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, Tübingen, Germany
{mkiefel, pgehler, hirsch}@amazon.de

Abstract

Today, a frame-based camera is the sensor of choice for machine vision applications. However, these cameras, originally developed for acquisition of static images rather than for sensing of dynamic uncontrolled visual environments, suffer from high power consumption, data rate, latency and low dynamic range.

An event-based image sensor addresses these drawbacks by mimicking a biological retina. Instead of measuring the intensity of every pixel in a fixed time interval, it reports events of significant pixel intensity changes. Every such event is represented by its position, sign of change, and timestamp, accurate to the microsecond.

Asynchronous event sequences require special handling, since traditional algorithms work only with synchronous, spatially gridded data. To address this problem we introduce a new module for event sequence embedding, for use in different applications. The module builds a representation of an event sequence by firstly aggregating information locally across time, using a novel fully-connected layer for an irregularly sampled continuous domain, and then across discrete spatial domain. Based on this module, we design a deep learning-based stereo method for event-based cameras. The proposed method is the first learning-based stereo method for an event-based camera and the only method that produces dense results. We show large performance increases on the Multi Vehicle Stereo Event Camera Dataset (MVSEC), which became the standard set for the benchmarking of event-based stereo methods.

1. Introduction

Stereo matching is the problem of finding for every point in an image taken from one viewpoint its physically corresponding one in an image taken from another viewpoint. Given the parameters of a stereo camera setup, the matching results allow to compute the 3d structure of a scene, with many applications, *e.g.* in robotics [29], medical imaging [32], remote sensing [47], or computational photography [55, 2].

1.1. Deep stereo for frame-based cameras

Currently, most successful stereo matching methods are based on deep learning. First successes of deep learning in stereo matching were achieved by replacing individual algorithmic elements in legacy methods (often [28, 16]) with neural networks, *e.g.* similarity metric [62, 24, 51, 61, 8], smoothness penalty [45, 21], matching confidence [46] and disparity post-processing [13].

Current works solve the stereo matching by training a neural network end-to-end, which combines embedding, matching, regularization, and sometimes refinement modules in a single model [10, 27, 20, 63, 36, 19, 23, 7, 52, 59, 50]. An embedding module computes image descriptors for left and right images, a matching module performs a correlation [10, 27, 36, 19, 23, 59, 50], computes matching signatures [52] or simply concatenates [20, 7, 63] left and shifted right descriptors for every disparity. The regularization module, implemented as an hourglass network with shortcut connections between the contracting and the expanding parts and 2d [27, 10, 36, 23] or 3d [20, 52, 63, 19, 7] convolutions, enforces stereo matching constraints and computes disparities or a distribution over disparities. Finally, some methods [36, 23, 19] have a refinement module, that improves the initial low-resolution disparity relying on left-right warping error.

Best results are obtained with fully-supervised training on large synthetic datasets with ground truth [27] and an L^1 or cross-entropy [52] loss, while some methods use weakly supervised settings [63, 39, 59], relying on geometric constraints of the task. Most recent methods [59, 50] further

Table 1. Comparison of event-based image sensor, such as [3] to a frame-based sensor. The numbers show orders of magnitude for every characteristic, rather than precise values. Advantages of event-based sensors are highlighted.

Characteristic	Frame-based	Event-based
Dynamic range, [dB]	50	130
Power consumption, [W]	1	0.01
Data rate, [Mb/s]	100	0.1
Latency, [ms]	10	0.001
Resolution, [MP]	1	0.01
Intensity information	✓	✗

improve results using multi-task learning.

Many stereo methods exist for frame-base cameras, however, stereo matching with novel bio-inspired event-based sensors is a relatively new area of research, with many interesting research challenges.

1.2. Event-based sensors

Conventional frame-based camera sensors capture stroboscopic sequences of still pictures at a fixed time interval or frame rate. In contrast, the retina in the human eye operates on completely different principles. Nobel prize wining experiments [18] showed that the retina is most sensitive to temporal brightness gradients, and is blind to static scenes in absence of eye movements [41]. These principles inspired the development of event-based image sensors [49, 3].

In an event-based image sensor, pixels are sensitive to temporal brightness contrast and trigger binary events with a rate proportional to the temporal gradient of photocurrent. Events can have positive or negative polarity depending on the sign of the gradient (*i.e.* "dark to bright" or "bright to dark"). Triggered events are recorded from the sensor in an asynchronous manner with information about their spatial positions, polarities, and timestamps accurate to microseconds.

Event-based image sensors have several advantages over frame-based sensors. First, they have higher dynamic range and thus do not saturate in extreme lighting conditions, such as bright daylight and night with minimum illumination, thanks to pixel-wise gain and integration time control. Secondly, they have lower power consumption and data rates, since they only transmit information about significant brightness changes. This enables use in power-constrained systems. Finally, due to immediate transmission of every triggered event with a microsecond-accurate timestamp, event-based sensors have lower latency and can be used in time-critical applications. However, these advantages come at a cost. Event-based sensors have lower resolution, because their pixels are more complex, and do not provide rich intensity information. A comparison between frame-based and event-based sensors is shown in Table 1.

Unique properties of event-based image sensors make them attractive for low-latency dynamic vision applications in environments with uncontrolled illumination, such as tracking [12], robot control [25], or object recognition [48]. Depth estimation, that we investigate in this work, could drastically improve performance in these applications, and open the way to novel use cases, such as augmented reality.

1.3. Event-based stereo

Due to the novelty of event-based image sensors, only a few event-based stereo methods have been proposed, none of which is learning-based.

One line of research investigates how to represent and compare events. This is a hard problem because events have few spatio-temporal neighbors and only one binary feature. Early methods [22, 42] compared events using only their timestamps which led to matching ambiguities, due to the noise, variable cameras sensitivity, and imperfect camera synchronization. Therefore, later methods relied on hand-crafted descriptors [4, 68, 69] and similarity measures [44, 67, 38]. In [4] descriptors are computed as a bank of orientation-sensitive spatial filter responses, in [68] as a vector of distances to closest events in several spatial directions, in [69] as a histogram of orientations of vectors pointing to the closest events in a spatial window. As for similarity measures, spatial windows with event are compared in [44] using average distances to the closest events, in [38] using average inverse timestamp difference between corresponding events, and in [67] using intersection-over-union of events.

