Unsupervised Event-Based Video Reconstruction
SUPPLEMENTAL MATERIAL

Gereon Fox
MPI Informatik
Saarland Informatics Campus
gfox@mpi-inf.mpg.de

Xingang Pan
Nanyang Technological University
xingang.pan@ntu.edu.sg

Ayush Tewari
MIT
ayusht@mit.edu

Mohamed Elgharib
MPI Informatik
Saarland Informatics Campus
elgharib@mpi-inf.mpg.de

Christian Theobalt
MPI Informatik
Saarland Informatics Campus
theobalt@mpi-inf.mpg.de

1. Exposure gaps

Long exposure times are more desirable than high framerates for the DAVIS 346C. This is because as one shortens exposure times to obtain higher framerates, the "gap time" in between exposures remains rather constant, as can be observed in Tab. 1 and Fig. 2. The mapping from exposure times to framerates is thus not at all linear (see Fig. 1) and the "exposure coverage", i.e. the percentage of the sequence time during which the shutter is actually open decreases as exposure time is reduced (see Fig. 3). This means that the shorter one makes exposures (e.g. to reduce motion blur), the less information about the course of the sequence can actually be represented by the frames. As a simple example, consider a laser pointer moving nonlinearly on a wall: A long exposure frame representing this motion will be blurry, but it will inform the viewer about the exact trace of the laser point. Two short exposure frames with a considerable exposure gap between them will only give 2 positions of the points and not inform the viewer about the path the point was following between those positions. It is for this reason that we chose to configure the camera with long exposures, as this gives us the maximum of information about the scene.

2. Frame dimensions and color

Our method is not specific to any particular model of event camera, but in our experiments we used the DAVIS 346C. Its pixel matrix, resolution 346×260, is equipped with a Bayer filter, which means that each pixel records brightness in only one of three wavelength ranges, see Fig. 4. In our implementation we account for the Bayer filter by treating each 2×2 square of the pixel matrix as 1 pixel

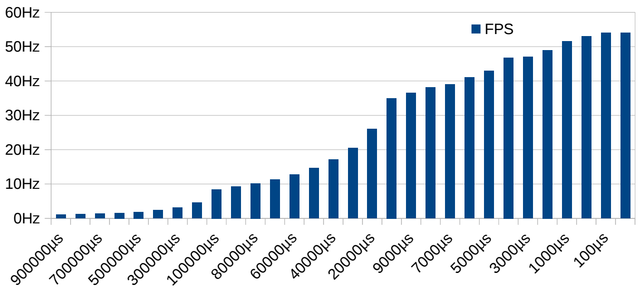

Figure 1. A visualization of the "FPS" column of Tab. 1, mapping exposure times to the framerates that the DAVIS 346C delivers. Even for the shortest exposure times, we cannot reach more than 60FPS.

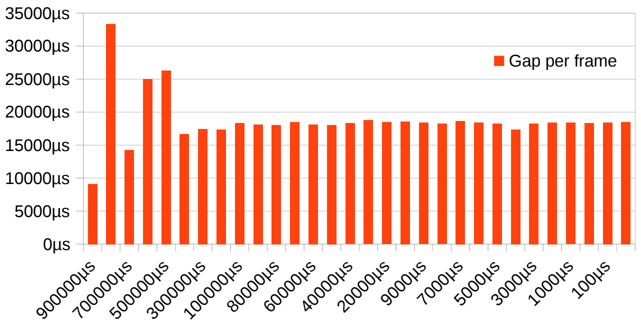

Figure 2. A visualization of the "Gap per frame" column of Tab. 1, mapping exposure times to exposure "gaps". As exposures become shorter and shorter, gap times stay at around $18ms$. This is the reason why shortening exposure time cannot indefinitely increase framerate for the DAVIS 346C.

