Supplementary materials for paper: Learning an event sequence embedding for dense event-based deep stereo.

Stepan Tulyakov
Space Engineering Center at
École Polytechnique Fédérale de Lausanne
stepan.tulyakov@epfl.ch

Francois Fleuret
École Polytechnique Fédérale de Lausanne
and Idiap Research Institute
francois.fleuret@idiap.ch

Martin Kiefel, Peter Gehler, Michael Hirsch
Amazon Research Tuebingen
{mkiefel, pgehler, hirsch}@amazon.de

1. Network architecture

The architecture of the proposed stereo network is shown in Table 3. In this network, the temporal aggregation can be implemented as either a hand-crafted function, a temporal convolutional network or the continuous fully-connected layer as described in the paper in §2.2. The best-performing architecture of the temporal convolutional network is shown in Table 1. The best-performing architecture of the kernel network in the continuous fully-connected layer is shown in Table 2. The temporal convolutional network and the continuous fully-connected layer performs best with an event queue with a capacity of $\kappa = 7$ events and a time horizon of $\tau = 0.5$ seconds, while the hand-crafted aggregation performs best with an event queue of infinite capacity and a time horizon of $\tau = 0.2$ seconds. All best-performing architectures were selected based on experiments with shallow stereo network from [3].

Table 1. Architecture of the best-performing temporal convolutional network. The convolutions are computed without padding. The network takes as an input an event queue of size $2 \times 7 \times h \times w$ and produces an event image of size $64 \times h \times w$. By using 3d convolutions with a kernel size of $3 \times 1 \times 1$ (instead of 1d convolutions with a kernel size of 3), we can process the entire event queue. The network has 25k parameters in total.

Layer Description	Output Size
3D conv. $2 \times 3 \times 1 \times 1 \times 64 \circ$ ReLU	$64 \times 5 \times h \times w$
$2 \times$ 3D conv. $64 \times 3 \times 1 \times 1 \times 64 \circ$ ReLU	$64 \times 3 \times h \times w$
3D conv. $64 \times 3 \times 1 \times 1 \times 64 \circ$ ReLU	$64 \times h \times w$

2. Training parameters

In our experiments, all networks are trained for 12 epochs using the full-sensor event sequences, without augmentation. The learning rate is set to 0.1 for the hand-crafted temporal aggregation, 10^{-4} for the temporal convolutional network. In the experiment with the continuous fully-connected layer the learning rate is 10^{-3} for the kernel network and 10^{-4} for the rest of the network. In all cases the learning rates are kept fixed for 8 epochs and then are halved every 2 epochs.

Table 2. Architecture of the best-performing kernel network in the fully-connected continuous layer. The network takes as an input event timestamps of size $1 \times 7 \times h \times w$ from the event queue and produces a weights tensor of size $64 \times 7 \times h \times w$. By using 3d convolutions with kernel $1 \times 1 \times 1$ instead of a fully-connected layer we can process the entire event queue. The network has 12.5k parameters in total.

Layer Description	Output Size
3D conv. $1 \times 1 \times 1 \times 1 \times 64 \circ$ ReLU	$64 \times 7 \times h \times w$
$2 \times$ 3D conv. $64 \times 1 \times 1 \times 1 \times 64 \circ$ ReLU	$64 \times 7 \times h \times w$
3D conv. $64 \times 1 \times 1 \times 1 \times 64$	$64 \times 7 \times h \times w$

3. Kernel network initialization

For the kernel network we developed a custom initialization. Usually network weights are initialized using a normal distribution $\mathbb{N}\left(0, \frac{2}{N_l + N_{l-1}}\right)$, where N_{l-1} and N_l are the numbers of inputs and outputs respectively. This initialization is called Xavier initialization [1] and it ensures that the variances of network activations and parameter gradients are kept constant across all layers. Since the kernel network essentially produces weights of the continuous fully-connected layer, we initialize its parameters such that its output is normally distributed. This is done computationally by sampling weights of the continuous fully-connected layer for timestamps $t_1, t_2, \ldots t_M$ from a normal distribu-

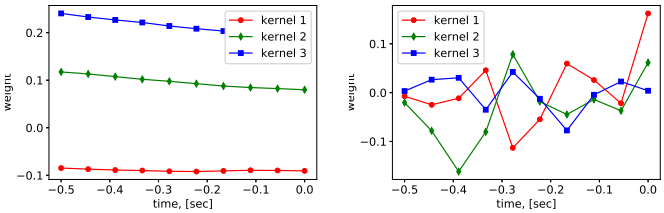
Table 3. Architecture of the proposed stereo matching network. The residual blocks consist of two 2d convolutions followed by shortcut connections. The convolutions and transposed convolutions, including these in the residual blocks, are followed by LeakyReLU with negative slope 0.2 and Instance Normalization (IN) [2], unless explicitly stated otherwise. The network receives as an input left and right event queues of size $2 \times 7 \times h \times w$ and returns disparity tensor of size $h \times w$.