The second research direction explores how to apply regularization in stereo matching. This is a challenging problem due to the sparsity of data in both time and space, which consequently cannot be represented with conventional Markov Random Field (MRF) models. Some works [37, 11, 38] adopt heuristic cooperative regularization from [26] by defining a spatio-temporal inhibitory and excitatory neighborhood for each event, while others [56, 57] use belief propagation and semi-global matching on sparse MRF models, were nodes are active only during a fixed interval after receiving an event.

The third line of research explores how to accumulate events over time to cope with the fact of individual events being noisy and not very informative, though leading to the undesirable effect of blurring object boundaries when using long accumulation times. Most methods use fixed accumulation intervals [22, 42, 4, 44], while [69] sets accumulation time equal to the average of the inverse of the event rate, and [67] warps event positions as if they were all triggered at the same time using depth hypothesis and known camera motion.

Another challenging problem is to perform dense stereo matching using sparse event data. While most of the methods produce sparse disparity, the work [64] reconstructs semi-dense disparities by fusing information from several view-points using a known camera motion. In [69] a disparity is computed at every location without an event by fitting a plane to its neighbor disparities.

Finally, a variety of works is dedicated to implement and perform stereo matching on neuromorphic chips and field-programmable gate arrays (FPGA) [1, 9].

All existing methods use hand-crafted event representations and grid-based image models, which support only simplistic priors, such as smoothness. Meanwhile, most successful frame-based stereo methods use learning-based representations [61] and regularization based on deep networks, which are able to perform area-based regularization [20] and even use monocular depth cues [14]. This motivates our work: an end-to-end deep learning model for event-based stereo matching.

1.4. Deep learning with event sequences

Event sequences can be processed using convolutional neural networks (CNNs), recurrent neural networks (RNNs) and specialized networks for event data.

One option is to use off-the-shelf CNNs, that are successful in frame-based image processing. However, one problem is that convolutional modules work with dense images, where each pixel lies in a 2d or 3d (in case of video) discrete space, with an intensity value assigned to it. In contrast, an event sequence is a sparse number of 3d points, with two discrete spatial dimensions, one continuous temporal dimension, and with a binary variable (polarity) as a feature. Therefore, such a sequence needs to be transformed to a frame-based representation before it can be input to a standard CNN. Note, however, that naive binning of the time dimension is problematic since it would produce tensors with a prohibitively large temporal dimensions.

Existing methods [34, 25, 30, 53, 66, 60] use hand-crafted transformations to convert event sequences to frame-based representations, that we call *event images*. For example, [34] saves the polarity of the last event, [30] sums event polarities in every location during a predetermined time interval, and [25] counts the number of positive and negative events to avoid information loss due to polarity cancellation. To preserve time information, in addition to positive and negative event counts, [66] stores the timestamp of the last positive and negative events at every location, while [60] saves the average timestamps of the updates. To capture time dynamic, [53, 60] stack several event images described earlier, for consecutive time intervals. The main drawback of all these representations is that they loose precise timing information about events.

Another seemingly natural choice, given the sequential nature of the data, is to use RNNs. An application to even-based recognition with RNNs and long-term memory is [33]. That model has the drawback that it does not preserve the spatial information, which is a crucial ingredient for example to stereo matching. Note, that convolutional RNNs, such as [58], preserve spatial information, but are not applicable for the same reason as CNNs.

Finally, one can use asynchronous networks, where every neuron has an internal state that is updated by events, such as Spiked Neural Networks (SNN) [6] or specially designed convolutional neural networks [5, 35]. Unfortunately, it is hard to build and train such networks, because they are not easily differentiable, and additionally difficult to implement in a standard framework that enable the use of available computational back-ends such as GPUs.

1.5. Contribution

The contributions of this paper are the following:

1. We propose a learnable representation (embedding) for event sequences that, explicitly treats sequences as stream of sparse 3d points with two discrete spatial and one continuous temporal coordinate. It takes into account both spatial positions and accurate timeing information of all recorded events.

2. We use this embedding to design the first deep learning-based method for event-based stereo reconstruction. The method is based on an architecture with large receptive field that uses large context and allows stereo reconstruction for locations without events. We demonstrate that this method significantly outperforms other competing approaches on the Multi Vehicle Stereo Event Camera Dataset (MVSEC).

2. Method

Let the left and right event sequences be E^l, E^r, each consisting of n events sorted by the time of arrival $E = ((x_i, y_i, t_i, p_i) \mid t_{i+1} > t_i)_{i=1\dots n}$. Each event is a point in a three dimensional space with two discrete spatial coordinates and one continuous temporal coordinate $(x, y, t) \in [0 \dots w) \times [0 \dots h) \times \mathbb{R}$ and one polarity feature $p \in \{-1, 1\}$, where w and h correspond to the width and height of an image sensor. Given both E^l, E^r, the network computes an *estimated disparity* tensor $\widehat{\mathbf{D}}$ as

$$\widehat{\mathbf{D}} = \text{Net}(E^l, E^r \mid \Theta, d_{max}) \in [0, d_{max}]^{h \times w}, \qquad (1)$$

where Θ is the tensor of network parameters and d_{max} is a maximum disparity. An element of the disparity tensor $\widehat{D}_{y,x}$ specifies that a pixel with coordinates (x, y) from the left camera matches the pixel with coordinates $(x - \widehat{D}_{y,x}, y)$ from the right camera. In §2.1 we describe the proposed network architecture for event-based stereo matching, and in §2.2 explain the novel embedding for event sequences.

2.1. Network architecture

Our architecture is inspired by recent stereo networks for frame-based cameras, which use large image contexts and produce disparities with sub-pixel accuracy [52, 20]. It consists of *embedding*, *matching*, *regularization* modules, followed by an *estimator*. The embedding module takes as input an event sequence E and computes its descriptor $\mathbf{F}$ of size $c \times \frac{h}{4} \times \frac{w}{4}$. The same module is applied to the left and right event sequence independently. This approach will be described in detail in §2.2. The matching module for each disparity then takes the left descriptor and shifted right descriptor and computes a *matching signature* of size $\frac{c}{8} \times \frac{h}{4} \times \frac{w}{4}$. All matching signatures for all disparities are concatenated to a 4d tensor of size $\frac{c}{8} \times \frac{d_{max}}{4} \times \frac{h}{4} \times \frac{w}{4}$, which are then passed to the regularization module. The regularization module is an hourglass neural network with 3d convolutions and shortcut connections between the contracting and the expanding parts. It produces a *matching cost* tensor $\mathbf{C}$ of size $\frac{d_{max}}{2} \times h \times w$ (matching costs are computed only for even disparities to save space), passed to the sub-pixel estimator [52] which produces an estimated disparity tensor $\widehat{\mathbf{D}}$ as

$$\widehat{D}_{y,x} = \sum_{j} d(j) \cdot \underset{j:|\hat{j}-j|\le\delta}{\operatorname{softmin}} \left(C_{j,y,x}\right)$$
$$\text{with } \hat{j} = \arg\min_{j} \left(C_{j,y,x}\right), \quad (2)$$

where $\Delta = 2$ is an *estimator support* and $d(j) = 2 \cdot j$ is a disparity, corresponding to index j in the matching cost tensor. More details about the network architecture are described in the supplementary material.