Exposure	FPS	Gap per frame	Coverage
$900000\mu s$	1.1Hz	$9090\mu s$	99.00%
$800000\mu s$	1.2Hz	$33333\mu s$	96.00%
$700000\mu s$	1.4Hz	$14285\mu s$	98.00%
$600000\mu s$	1.6Hz	$25000\mu s$	96.00%
$500000\mu s$	1.9Hz	$26315\mu s$	95.00%
$400000\mu s$	2.4Hz	$16666\mu s$	96.00%
$300000\mu s$	3.15Hz	$17460\mu s$	94.50%
$200000\mu s$	4.6Hz	$17391\mu s$	92.00%
$100000\mu s$	8.45Hz	$18343\mu s$	84.50%
$90000\mu s$	9.25Hz	$18108\mu s$	83.25%
$80000\mu s$	10.2Hz	$18039\mu s$	81.60%
$70000\mu s$	11.3Hz	$18495\mu s$	79.10%
$60000\mu s$	12.8Hz	$18125\mu s$	76.80%
$50000\mu s$	14.7Hz	$18027\mu s$	73.50%
$40000\mu s$	17.15Hz	$18309\mu s$	68.60%
$30000\mu s$	20.5Hz	$18780\mu s$	61.50%
$20000\mu s$	26.0Hz	$18461\mu s$	52.00%
$10000\mu s$	35.0Hz	$18571\mu s$	35.00%
$9000\mu s$	36.5Hz	$18397\mu s$	32.85%
$8000\mu s$	38.1Hz	$18246\mu s$	30.48%
$7000\mu s$	39.0Hz	$18641\mu s$	27.30%
$6000\mu s$	41.0Hz	$18390\mu s$	24.60%
$5000\mu s$	43.0Hz	$18255\mu s$	21.50%
$4000\mu s$	46.8Hz	$17367\mu s$	18.72%
$3000\mu s$	47.0Hz	$18276\mu s$	14.10%
$2000\mu s$	49.0Hz	$18408\mu s$	9.80%
$1000\mu s$	51.6Hz	$18379\mu s$	5.16%
$500\mu s$	53.0Hz	$18367\mu s$	2.65%
$100\mu s$	54.0Hz	$18418\mu s$	0.54%
$10\mu s$	54.0Hz	$18508\mu s$	0.05%

Table 1. We configured the DAVIS 346C with multiple different exposure times and then measured the resulting framerates that the camera was able to deliver. From these measurements we can compute how much "gap" time goes by between exposures. We also computed exposure "coverage", i.e. the percentage of sequence time during which the shutter is actually open.

of depth 4 (red, green, green, blue). The brightness frames therefore have the dimensions $w \times h \times d = 173 \times 130 \times 4$. All the results we show in this work have been demosaiced in order to obtain RGB images.

3. Computation of control point gradients

In Sec 3.2 of the main paper, we stated:

> Given $\bar{b}$, chaining Eq. (2) in the form $p_{j+1} \cdot (\tilde{b}^*(t_{j+1}) - \tilde{b}^*(t_j)) = c_j$ admits only one possible valuation for those g_k that are *event*-based , if one assumes that brightness is constant between events.

We now give a detailed description of how to compute this

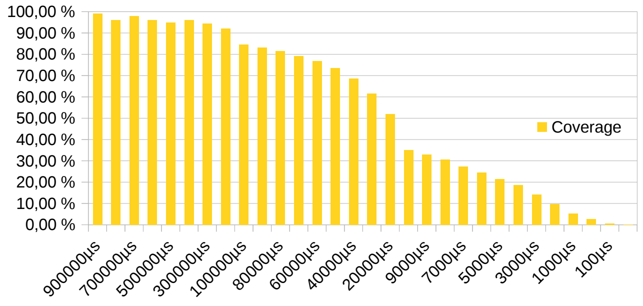

Figure 3. A visualization of the "Coverage" column of Tab. 1, mapping exposure times to the percentage of sequence time during which the shutter is open. Shorter exposure times, while somewhat increasing framerate (see Fig. 1), lead to less coverage and therefor less information being captured.

valuation, which basically means that we compute the exact geometric shape of the orange-hatched area in Fig. (2) of the main paper:

Assuming that for a given pixel the camera reported exactly $n-2$ events of the form (t_j, p_j), with $j \in \{1, ..., n-2\}$, we add the artificial events $(T_0, \Omega_0), (T_1, \Omega_1)$ (where Ω_0, Ω_1 denote undefined polarities), such that there are n events in total. The artificial events are added only to simplify the mathematical formulation and their undefined polarities will never need to be used in any actual calculation.

