\Delta}+I_{cij}(t+1)+g_{\mathrm{l},c}E_{\mathrm{leak},c}}{C_{\mathrm{m},c}+g_{\mathrm{l},c}\Delta} \tag{8}$$

2. Network Architecture

We develop our proposed framework using [6] as a baseline, with major modifications in the feature embedding sub-network. The architecture of DTC-SPADE network is shown in Table 1. The network receives as an input left and right SBT stacks of size $15\times5\times h\times w$ and returns disparity tensor of size $h\times w$. For the DTC module, $x_{cij}(0)$ is initialized by zero and $I_{cij}(t)$ obtained by 2D convolution followed by batch normalization. For the SPADE module, we perform multi-scale dilated convolution and stack their outputs, based on which we further extract modulation parameters.

*These authors contribute equally to this work.
†Corresponding author. lengluziwei@huawei.com

#	**Layer Description**	**output size**
	load data & pre-process	
origin	SBT	$15 \times 5 \times h \times w$
	DTC module	
T	2D conv. $5 \times 3 \times 3 \times 32$ stride 2 with BN, summed with $\tau x(t)$	$32 \times h \times w$
	Spatial aggregation	
S1	2D conv. $32 \times 5 \times 5 \times 32$ stride 2	$32 \times \frac{1}{2}h \times \frac{1}{2}w$
S2	2D conv. $32 \times 5 \times 5 \times 32$ stride 2	$32 \times \frac{1}{4}h \times \frac{1}{4}w$
S3	$2\times$ residual block with $32 \times 3 \times 3 \times 32$ 2D conv.	$32 \times \frac{1}{4}h \times \frac{1}{4}w$
S4-redir	2D conv. $32 \times 3 \times 3 \times 8$ no IN, LeakyReLU	$8 \times \frac{1}{4}h \times \frac{1}{4}w$
	SPADE module	
SP_pre0	Take the last stack from origin	$5 \times h \times w$
SP_pre1	2D conv. $5 \times 1 \times 1 \times 16$ with avg_pooling	$16 \times \frac{1}{2}h \times \frac{1}{2}w$
SP_pre2	2D conv. $16 \times 1 \times 1 \times 32$ with avg_pooling	$32 \times \frac{1}{4}h \times \frac{1}{4}w$
SP1	2D conv. $32 \times 3 \times 3 \times 32$ stride 1 with ReLU	$64 \times \frac{1}{4}h \times \frac{1}{4}w$
SP2	$4 \times$ 2D conv. $64 \times 3 \times 3 \times 16$ with dilation $\{1, 3, 4, 5\}$ and concatenation	$64 \times \frac{1}{4}h \times \frac{1}{4}w$
SP3	$2 \times$ 2D conv. $64 \times 5 \times 5 \times 32$, for γ and β	$64 \times \frac{1}{4}h \times \frac{1}{4}w$
SP4	Apply BN to S3 and adjust it with γ and β: $\mathrm{S3}_{\mathrm{BN}}(1+\gamma)+\beta$	$32 \times \frac{1}{4}h \times \frac{1}{4}w$
	Matching module	
M1	concatenate left-right embeddings SP4	$64 \times \frac{1}{4}h \times \frac{1}{4}w$
M2	2D conv. $64 \times 3 \times 3 \times 64$ stride 2	$64 \times \frac{1}{4}h \times \frac{1}{4}w$
M3	$2\times$ residual block with $64 \times 3 \times 3 \times 64$ 2D conv.	$64 \times \frac{1}{4}h \times \frac{1}{4}w$
M4	2D conv. $64 \times 3 \times 3 \times 8$ no IN,LeakyReLU	$8 \times \frac{1}{4}h \times \frac{1}{4}w$
	Regularization module	
R1	concatenate joint embeddings M4	$8 \times \frac{1}{4}d_{max} \times \frac{1}{4}h \times \frac{1}{4}w$