Network training uses a sub-pixel cross entropy loss [52]

$$L(\Theta) = \frac{1}{wh} \sum_{y,x} \sum_{j} \operatorname{Laplace}(d(j) \mid \mu = D^{GT}_{y,x}, b) \times$$
$$\log(\underset{j}{\operatorname{softmin}}(C_{j,y,x})), \quad (3)$$

where $\operatorname{Laplace}(d \mid \mu = D^{GT}_{y,x}, b)$ is a discretized and normalized Laplace probability density function over disparities with mean equal to the ground truth disparity $\mu = D^{GT}_{y,x}$ and diversity $b = 2$.

2.2. Events sequence embedding

In this work, we focus on a special family of embedding functions that can be represented as a composition $f_S(f_\tau(\cdot))$ of two functions: *temporal aggregation* $f_\tau(\cdot)$ and *spatial aggregation* $f_S(\cdot)$. The temporal aggregation function is defined per-location and it takes a *local event sequence* $E(x,y) = ((x_i, y_i, t_i, p_i) \in E \mid (x_i, y_i) = (x,y))$ as input, and produces a *event image* $\mathbf{I}$ of size

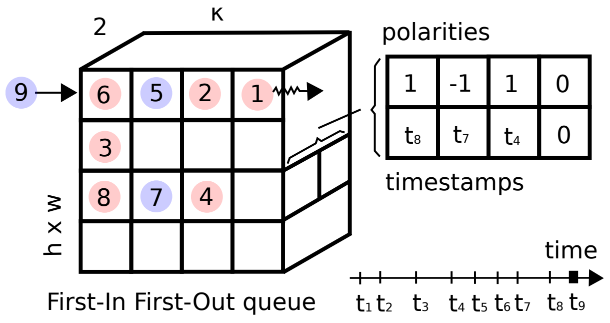


Figure 1. The **event queue** is a 4d tensor of size $2 \times \kappa \times h \times w$, where κ is a *queue capacity*. The figure depicts the queue with width and height fused to a single dimension. It stores polarities and timestamps of the κ most recent events at each location in order of arrival. When a new event, here (9), arrives, it pushes older events (6, 5, 2) to the end of the queue and occupies the first position and the oldest event (1) is pushed out of the queue.

$c \times h \times w$, as $I_{y,x} = f_\tau(E(x,y))$. The spatial aggregation is a translation-invariant function that is applied to sub-windows of the event image and produces a event descriptor $\mathbf{F}$ such that $\forall y, x,\ F_{y,x} = f_S\left(\mathbf{I}_{y-\Delta:y+\Delta, x-\Delta:x+\Delta}\right)$.

The spatial grid structure of the event image allows the use of standard 2d convolutions. Therefore, throughout our experiments we use two convolutional residual blocks [15] and focus on different temporal aggregation methods.

To implement different temporal aggregation methods we need a way to efficiently accumulate events in each location. For that we propose to use a First-In First-Out (FIFO) queue shown in Figure 1. It saves the κ most recent events at each location sorted by time of their arrival. This queue could be efficiently implemented using linked lists or simpler circular buffers. Note also, that this queue works well regardless of the amount of motion: in presence of fast motion, when events are frequent, it stores only the recent ones, while in presence of slow motion, when events are rare, it preserves old ones. We prune events that arrived more than τ seconds ago from the queue and replace them with zeros before applying the temporal aggregation. We call κ the *capacity* and τ the *time horizon* of the queue.

Hand-crafted. In §1.4 we reviewed existing methods for converting event sequences to event images. All of them can be thought of as hand-crafted temporal aggregations. One of these methods produces an event image by counting the number of positive and negative events, and recording timestamps of the most recent positive and negative events at every location. Since similar methods [34, 25, 30, 53, 66, 60] worked well in many applications, we use this solution as our baseline.

Temporal convolutional network. Temporal convolutional network seems like a natural choice for temporal aggregation. However, a convolutional network usually applies to regularly sampled data, whereas in our case event timestamps are sampled irregularly and the temporal dimen-

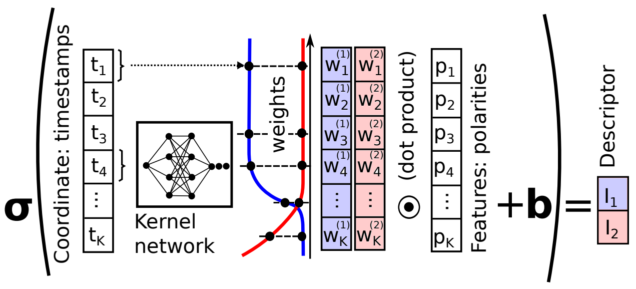

σ
Coordinate: timestamps
t_1
t_2
t_3
t_4
t_K
Kernel network
weights
$w_1^{(1)}$ $w_2^{(1)}$ $w_3^{(1)}$ $w_4^{(1)}$ $w_K^{(1)}$
$w_1^{(2)}$ $w_2^{(2)}$ $w_3^{(2)}$ $w_4^{(2)}$ $w_K^{(2)}$
(dot product)
p_1 p_2 p_3 p_4 p_K
Features: polarities
+b
=
Descriptor
l_1
l_2

Figure 2. In the **continuous fully-connected layer** depicted here, for every timestamp the kernel network computes two weights, and for all timestamps two vectors, each corresponding to a continuous kernel. To get an event sequence descriptor we multiply each of these vectors by polarity vector using dot product, add bias and apply non-linearity. The corresponding weight vector, descriptor element and continuous kernel are shown in same color.