By chained application of the aforementioned equation to the effective thresholds c_j, for all events from the first one (artificial, at time T_0) up to event j, we can compute the factor ${f_j}^+$ with which an initial brightness value must be multiplied in order to obtain the correct brightness value after event j:

$$ {f_j}^+ := \exp\left(\sum_{l=1}^{j} p_l \cdot c_l\right) \qquad (1) $$

Likewise, we can compute the factor ${f_j}^-$, by which a *final* brightness value must be multiplied in order to obtain the correct brightness value after event j:

$$ {f_j}^- := \exp\left(\sum_{l=j}^{n-2} -p_l \cdot c_l\right) \qquad (2) $$

Note that, because of the definition of $\sum$, Eqs. (1) and (2) entail ${f_0}^+ = 1 = {f^-}_{n-1}$, although p_0 and p_{n-1} are undefined.

Based on our average brightness parameter $\bar{b}$ we can now define brightness values (*i.e.*, gradients) that M should have between events j and $j+1$:

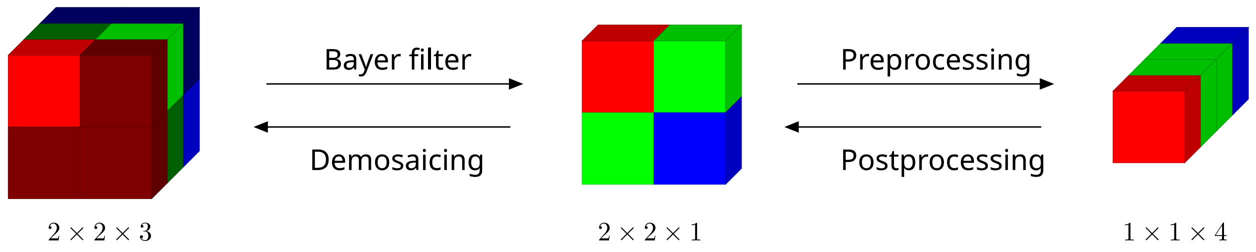

Figure 4. The DAVIS 346C records brightness in three wavelength ranges: red, green and blue. However, because of the Bayer filter in the camera, each pixel sees only one of these ranges, with 25% of the pixels seeing red, 25% of the pixels seeing blue and 50% of pixels seeing green, leading to a $2 \times 2 \times 1$ "Bayer mosaic". We implemented this Bayer filtering for our synthetic data as well. For any footage, synthetic or real, our preprocessing for all methods, ours and previous, turns the Bayer mosaic into a $1 \times 1 \times 4$ pattern, which we undo once we obtained the output from each method. This preprocessing is necessary to make sure that all the pixels in a channel belong to the same wavelength range, as otherwise a red apple for example would lead to a distinct spatial grid pattern in the mosaic, that would throw off convolutional neural networks. Demosaicing is the process of interpolating missing RGB-values from the surrounding pixels, which is a standard operation in frameworks such as openCV and was used in previous works [3] as well. Demosaicing causes interpolation artefacts.

$$g_j := \frac{1}{2} \frac{\bar{b} \cdot \Delta T}{\sum_{l=0}^{n-2} {f_l}^+ \cdot \Delta t_l} \cdot {f_j}^+ + \frac{1}{2} \frac{\bar{b} \cdot \Delta T}{\sum_{l=0}^{n-2} {f_l}^- \cdot \Delta t_l} \cdot {f_j}^- \tag{3}$$

where $\Delta t_j := t_{j+1} - t_j$ and $\Delta T := T_1 - T_0$. These are the exact values of b^* at the red and blue points of Fig. (2) in the main paper, which completely determines the shape of the orange-hatched region and makes its size exactly $\Delta T \cdot \bar{b}$.