R2	3D conv.$8 \times 3 \times 3 \times 3 \times 8$	$8 \times \frac{1}{4}d_{max} \times \frac{1}{4}h \times \frac{1}{4}w$
R3	3D conv.$8 \times 3 \times 3 \times 3 \times 8$, stride 2	$16 \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
R4	R3+S4-redir	$16 \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
R5	3D conv.$16 \times 3 \times 3 \times 3 \times 16$	$16 \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
R6	R5+R4	$16 \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
R7	3D conv.$16 \times 3 \times 3 \times 3 \times 32$, stride 2	$32 \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
R8	3D conv.$32 \times 3 \times 3 \times 3 \times 32$	$32 \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
R9	R8+R7	$32 \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
R10	3D conv.$32 \times 3 \times 3 \times 3 \times 64$, stride 2	$64 \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
R11	3D conv.$64 \times 3 \times 3 \times 3 \times 64$	$64 \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
R12	R11+R10	$64 \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
R13	3D conv.$64 \times 3 \times 3 \times 3 \times 128$, stride 2	$128 \times \frac{1}{64}d_{max} \times \frac{1}{64}h \times \frac{1}{64}w$
R14	3D transposed conv.$128 \times 4 \times 4 \times 4 \times 64$, stride 2	$64 \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
R15	R14+R11	$64 \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
R16	3D conv.$64 \times 3 \times 3 \times 3 \times 64$	$64 \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
R17	3D transposed conv.$64 \times 4 \times 4 \times 4 \times 32$, stride 2	$32 \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
R18	R17+R8	$32 \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
R19	3D conv.$32 \times 3 \times 3 \times 3 \times 32$	$32 \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
R20	3D transposed conv.$32 \times 4 \times 4 \times 4 \times 16$, stride 2	$16 \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
R21	R20+R5	$16 \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
R22	3D conv.$16 \times 3 \times 3 \times 3 \times 16$	$16 \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
R23	3D transposed conv.$16 \times 4 \times 4 \times 4 \times 8$, stride 2	$8 \times \frac{1}{4}d_{max} \times \frac{1}{4}h \times \frac{1}{4}w$
R24	R23+R3	$8 \times \frac{1}{4}d_{max} \times \frac{1}{4}h \times \frac{1}{4}w$
R25	3D conv.$8 \times 3 \times 3 \times 3 \times 8$	$8 \times \frac{1}{4}d_{max} \times \frac{1}{4}h \times \frac{1}{4}w$
R26	3D transposed conv.$8 \times 4 \times 4 \times 4 \times 4$, stride 2	$4 \times \frac{1}{2}d_{max} \times \frac{1}{2}h \times \frac{1}{2}w$
R27	3D transposed conv.$4 \times 3 \times 4 \times 4 \times 1$, stride (1,2,2) no IN,LeakyReLU	$\frac{1}{2}d_{max} \times h \times w$
	Estimator	
	see equation in paper	$h \times w$