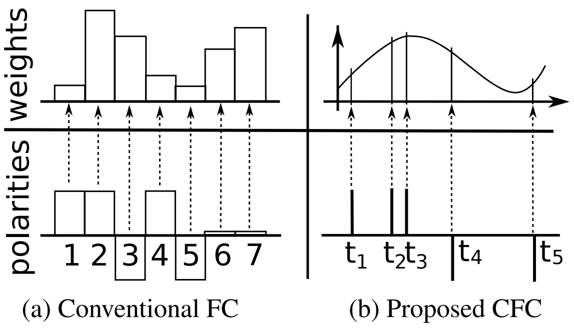


Figure 3. Comparison of (a) conventional fully-connected (FC) layer to (b) proposed continuous fully-connected (CFC) layer. In contrast to FC, CFC allows to embed event sequences with irregularly spaced time intervals between events.

sion in the queue only reflects the order of event arrival. Actual timestamp difference between nearby events in the queue might be different and arbitrary. To compensate for that, we feed the timestamp of each event to the network along with its polarity as a feature. Details of the network with temporal convolutions can be found in our supplementary material.

Continuous fully-connected layer. Ideally, the fact that event timestamps are continuous and sampled irregularly is taken into account. To do so, we use a *continuous fully-connected layer (CFC)*, where continuous kernels are themselves approximated by a multi-layer perceptron (MLP), that we call a *kernel network*. This network allows to model arbitrary complex kernels by modulating their capacity, and can be trained end-to-end along with the rest of the architecture. The overall idea is illustrated by Figure 2. Details about kernel networks can be found in our supplementary material.

Let us compare the proposed layer to a standard fully-connected (FC) layer, to appreciate the differences. Given event polarities $\mathbf{p} = [p_1, p_2, p_3, p_4, p_5, 0, 0]$ for some location stored in the event queue, a single output of the conventional FC layer is computed as $I = \sigma(\sum_i^7 w_i p_i + b)$, where $\mathbf{w}$ is a weights vector, b is a bias and $\sigma(\cdot)$ is a non-linearity. In contrast, a single output of the proposed CFC layer is computed as $I = \sigma(\frac{1}{5}\sum_i^5 w(t_i) \cdot p_i + b)$. Note that as shown in Figure 3, for a standard FC layer, the weight of each polarity simply depends on the events order i, while for the proposed CFC, the weight is a continuous parametric function $w(t_i) = \text{KernelNet}(t_i)$ (MLP), of real-valued event timestamp t_i. This allows to embed event sequences with irregularly spaced time intervals between events.

A similar construction was used in [54] but with the use of continuous convolutional layers. Here, we propose continuous fully-connected layer. Another difference is that in [54] the input are 3d LIDAR points in a Euclidean space.

3. Experiments

All experiments are done using the PyTorch framework [40]. Network learning uses RMSprop with standard settings. In all experiments we normalize event polarities in the queue to $\mathcal{N}(0,1)$ and subtract the timestamp of the most recent event from all other event timestamps.

All experiments are done on publicly available datasets, and our code is available on GitHub [1].

3.1. Dataset and evaluation protocol

We use the Multi Vehicle Stereo Event Camera Dataset (MVSEC) [65] which is available online [31]. MVSEC is the only large publicly available dataset captured with a real event-based stereo system, and over recent years it has became the de-facto standard for comparing event-based stereo methods [67, 64]. It is collected by a system composed of a LIDAR and two event-based cameras with a resolution of 346×260 pixels mounted on various vehicles, such as a drone, a car and a motorcycle. LIDAR records frames with sparse depth measurements at 20Hz, while the event-based cameras acquire continuous streams of events and gray-scale video frames, which we use for visualization purposes only.

We unpack the original data in ROS bag format [43]. The depths are converted to left-view sub-pixel disparities and saved as images (sub-pixel precision is preserved by scaling the disparities). All pixels with disparities > 36 are assumed to have unknown disparities. Then, for each depth, we find the closest gray-scale image in time, and events preceding the depth by 0.05 seconds in the left and right view. We correct their optical distortions, rectify and save them. The script for data conversion is available online with the rest of the source code.

We use the *Indoor Flying* dataset from MVSEC, which is captured from a drone flying in a room with various ob-

[1]https://github.com/tlkvstepan/event_stereo_ICCV2019

Table 2. Summary of Indoor Flying splits. For each split we specify which sequences and frames are used for training and test. For example, $S^1_{140,..,1200}$ means that from sequence one only the frames 140 to 1200 are used. We use the same test intervals as in [67] to allow a fair comparison.

#	Set	Sequence and frames	Size
1	Training	$S^2_{160,..,1580} \cup S^3_{125,..,1815}$	3110
	Validation	$A \in S^1_{140,..,1200}$	200
	Test	$B \in S^1_{140,..,1200} \mid A \cap B = \emptyset$	861
2	Training	$S^1_{80,..,1260} \cup S^3_{125,..,1815}$	2870
	Validation	$A \in S^2_{120,..,1420}$	200
	Test	$B \in S^2_{120,..,1420} \mid A \cap B = \emptyset$	1101
3	Training	$S^1_{80,..,1260} \cup S^2_{160,..,1580}$	2600
	Validation	$A \in S^3_{73,..,1615}$	200
	Test	$B \in S^3_{73,..,1615} \mid A \cap B = \emptyset$	1343

jects. We compare our method to existing methods using the protocol from [67] and report results on the three sequences. The results are summarized in Table 2. Following [67], take-off and landing frames are removed. The test sequences are the same as in [67].

Similar to [67], we compute and report the *mean depth error* (MDE) and *one-pixel-accuracy* (1PA) computed in sparse locations corresponding to 15'000 events preceding each depth measurement. The one-pixel-accuracy is the percentage of locations for which the predicted disparity is off by less than one pixel.

3.2. Comparison of temporal aggregation methods

In this section, we compare performance of the temporal aggregation variants described in §2.2. We train the network three times for each method using different random initializations. For every trial we select the network that achieves the highest 1PA on the validation set over all epochs. The selected networks are then used to compute the performance on the test set. In Table 3 we report average test results along with standard deviations.

For each variant we use the architecture that was found during the grid-search experiments with the shallow stereo network from [61] on validation set. During training, we consider only ground truth at locations corresponding to the most recent 15'000 events.

All networks are initialized using the default PyTorch initialization, except the kernel network, for which we developed a custom initialization that ensures that the outputs of the network follow a normal distribution. More details can be found in the supplementary materials.