The two summands in Eq. (3) do not differ from one another, the equation could be simplified. However, we found that computing the g_j only based on one direction (*i.e.* either based only on Eq. (1) or only based on Eq. (2)) leads to an asymmetric flow of gradients during gradient descent: For example, if only ${f_l}^+$ were used, gradients flowing into g_j could only back-propagate into c_l with $l \leq j$. Experiments have shown that this leads to approximation errors accumulating over the course of the sequence. Instead, we want gradients flowing through g_j to affect *all* c_l.

Furthermore, Eqs. (1) to (3) are prone to numerical problems due to possible division by values near 0 and because of $\exp$ values exceeding the datatype range. Our implementation thus computes the equations in a slightly different way than we present them here, in order to avoid extreme intermediate values and undefined gradients.

For each control point P_k that is event-based and thus represents exactly one event j (where $j = 0$ represents the first, artificial event at T_0), we can now set $g_k := g_j$. As defined in the main paper, the remaining, exposure-based g_k are defined by interpolation between the event-based ones.

4. Bézier construction

At the end of section 3.2 in the main paper we stated that our method uses the control point parameters $w_k^{\text{right}}, w_{k+1}^{\text{left}}$ to define a Bézier curve between the control points P_k, P_{k+1}. This definition is as follows:

By Eq. (5) in the main paper, we are required to find a smooth curve between two given points,(t_k, y_k) and (t_{k+1}, y_{k+1}), with the additional constraint of the derivative of this curve with respect to the t-axis having specific values at times t_k, t_{k+1}. Our Bézier curve must therefor be at least a cubic one, which means that we need two additional helper points in-between (t_k, y_k) and (t_{k+1}, y_{k+1}). We thus set $d_\text{A} := w_k^{\text{right}} \cdot (t_{k+1} - t_k)$ and $d_\text{B} := w_{k+1}^{\text{left}} \cdot (t_{k+1} - t_k)$ and define the helper points

$$\begin{aligned} P_{k,k+1}^\text{A} &:= P_k + \begin{pmatrix} d_\text{A} \\ g_k d_\text{A} \end{pmatrix} \\ P_{k,k+1}^\text{B} &:= P_{k+1} - \begin{pmatrix} d_\text{B} \\ g_{k+1} d_\text{B} \end{pmatrix} \end{aligned} \tag{4}$$

The cubic Bézier curve for the points $P_k, P_{k,k+1}^\text{A}, P_{k,k+1}^\text{B} P_{k+1}$ completely determines M between P_k and P_{k+1}. The control point parameters w_k^{right} and w_{k+1}^{left} control how quickly the gradient of M transitions from g_k to g_{k+1}.

5. Implementation details

DAVIS 346C settings. We used the DAVIS 346C pretty much with the default parameters as set by the IDE from the camera manufacturer[1]. We enabled FPGA filtering. Some of our sequences were recorded with the background activity filter, while others were not. While the rates of events differ noticeably depending on the usage of the DVS background activity filter, we did not notice any visual differences in our results.

Optimization. We implemented our method in PyTorch. For optimization, we use the Adam optimizer [1] for 1000

[1]https://inivation.gitlab.io/dv/dv-docs/

	Reference exposure 0.1s			Reference exposure 0.002s		
Variant	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Linear interpolation	40.72dB	0.9866	0.0112	29.50dB	0.9006	**0.0624**
Parabolic interpolation	39.21dB	0.9795	0.0174	28.95dB	0.8889	0.0793
No confidences	43.96dB	0.9933	0.0047	**30.25dB**	**0.9105**	0.0647
No exposure-based CP	43.69dB	0.9929	0.0052	29.52dB	0.9007	0.0755
Without $\mathcal{L}_{\text{confidence}}$	**45.80dB**	**0.9957**	**0.0032**	27.70dB	0.8677	0.1153
Without $\mathcal{L}_{\text{linearity}}$	42.61dB	0.9915	0.0096	20.92dB	0.6678	0.2613
Ours (full)	45.17dB	0.9949	0.0035	29.69dB	0.9039	0.0739