Table 1. Detailed architecture of DTC-SPADE. The residual blocks consist of two 2d convolutions followed by shortcut connections. The convolutions and transposed convolutions, including these in the residual blocks, are followed by LeakyReLU with negative slope 0.2 and Instance Normalization (IN) [7], unless explicitly stated otherwise. BN denotes Batch Normalization. The network receives as an input left and right SBT stacks of size $15 \times 5 \times h \times w$ and returns disparity tensor of size $h \times w$

Δt [ms]	n	**MDE, [cm]** ↓	**1PA, [%]** ↑
50	1	15.3±0.2	91.3±0.2
50	5	15.4±0.1	91.2±0.1
50	10	15.7±0.8	90.2±1.7

Table 2. Network (DTC-PDS) performance with different SBT parameters.

3. Experiments on the MVSEC dataset

Training setup

Both DTC-PDS and DTC-SPADE are trained for 44 epochs with batch size 1 on a single NVIDIA Tesla V100 (32G) GPU. We use RMSprop optimizer. DTC module uses learning rate of 0.005 while other modules use learning rate of 0.001, both halve in epoch 15 and epoch 23. We apply positive constraint on τ of the DTC module during training for a stable accumulation of past history.

Experiments with different SBT parameters

For various dynamic scenes, n can be adjusted and the input can be updated on single channel ($[f_{t_1}, f_{t_2}...f_{t_n}] \rightarrow [f_{t_2}, f_{t_3}...f_{t_{n+1}}]$), enabling $\frac{\Delta t}{n}$ ms input resolution of the network. The ground truth disparity of the MVSEC dataset is 20 Hz, leading to our choice of a spanning time of $\Delta t = 50$ ms for one SBT stack. In a stack, events form $n = 5$ frames so the minimum temporal resolution of the input is 10 ms. Ideally, increasing n maintains more temporal information, but it also increases spatial sparsity of the data and could make the training harder. We found the network has slightly decreasing average precision when increasing n, as shown in table 2.

CTC feature maps

The feature map of CTC is shown in Fig. 1. The feature map aggregates its past states and gradually forms a denser spatial representation. Compared to DTC, the feature map of CTC have more uniformed color on background scenes, due to a direct sum from the previous membrane potential without normalization of the sigmoid activation function in DTC. The bottom row shows four feature maps with different time constants at the end of evolution. Channels with larger τ remember more history information than those with smaller τ.

Random seed experiments for SPADE

DTC-PDS is the ablation model for DTC-SPADE since all other parts of both networks are the same except for SPADE. We trained DTC-SPADE with 3 different random seeds and the result shows that SPADE can statistically improve the network performance, with 1PA: 92.9±0.1, MDE: 13.5±0.1 on split 1 of the MVSEC dataset.

Method	EO	1PE ↓	2PE ↓	MAE ↓	RMSE ↓
EIS-EI [4]	✗	5.814	1.055	0.396	0.905
EIS-ES [4]	✓	9.958	2.645	0.529	1.222
DDES [6]	✓	10.915	2.905	0.576	1.386
DTC-PDS	✓	10.462	2.627	0.562	1.34
DTC-SPADE	✓	10.256	2.716	0.558	1.35
DTC-PDS (×2)	✓	9.517	2.356	0.527	1.264
DTC-SPADE (×2)	✓	9.27	2.405	0.526	1.285

Table 3. Results on the DSEC dataset. EO denotes event-only input for both training and inference. ×2 denotes twice the channel number of DDES. For more comparisons and detailed evaluations on individual sequences please refer to the DSEC disparity benchmark website [1].

Plot of Fig.4

We use 'jet' colormap in python matplotlib with a scale [0, 37], which corresponds to the disparity range of the MVSEC dataset. The absence of qualitative comparison with EITnet is because their code is not publicly available. Besides, no colormap information was found in their paper, we tried but were still unable to plot our result in the same colormap as theirs.

Streaming experiments

In the streaming experiments, the entire test split (including the test and the validation set, following the setup of DDES [6]) of '1' (frame indices 140-1200) and '3' (frame indices 73-1615) are sequentially fed into the model, which evolves an equal length of steps and estimates the corresponding disparities. We then explicitly picked frame indices belonging to the test set (obtained from the default random seed from the released code of DDES) in standard training setup to make an accurate statistic.

For the FPS calculation, the time consumption of SBT is not considered. Event cameras such as *Davis* and *Prophesee* are embedded with accumulator modules that directly output event stacks in SBT or SBN, and in real application this duration is minimum (< 0.1 ms tested with *Davis346*). The recalculated FPS including SBT preparation (additional 2 ms on CPU) is 90 FPS for DTC-PDS and 57 FPS for DTC-SPADE.

4. Experiments on the DSEC dataset

For both DTC-PDS and DTC-SPADE network, we use voxel grid method as in [2] to compress events within 100 ms into 10 frames as a frame stack and take two stacks as the input. We randomly crop the input from 480×640 into 448×576 with probability of 50%. Because DSEC dataset is much bigger than MVSEC dataset, we set the spatial embedding channel to 128 (4 times larger than what we applied to the MVSEC dataset and 2 times larger compared