As shown in Table 3, the proposed learning-based methods for temporal aggregation outperform the hand-crafted method, probably due to the fact that they utilize timestamps of individual events. Among the learning-based methods, the network with the continuous fully-connected layer shows the best performance as it explicitly handles events which are irregularly sampled from the continuous time domain. In all following sections we use the latter method, and call the resulting stereo matching method *Deep Dense Event Stereo* (DDES).

Table 3. Empirical results on the first split test set of the Indoor Flying dataset. Shown is the average test set results over three trials with the best performing method highlighted. Note, that all proposed learning-based methods outperform the hand-crafted method.

Method	MDE, [cm]	1PA, [%]
Hand-crafted	16.5 ± 0.5	87.3 ± 0.2
Temporal convolutional network	13.8 ± 0.1	90.7 ± 0.1
Continuous fully-connected layer	13.6 ± 0.2	91.3 ± 0.9

3.3. Empirical results

Next, we compare the proposed stereo method to the state-of-the-art event-based methods [67, 64, 38], and to two traditional methods [16, 17] which were adopted to work on event images in [64].

For quantitative comparison we use the protocol from [67] described in §3.1. According to this protocol, results are evaluated in sparse locations corresponding to 15'000 most recent events. We use the same parameters and experiments settings as in §3.2. During the experiments we noticed that for the second split there is a significant difference between test and training set. The test set has more abrupt motions, triggering a larger number of events compared to the training set (for details please refer the supple-

Table 4. Results on the Indoor Flying dataset using sparse ground truth, following the protocol from [66] described in §3.1. Results for TSES [67] and CopNet [38] are from [67] and results for Semi-Dense 3D [64], SGM* [16, 64] and FCVF* [17, 64] are from [64]. SGM* and FCVF* methods implemented in [64] are similar to the original frame-based methods but operate on event images. For Semi-Dense 3D, SGM* and FCVF* results for the second split are not available. We report average test set errors including standard deviations over the three randomized training trials. For other methods the standard deviation are not available. All methods are sorted in ascending order according to their test error. Our proposed method dubbed Deep Dense Event Stereo (DDES) is highlighted. Note, that it outperforms other single viewpoint methods, such as TSES, CopNet, SGM* and FCVF*, and even performs on-par with Semi-Dense 3D method that fuses depths from several viewpoints using known camera motion.

Method	Mean depth error, [cm]		
	Split 1	Split 2	Split 3
Semi-Dense 3D [64]	13	–	33
DDES (proposed)	13.6 ± 0.2	18.0 ± 0.2	18.4 ± 0.5
TSES [67]	36	44	36
CopNet [38]	61	100	64
SGM* [16, 64]	93	–	119
FCVF* [17, 64]	99	–	103

Table 5. Performance on the Indoor Flying dataset evaluated using dense ground truth. We train our method using the full ground truth disparity, taking into account all locations, including those without events. We select the network with the highest validation 1PA during a single training pass and report its results on the test set. Note that the results are only slightly worse than results obtained using sparse ground truth.

Mean depth error, [cm]			**One pixel accuracy, [%]**		
Split 1	**Split 2**	**Split 3**	**Split 1**	**Split 2**	**Split 3**
16.7	29.4	27.8	89.8	61.0	74.8

mentary materials). As a partial remedy, for the second split we trained the network using a fixed number of 130'000 events instead of a fixed time horizon and show the results in the tables. However, we believe that due to the domain shift this split has limited significance and should not be used.

The results are summarized in Table 4. Our proposed Dense Deep Event Stereo (DDES) method performs better than other single viewpoint methods, such as TSES [67], CopNet [38], SGM* [16, 64] and FCVF* [17, 64] and even performs on-par with the Semi-Dense 3D method [64] that fuses depth from several viewpoints using known camera motion.

We also train and test our method using the entire ground truth, taking into account all locations, including those without events. We select the network with the highest validation 1PA during a single training pass and report its results on the test set. The results are summarized in Table 5. Note, that the results are only slightly worse than results using the sparse ground truth. To our knowledge, this is the first successful attempt to compute dense stereo results for event-based cameras.

For qualitative comparison we estimate disparity using DDES trained on the full ground truth for example cases similar to the ones used in [67, 64]. Figure 5 contains a visual comparison of our results with those of TSES [67] and Semi-Dense 3D [64] borrowed from the respective papers. Unlike previous techniques, DDES computes truly dense and sub-pixel accurate disparity.

Our implementation of DDES runs at about 10 frames per second on a desktop PC with a GeForce GTX TITAN X GPU.

3.4. Weights of continuous fully-connected layer.

In this section, we visualize the output of the kernel network. To this end, we input uniformly sampled timestamps $\in [-0.5, 0]$ to the kernel network and plot every row of the CFC weights tensor as a smooth curve, which we call *weight kernel*.

Resulting kernels before and after the training are shown in Figure 4. At the start of training, the output of the kernel network is (by design) normally distributed, due to the initialization. After training, the weight kernels become

Table 6. Impact of the event queue capacity on performance. The table shows validation errors for split # 1 of the Indoor Flying set averaged over 2 trials.

Queue capacity κ	**1**	**3**	**7**	**15**
Mean depth error, [cm]	13.3	13.4	13.5	13.3

smooth in time and converge to one of two shapes: bell-shaped (kernels 2 and 3) or derivative (kernel 1). The bell-shaped kernels detect events with particular timestamps, while the derivative kernels compute event count changes (time-derivative) at varying time scales. Most of the kernels assign close to zero weights to old events.

3.5. Importance of spatial and temporal context.

During our initial experiments with the temporal embedding we used the shallow stereo network with a small receptive field of size 9×9 from [61]. The shallow networks had no access to a large spatial context and larger event queue capacity and thus larger temporal context clearly helped to achieve better results. For example, with an event queue capacity $\kappa = 1$ the MDE validation error was 80.4 cm, while with $\kappa = 7$ it was 67.9 cm (the error was computed for Split 1 and averaged over two trials).

For the deep architecture from § 2.1, we noticed that the performance became very similar for different event queue capacities κ as shown in Table 6. This indicates that a network with access to a larger spatial context tends to ignore temporal context. We hypothesise, that spatial context is more reliable than temporal context, particular in dynamic sequences, such as drone videos.