Table 2. An extended version of Tab. (3) in our main paper, with more decimal digits and additional columns for LPIPS scores.

iterations. To define the learning rate r_i for iteration i we first set $\alpha_i := (1 - \frac{i-100}{900})^{1.5}$ and then specify:

$$r_i := \begin{cases} 10^{-2} & : i = 0 \\ \frac{i}{100} \cdot r_0 & : 0 < i < 100 \\ (r_0 - r_{999}) \cdot \alpha_i + r_{999} & : 100 \leq i < 999 \\ 10^{-3} & : i = 999 \end{cases} \tag{5}$$

The tensors representing our model parameters are usually in the range [-1; 1] (even though the range of the mathematical variable they represent may be a different one!), but some of them, like the confidence weights for example, have a far smaller range, which effectively increases their learning rate. These details are best studied directly in our source code, which we plan to release.

Initialization and invariants. Our model parameters are initialized and bounded as follows:

1. We always initialize the thresholds as $c_{+1} = c_{-1} = 0.15$ and keep them in the interval $[0.01; 1.8]$.
2. For each pixel, $\bar{b}$ can easily be initialized to be the average brightness level of this pixel over all input brightness frames. We only bound $\bar{b}$ to be positive, because although the range of the brightness values in the recorded frames is limited, there is physically no maximal brightness value we could rely on.
3. The y_k and δ_k are initialized such that the orange-hatched area in Fig. (2) in the main paper satisfies $\mathcal{L}_{\text{exposure}}$ reasonably well. To do so, we have to take exposure gaps into account. The details of the initialization are best studied in our source code. Since the total amount of physical energy received by a pixel up to some time t is monotonically increasing, the function M should be monotonically increasing as well, requiring $\forall k : y_k \leq y_{k+1}$. The δ_k are kept in the range $[-1; 1]$.
4. The confidence weights γ_j are initialized with 1 and not bounded at all, because the $\mathrm{sigmoid}$ term in Eq. (6) of the main paper properly bounds the confidences we compute based on these weights. For the same reason, ω_{+1}, ω_{-1} (initialized to 1) and β_{+1}, β_{-1} (initialized to 0) are left unbounded.
5. The gradient weights $w_k^{\text{left}}, w_k^{\text{right}}$ are initialized to 0.45 and restricted to the range $[0.05; 0.45]$, to avoid numerical issues that might arise in the computation of Bézier interpolation.

Performance. We load all input data (frames and events) into the GPU once and thus need no overhead for data loading processes or data sampling. Every forward and backward pass for optimization is touching all the weights of our model and is supervised by the entirety of the input data. Computational performance varies greatly with the number of events and their distribution. Our recordings are typically 5 - 15 seconds long. None of them requires more than 48GB of GPU memory, most of them require less than 24GB. Some sequences are processed within tens of minutes, others might take 2 - 4 hours.

6. LPIPs scores for the quantitative ablation

In Tab. (3) of the main paper we gave PSNR and SSIM scores for our ablation study on synthetic data. To save space, we omitted the LPIPS scores for this experiment and rounded the numbers to a shorter decimal representation. Tab. 2 is an extended version of the table in the main paper, for completeness.

7. TimeLens is out of scope

Comparing our method to TimeLens(++) [4,5] would be unfair to TimeLens. This is because we are investigating the setting of long exposure RGB frames, which contain a lot of motion blur. However, TimeLens is designed for blur-free input images, recorded with a short exposure time. In addition, TimeLens expects the input to contain RGB brightness, but grayscale events. The DAVIS 346C cannot provide such input, because the events it records are inherently colored and there is no way of computing grayscale events from them, other than applying a method such as the one presented in this work.