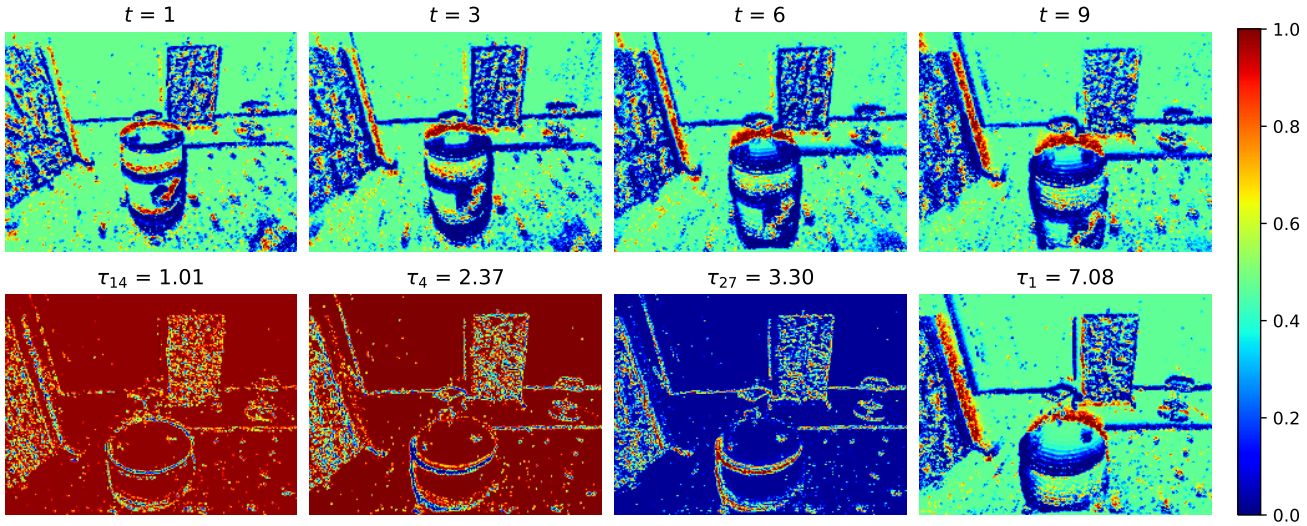

Figure 1. Feature maps of CTC. The upper row shows the evolution of a feature map at four different time steps ($t = 1, 3, 6, 9$). The feature map aggregates its past states and gradually forms a denser spatial representation. Compared DTC, the feature map of CTC have more uniformed color on background scenes, due to a direct sum from the previous membrane potential without normalization of the sigmoid activation function in DTC. The bottom row shows four feature maps with different time constants at the end of evolution. Channels with larger τ remember more history information than those with smaller τ.

to DDES) and the maximum matching channel to 256 correspondingly. We set the d_{max} for the matching volume to 128 (The default value for MVSEC was 64, corresponding to max disparity 37 of the dataset. The max disparity of the DSEC dataset is approximately 77, so we scaled d_{max} twice larger accordingly). We use RMSprop optimizer and learning rate 0.005 for the DTC module and 0.001 for the rest of the network. We use multi-step scheduler with learning rate halve in epoch 11, 22 and 33. Preliminary results (table 3) show that DTC-SPADE achieves state-of-the-art performance among event-only methods.

References

[1] Dsec disparity benchmark. https://dsec.ifi.uzh.ch/uzh/disparity-benchmark/. 3

[2] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*, 6(3):4947–4954, 2021. 3

[3] Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. A natural lottery ticket winner: Reinforcement learning with ordinary neural circuits. In *International Conference on Machine Learning*, pages 4082–4093. PMLR, 2020. 1

[4] Mohammad Mostafavi, Kuk-Jin Yoon, and Jonghyun Choi. Event-intensity stereo: Estimating depth by the best of both worlds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4258–4267, 2021. 3

[5] WH Pressa, SA Teukolsky, WT Vetterling, and BP Flannery. Numerical recipes 3rd edition: The art of scientific computing, 2007. 1

[6] Stepan Tulyakov, Francois Fleuret, Martin Kiefel, Peter Gehler, and Michael Hirsch. Learning an event sequence embedding for dense event-based deep stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1527–1537, 2019. 1, 3

[7] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 2