4. Conclusion

In this work, we proposed a novel learning-based method for embedding event sequences as recorded by event-based vision sensors. It allows to model events as a stream of sparse 3d data points, each with two discrete spatial coordinates and one continuous temporal coordinate, and is able to use timestamps and spatial positions of all events in a time interval. We demonstrated state-of-the-art performance for

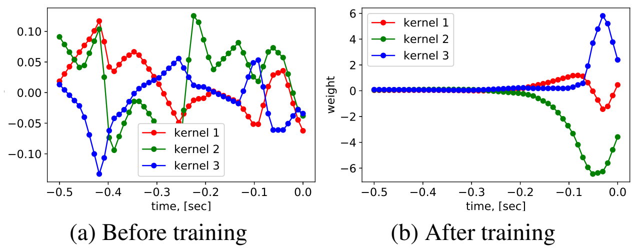
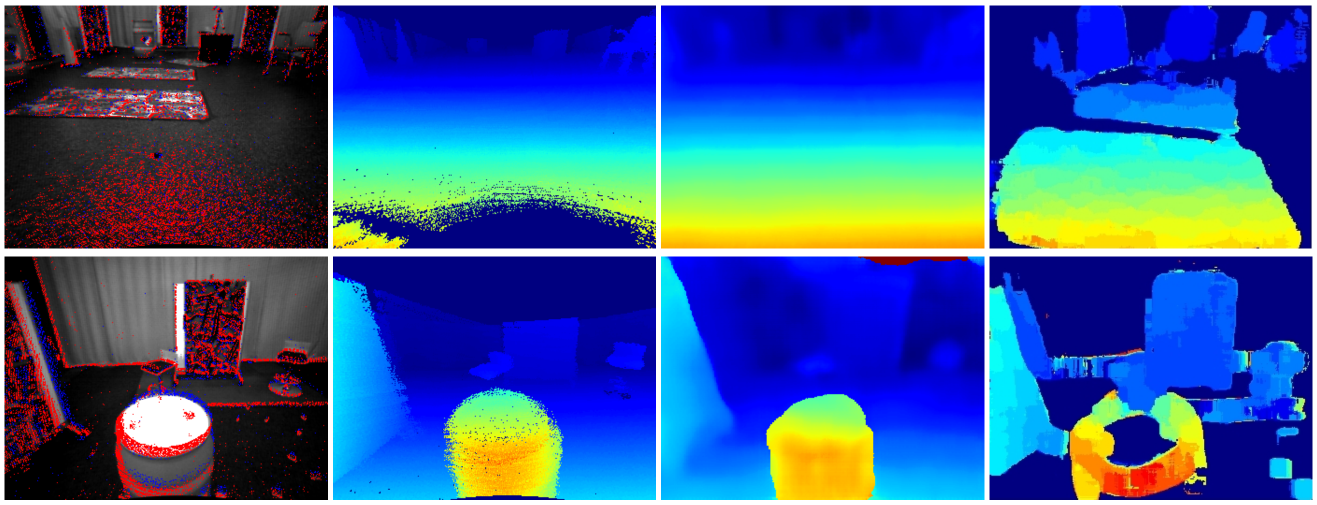
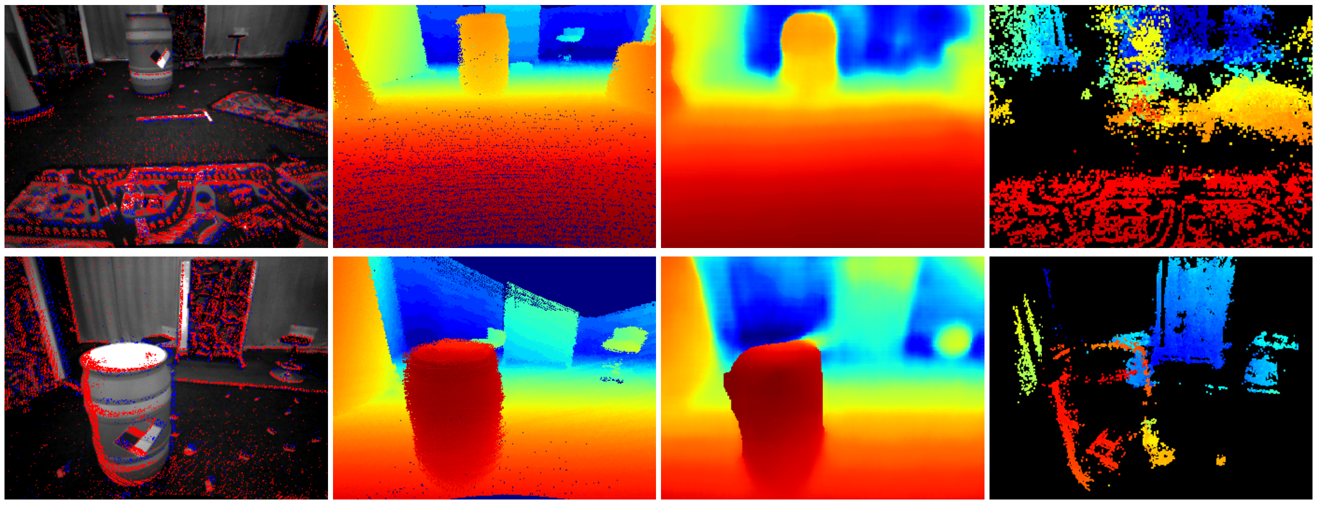


Figure 4. Visualization of kernel network output. Before training (a), the kernel network output is (by design) normally distributed. After training (b), the weight kernels have one of two shapes: bell-shaped (kernels 2 and 3) and derivative (kernel 1). Details are in the text. For clarity, we show 3 kernels out of 64.



(a) Events (b) Ground Truth (c) DDES (proposed) (d) TSES [67]



(a) Events (b) Ground Truth (c) DDES (proposed) (f) Semi-Dense 3D [64]

Figure 5. Qualitative comparison with recent event-based methods on the Indoor Flying dataset. For comparison, we select frames similar to the ones used in [67] and in [64]. Results for TSES [67] and Semi-Dense 3D [64] are borrowed from the respective papers. Note, that, unlike our method, Semi-Dense 3D fuses depth from several viewpoints using known camera motion. The rows correspond to frame #100 from sequence 1, frame #340 from sequence 1, frame #1700 from sequence 3 and frame #980 from sequence 1 correspondingly. To get the results for one sequence we trained the network using the remaining two. We tried to match the color-coding of the different outputs. In all figures warmer colors correspond to closer objects. In (a) we visualize the 15'000 most recent events from the left camera, overlaid with a gray-scale image, which is not used by the methods. Positive events are shown in red and negative events are shown in blue. In (b,c,d) locations without disparities are shown in dark blue and in (f) in black. Note, that our proposed method (c) computes dense disparities, while in TSES (d) some disparities are invalidated by outlier rejection and in Semi-Dense 3D (f) disparities are computed only for locations with events. Similarly to Semi-Dense 3D method, our proposed method (c) estimates sub-pixel disparities, while TSES (d) estimates integer disparities.

the task of stereo matching. Empirical results are better than the best hand-crafted as well as a learning-based embedding that uses on temporal convolutions in a discretized time domain. Using the proposed embedding we developed DDES, a deep neural network for stereo matching. This is the first deep learning-based stereo matching method for event-based cameras. We demonstrated that DDES perfoms better than prior state-of-the-art on the standard MVSEC dataset by a large margin.