To demonstrate that TimeLens is not suitable for our setting, we nevertheless applied it to some of our recordings and synthetic data. We did so by treating each quadrant of the Bayer mosaic (see Sec. 2 and Fig. 4) as its own separate sequence, i.e. by turning, for each of the 4 quadrants, the brightness pixels of this quadrant into one contiguous grayscale image and using only the events for this quadrant. This is still unfair, because TimeLens was trained for colored brightness frames and not for artificial grayscale frames. Fig. 5 shows that TimeLens is unable to compensate the motion blur present in the long exposure input. Quantitative evaluation on our 10Hz synthetic data (see Table 1 in our main manuscript) gave a PSNR score of only 22.51dB, showing once more that TimeLens is the wrong tool for our setting.

8. Limitations extended

The design choices of our method make it avoid many of the limitations of existing methods. For example, since we exploit the event semantics in a principled way, we do not need any pre-training and thus no training data, from which we could inherit a bias. As a second example, the fact that we treat all pixels rather independently from each other makes it easy for us to handle complex lighting interactions (like transparency in glass and water) non-linear motion and disocclusions, which are more challenging for methods that use optical flow for example.

However, our design choices of course give rise to limitations as well:

The importance of $\mathcal{L}_{\text{exposure}}$ makes it strictly necessary that we have exact time stamps for the brightness frame exposures, i.e. we must know the times at which the shutter opens and closes. Some existing datasets, such as te Color Event Camera Dataset [3] do not provide this information. In addition, methods like TimeLens [4] typically work on input frames that were recorded with minimal exposure time, which, if used as input to our method, is likely to give noisy results because short exposure times reduce the impact of $\mathcal{L}_{\text{exposure}}$, allowing noise in the event data to become more visible in our output.

Furthermore, an important limitation of our method is its generous usage of GPU RAM: Representing the entire sequence in memory requires significant GPU capacity. We experimented with applying our method only to pairs of consecutive brightness frames and stitching the results together. While we found this to lead to qualitatively comparable results with less memory demand, it does take more computation time, because frame pairs need to overlap (i.e. every frame is treated twice) and because optimizing for one frame pair simply does not fully use the parallelism offered by the GPU. We chose to value computation time higher than memory consumption and thus reported results from global optimization only.

As future work we consider an investigation into how in particular our confidence regularizer affects the ability of our method to estimate the contrast thresholds used by the event camera. Not many methods are able to do so (one exception being mEDI [2]), but knowledge about the thresholds for a particular sequence is often a valuable piece of information for further processing.

References

[1] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 3

[2] Liyuan Pan, Richard Hartley, Cedric Scheerlinck, Miaomiao Liu, Xin Yu, and Yuchao Dai. High frame rate video reconstruction based on an event camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. 5

[3] Cedric Scheerlinck, Henri Rebecq, Timo Stoffregen, Nick Barnes, Robert Mahony, and Davide Scaramuzza. Ced: Color event camera dataset. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1684–1693. IEEE, June 2019. 3, 5

[4] Stepan Tulyakov, Alfredo Bochicchio, Daniel Gehrig, Stamatios Georgoulis, Yuanyou Li, and Davide Scaramuzza. Time lens++: Event-based frame interpolation with parametric non-linear flow and multi-scale fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17755–17764, 2022. 4, 5, 6

[5] Stepan Tulyakov, Daniel Gehrig, Stamatios Georgoulis, Julius Erbach, Mathias Gehrig, Yuanyou Li, and Davide Scaramuzza. TimeLens: Event-based video frame interpolation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 4, 6