#	**Layer Description**	**Output Size**
	Temporal aggregation	
	Please, refer § 2.2 in the paper.	$64 \times h \times w$
	Spatial aggregation	
S1	2D conv. $3 \times 5 \times 5 \times 64$ stride 2	$64 \times \frac{1}{2}h \times \frac{1}{2}w$
S2	2D conv. $64 \times 5 \times 5 \times 64$ stride 2	$64 \times \frac{1}{4}h \times \frac{1}{4}w$
S3	$2\times$ residual block with $64 \times 3 \times 3 \times 64$ 2D conv.	$64 \times \frac{1}{4}h \times \frac{1}{4}w$
S4-redir.	2D conv. $64 \times 3 \times 3 \times 8$ no IN, LeakyReLU	$8 \times \frac{1}{4}h \times \frac{1}{4}w$
	Matching module	
M1	concatenate left-right embeddings S3	$128 \times \frac{1}{4}h \times \frac{1}{4}w$
M2	2D conv. $128 \times 3 \times 3 \times 64$	$64 \times \frac{1}{4}h \times \frac{1}{4}w$
M3	$2\times$ residual block with $64 \times 3 \times 3 \times 64$ 2D conv.	$64 \times \frac{1}{4}h \times \frac{1}{4}w \times 64$
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. $8 \times 3 \times 3 \times 3 \times 16$, 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	
	Please, refer Equation 2 in the paper	$h \times w$

Table 4. Average event rate (number of events per second) for all sequences of the Indoor Flying dataset. Note, that the average event rate for the second sequence is almost two times higher than the one for the other sequences.

Average events rate, [events/second]		
Sequence 1	**Sequence 2**	**Sequence 3**
180'000	280'000	190'000

tion $\mathbf{W} = [\mathbf{w}(t_1), \ldots, \mathbf{w}(t_M)] \sim \mathbb{N}\left(0, \frac{2}{N_l + N_{l-1}}\right)$ and fitting the kernel network to these weights. Besides keeping the variances in check, this initialization ensures diversity of the resulting continuous kernels. Its effect is shown in Figure 1. The bias weights of the fully-connected continuous layer are initialized with zeros.



(a) w/ initialization (b) w/o initialization

Figure 1. Continuous kernels before training with and without our custom initialization of the kernel network. Note, that without the custom initialization (a) the continuous kernels are mostly linear and very similar to each other, whereas with our proposed initialization (b) they have complex and diverse shapes. For clarity, we show 3 kernels out of 64.

4. Problem with the second split

During our experiments with various splits of the Indoor Flying dataset we noticed significant differences between the test and the training set of the second split. In the test set, composed of the second sequence, there are much more abrupt motions (triggering large numbers of events) than compared to the training set, composed of sequences one and three as shown in Table 4. This suggests that the test and the training set of this split are drawn from very different underlying distributions and thus should not be used in the experiments.

5. Videos

We present two videos showing results of our proposed method for sequences one and three. To compute the result for sequence one, we used the network trained on the sequences two and three (split one) and to compute the result for sequence three, we use the network trained on the sequences one and two (split three). We do not provide the result for the sequence two (split two) for the reasons discussed in § 4. The videos contain take-off and landing frames without events, which are not used during training and test time. The ground truth along with our results of our proposed method are shown with the same adaptive color-coding, *i.e.* warmer colors correspond to closer objects. Locations with unknown disparities are displayed in white. The bottom-left panel in the videos shows the left camera events that arrived during the last 0.5 seconds, which constitutes the input of our method together with the corresponding right camera events. The events are overlaid with a gray-scale image, which is not used by our method. Positive events are shown in red and negative events are shown in blue.

References

[1] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 1

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

[3] Jure Žbontar and Yann LeCun. Computing the Stereo Matching Cost With a Convolutional Neural Network. *CVPR*, 2015. 1