Event-based cameras offer advantages such as higher dynamic range and temporal resolution over traditional frame-based cameras but require specialized handling of their event streams. We hope that the proposed embedding finds applications to more imaging algorithms beyond stereo matching.

References

[1] Alexander Andreopoulos, Hirak J Kashyap, Tapan K Nayak, Arnon Amir, and Myron D Flickner. A low power, high throughput, fully event-based stereo system. In *CVPR*, 2018. 3

[2] Jonathan T Barron, Andrew Adams, YiChang Shih, and Carlos Hernández. Fast bilateral-space stereo for synthetic defocus. In *CVPR*, 2015. 1

[3] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240×180 130 db 3μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014. 2

[4] Luis Alejandro Camunas-Mesa, Teresa Serrano-Gotarredona, Sio Hoi Ieng, Ryad Benjamin Benosman, and Bernabe Linares-Barranco. On the use of orientation filters for 3d reconstruction in event-driven stereo vision. *Frontiers in neuroscience*, 8:48, 2014. 2

[5] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. Event-based convolutional networks for object detection in neuromorphic cameras. *CoRR*, 2018. 3

[6] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *IJCV*, 113(1):54–66, 2015. 3

[7] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. *CVPR*, 2018. 1

[8] Zhuoyuan Chen, Xun Sun, and Liang Wang. A Deep Visual Correspondence Embedding Model for Stereo Matching Costs. *ICCV*, 2015. 1

[9] Georgi Dikov, Mohsen Firouzi, Florian Röhrbein, Jörg Conradt, and Christoph Richter. Spiking cooperative stereo-matching at 2 ms latency with neuromorphic hardware. In *Conference on Biomimetic and Biohybrid Systems*, pages 119–137. Springer, 2017. 3

[10] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *CVPR*, 2015. 1

[11] Mohsen Firouzi and Jörg Conradt. Asynchronous event-based cooperative stereo matching using neuromorphic silicon retinas. *Neural Processing Letters*, 43(2):311–326, 2016. 2

[12] Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. Asynchronous, photometric feature tracking using events and frames. In *ECCV*, 2018. 2

[13] Spyros Gidaris and Nikos Komodakis. Detect, replace, refine: Deep structured prediction for pixel wise labeling. *CVPR*, 2017. 1

[14] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, pages 270–279, 2017. 3

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[16] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *PAMI*, 2008. 1, 6, 7

[17] Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *PAMI*, 2013. 6, 7

[18] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959. 2

[19] Zequn Jie, Pengfei Wang, Yonggen Ling, Bo Zhao, Yunchao Wei, Jiashi Feng, and Wei Liu. Left-right comparative recurrent model for stereo matching. In *CVPR*, 2018. 1

[20] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. *ICCV*, 2017. 1, 3, 4

[21] Patrick Knöbelreiter, Christian Reinbacher, Alexander Shekhovtsov, and Thomas Pock. End-to-end training of hybrid cnn-crf models for stereo. *CVPR*, 2017. 1

[22] Jurgen Kogler, Martin Humenberger, and Christoph Sulzbachner. Event-based stereo matching approaches for frameless address event stereo data. In *International Symposium on Visual Computing*, pages 674–685. Springer, 2011. 2

[23] Zhengfa Liang, Yiliu Feng, Yulan Guo Hengzhu Liu Wei Chen, and Linbo Qiao Li Zhou Jianfeng Zhang. Learning for disparity estimation through feature constancy. *CVPR*, 2018. 1

[24] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *CVPR*, 2016. 1

[25] Ana I Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *CVPR*, 2018. 2, 3, 4

[26] David Marr and Tomaso Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, 1976. 2

[27] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 1

[28] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang. On building an accurate stereo matching system on graphics hardware. In *ICCVW*, 2011. 1

[29] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015. 1

[30] Diederik Paul Moeys, Federico Corradi, Emmett Kerr, Philip Vance, Gautham Das, Daniel Neil, Dermot Kerr, and Tobi Delbrück. Steering a predator robot using a mixed frame/event-driven convolutional neural network. In *EBCCSP*, 2016. 3, 4

[31] Multi vehicle stereo event camera dataset. `https://daniilidis-group.github.io/mvsec/` Accessed: 09 March 2019. 5

[32] Kyoung Won Nam, Jeongyun Park, In Young Kim, and Kwang Gi Kim. Application of stereo-imaging technology to medical field. *Healthcare informatics research*, 2012. 1

[33] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *NIPS*, 2016. 3

[34] Anh Nguyen, Thanh-Toan Do, Darwin G Caldwell, and Nikos G Tsagarakis. Real-time 6dof pose relocalization for event cameras with stacked spatial lstm networks. *CVPR*, 2019. 3, 4

[35] Garrick Orchard, Cedric Meyer, Ralph Etienne-Cummings, Christoph Posch, Nitish Thakor, and Ryad Benosman. Hfirst: a temporal approach to object recognition. *PAMI*, 2015. 3

[36] Jiahao Pang, Wenxiu Sun, JS Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *ICCVW*, 2017. 1

[37] Ewa Piatkowska, Ahmed Belbachir, and Margrit Gelautz. Asynchronous stereo vision for event-driven dynamic stereo sensor using an adaptive cooperative approach. In *ICCVW*, 2013. 2

[38] Ewa Piatkowska, Jurgen Kogler, Nabil Belbachir, and Margrit Gelautz. Improved cooperative stereo matching for dynamic vision sensors with ground truth evaluation. In *CVPRW*, 2017. 2, 6, 7

[39] Andrea Pilzer, Dan Xu, Mihai Puscas, Elisa Ricci, and Nicu Sebe. Unsupervised adversarial depth estimation using cycled generative networks. In *3DV*. IEEE, 2018. 1