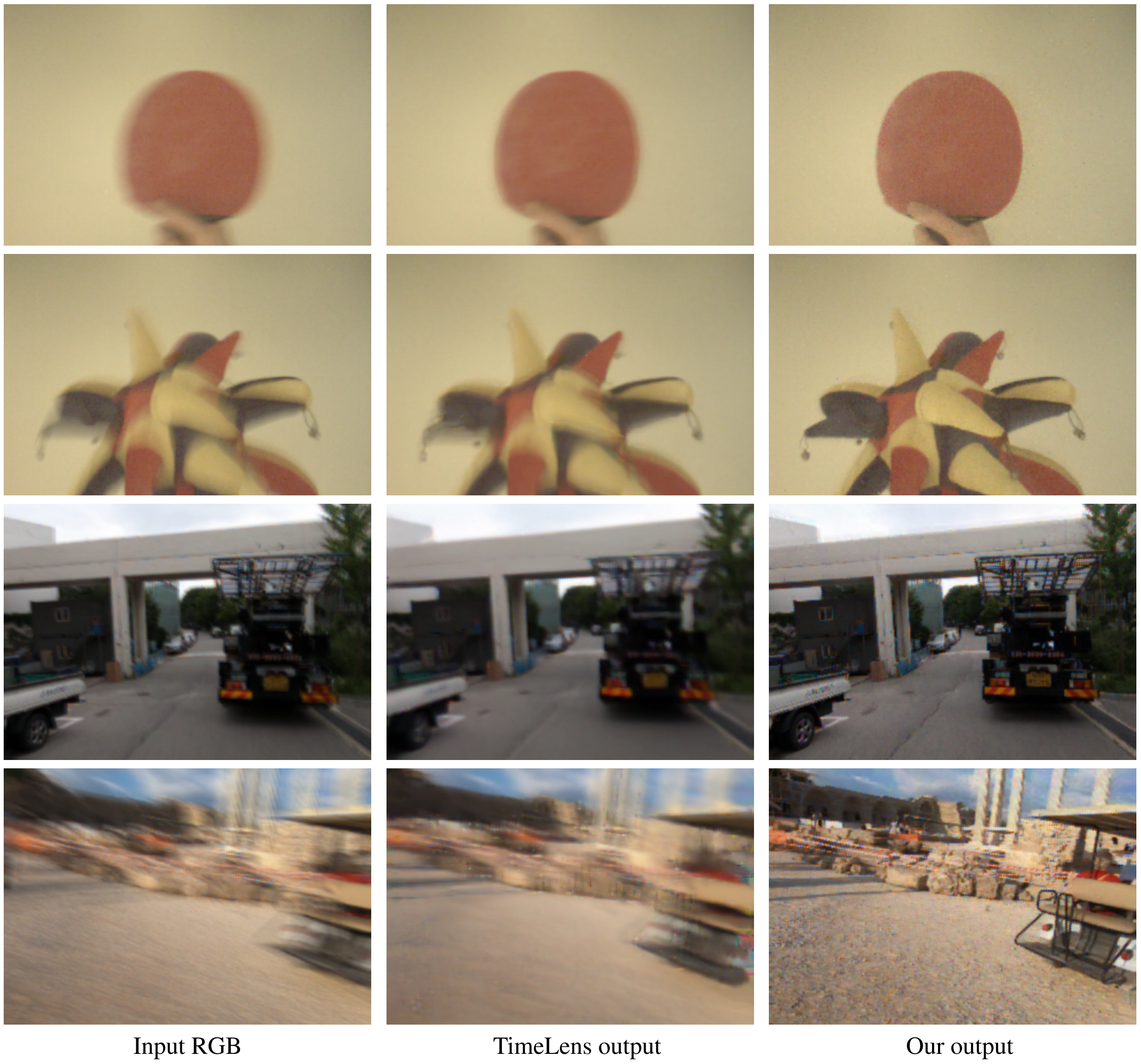

Figure 5. A comparison of our results with those of TimeLens [4, 5] , at output exposure time 0.002s. The top two rows show results on recordings (see Figure 3 in the main manuscript), while the bottom two rows show results on our synthetic dataset. Since TimeLens was neither designed nor trained for long exposure RGB brightness frames and colored events, it fails to resolve the motion blur present in the input. We therefor argue that it would be unfair to include TimeLens in a comparison to our method.