[40] Pytorch web site. `http://http://pytorch.org/` Accessed: 08 March 2019. 5

[41] Lorrin A Riggs, Floyd Ratliff, Janet C Cornsweet, and Tom N Cornsweet. The disappearance of steadily fixated visual test objects. *JOSA*, 43(6):495–501, 1953. 2

[42] Paul Rogister, Ryad Benosman, Sio-Hoi Ieng, Patrick Lichtsteiner, and Tobi Delbruck. Asynchronous event-based binocular stereo matching. *IEEE Transactions on Neural Networks and Learning Systems*, 23(2):347–353, 2012. 2

[43] Robotic operation system. `http://www.ros.org/` Accessed: 09 March 2019. 5

[44] Stephan Schraml, Ahmed Nabil Belbachir, and Horst Bischof. Event-driven stereo matching for real-time 3d panoramic vision. In *CVPR*, 2015. 2

[45] Akihito Seki and Marc Pollefeys. Sgm-nets: Semi-global matching with neural networks. *CVPR*, 2017. 1

[46] Amit Shaked and Lior Wolf. Improved stereo matching with constant highway networks and reflective confidence learning. *CVPR*, 2017. 1

[47] David E Shean, Oleg Alexandrov, Zachary M Moratto, Benjamin E Smith, Ian R Joughin, Claire Porter, and Paul Morin. An automated, open-source pipeline for mass production of digital elevation models (DEMs) from very-high-resolution commercial stereo satellite imagery. *ISPRS*, 2016. 1

[48] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: histograms of averaged time surfaces for robust event-based object classification. In *CVPR*, 2018. 2

[49] Bongki Son, Yunjae Suh, Sungho Kim, Heejae Jung, Jun-Seok Kim, Changwoo Shin, Keunju Park, Kyoobin Lee, Jinman Park, Jooyeon Woo, et al. A 640$\times$480 dynamic vision sensor with a 9μm pixel and 300meps address-event representation. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 66–67. IEEE, 2017. 2

[50] Xiao Song, Xu Zhao, Hanwen Hu, and Liangji Fang. Edgestereo: A context integrated residual pyramid network for stereo matching. *CoRR*, 2018. 1

[51] S. Tulyakov, A. Ivanov, and F. Fleuret. Weakly supervised learning of deep metrics for stereo reconstruction. In *ICCV*, 2017. 1

[52] Stepan Tulyakov, Anton Ivanov, and Francois Fleuret. Practical Deep Stereo (PDS): Toward applications-friendly deep stereo matching. In *NeurIPS*, 2018. 1, 4

[53] Lin Wang, Yo-Sung Ho, Kuk-Jin Yoon, et al. Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. *CVPR*, 2019. 3, 4

[54] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *CVPR*, 2018. 5

[55] Ting-Chun Wang, Manohar Srikanth, and Ravi Ramamoorthi. Depth from semi-calibrated stereo and defocus. In *CVPR*, 2016. 1

[56] Zhen Xie, Shengyong Chen, and Garrick Orchard. Event-based stereo depth estimation using belief propagation. *Frontiers in neuroscience*, 11:535, 2017. 2

[57] Zhen Xie, Jianhua Zhang, and Pengfei Wang. Event-based stereo matching using semiglobal matching. *International Journal of Advanced Robotic Systems*, 15(1):1729881417752759, 2018. 2

[58] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, pages 802–810, 2015. 3

[59] Guorun Yang, Hengshuang Zhao, Jianping Shi, Zhidong Deng, and Jiaya Jia. Segstereo: Exploiting semantic information for disparity estimation. In *ECCV*, 2018. 1

[60] Chengxi Ye, Anton Mitrokhin, Chethan Parameshwara, Cornelia Fermüller, James A Yorke, and Yiannis Aloimonos. Unsupervised learning of dense optical flow and depth from sparse event data. *CoRR*, 2018. 3, 4

[61] Jure Žbontar and Yann LeCun. Computing the Stereo Matching Cost With a Convolutional Neural Network. *CVPR*, 2015. 1, 3, 6, 7

[62] Feihu Zhang and Benjamin W Wah. Fundamental principles on learning new features for effective dense matching. *IEEE Transactions on Image Processing*, 27(2):822–836, Feb 2018. 1

[63] Yiran Zhong, Yuchao Dai, and Hongdong Li. Self-supervised learning for stereo matching with self-improving ability. *CoRR*, 2017. 1

[64] Yi Zhou, Guillermo Gallego, Henri Rebecq, Laurent Kneip, Hongdong Li, and Davide Scaramuzza. Semi-dense 3d reconstruction with a stereo event camera. In *ECCV*, 2018. 2, 5, 6, 7, 8

[65] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2018. 5

[66] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Ev-flownet: self-supervised optical flow estimation for event-based cameras. *Robotics: Science and Systems*, 2018. 3, 4, 6

[67] Alex Zihao Zhu, Yibo Chen, and Kostas Daniilidis. Realtime time synchronized event-based stereo. In *ECCV*, 2018. 2, 5, 6, 7, 8

[68] Dongqing Zou, Ping Guo, Qiang Wang, Xiaotao Wang, Guangqi Shao, Feng Shi, Jia Li, and Paul-KJ Park. Context-aware event-driven stereo matching. *ICIP*, 2016. 2

[69] Dongqing Zou, Feng Shi, Weiheng Liu, Jia Li, Qiang Wang, Paul-KJ Park, Chang-Woo Shi, Yohan J Roh, and Hyunsurk Eric Ryu. Robust dense depth map estimation from sparse dvs stereos. In *BMVC*, 2017. 2, 3

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 $t_1, t_2, \dots t_M$ from a normal distribu-

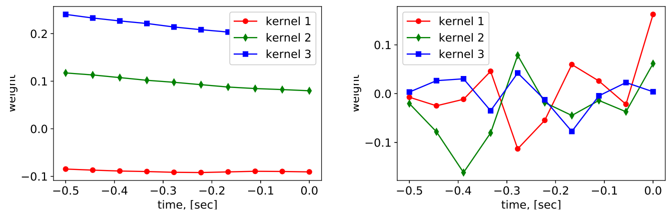
Table 3. Architecture of the proposed stereo matching network. The residual blocks consist of two 2d convolutions followed by shortcut connections. The convolutions and transposed convolutions, including these in the residual blocks, are followed by LeakyReLU with negative slope 0.2 and Instance Normalization (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