Time Lens++: Event-based Frame Interpolation with Parametric Non-linear Flow and Multi-scale Fusion

Stepan Tulyakov[1]   Alfredo Bochicchio[1]   Daniel Gehrig[2]   Stamatios Georgoulis[1]
Yuanyou Li[1]   Davide Scaramuzza[2]

[1] Huawei Technologies, Zurich Research Center
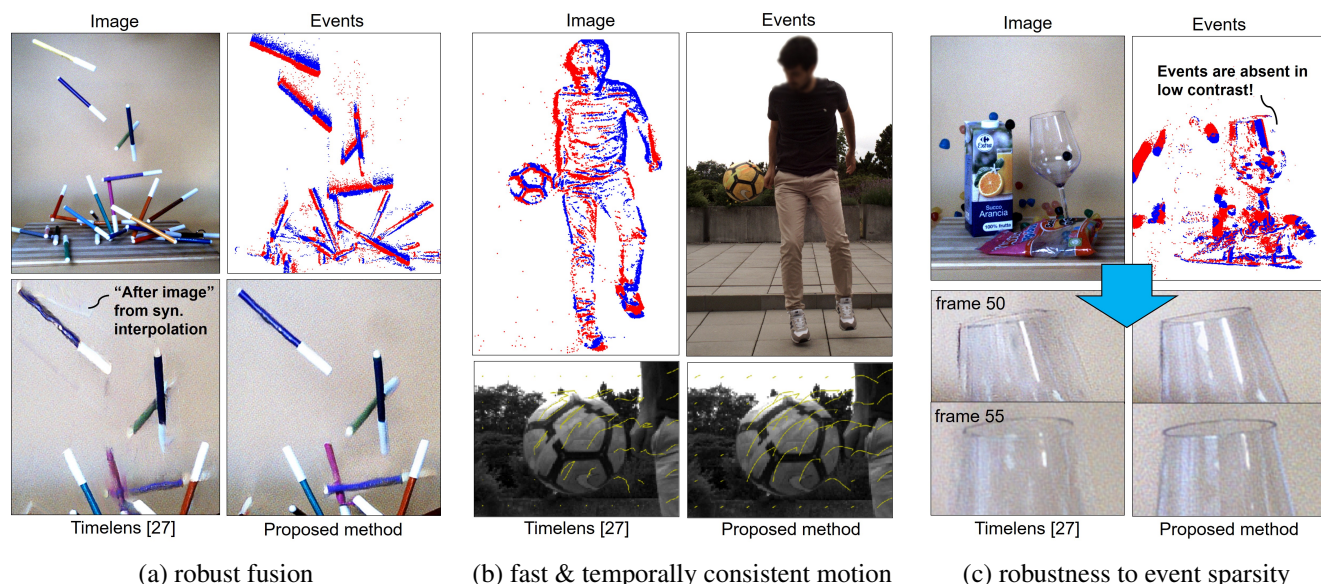[2] Dept. of Informatics, Univ. of Zurich and Dept. of Neuroinformatics, Univ. of Zurich and ETH Zurich

Figure 1. Comparison to state-of-the-art event- and image-based video interpolation method Time Lens [24]. Our method makes a series of key innovations to address the limitations of current approaches. First, it uses feature-level multi-scale fusion which is robust to artifacts in the fused images (a). Second, it computes continuous flow, parametrized by splines, which have inherent temporal consistency (b, bottom right vs. left) and can be efficiently sampled, thereby significantly reducing computation for multi-frame interpolation (b). Finally, it combines images and events to generate flow, even where few events are triggered, thereby mitigating artifacts as in (c).

Abstract

Recently, video frame interpolation using a combination of frame- and event-based cameras has surpassed traditional image-based methods both in terms of performance and memory efficiency. However, current methods still suffer from (i) brittle image-level fusion of complementary interpolation results, that fails in the presence of artifacts in the fused image, (ii) potentially temporally inconsistent and inefficient motion estimation procedures, that run for every inserted frame and (iii) low contrast regions that do not trigger events, and thus cause events-only motion estimation to generate artifacts. Moreover, previous methods were only tested on datasets consisting of planar and faraway scenes, which do not capture the full complexity of the real world. In this work, we address the above problems by introducing multi-scale feature-level fusion and computing one-shot non-linear inter-frame motion—which can be efficiently sampled for image warping—from events and images. We also collect the first large-scale events and frames dataset consisting of more than 100 challenging scenes with depth variations, captured with a new experimental setup based on a beamsplitter. We show that our method improves the reconstruction quality by up to 0.2 dB in terms of PSNR and up to 15% in LPIPS score.

Multimedia Material

For videos, datasets and more visit `https://uzh-rpg.github.io/timelens-pp/`.

1. Introduction

High-speed videography has captured the imagination of the public by producing stunning slow-motion footage of high-speed phenomena [1, 11, 24]. While historically this was only possible with specialized and expensive equipment, today this technology is coming to our smartphones thanks to high-speed cameras and *video frame-interpolation (VFI)* techniques.

VFI techniques generate high frame rate video by inserting intermediate frames between consecutive frames of a low framerate input video. To this end, they estimate image changes in the blind time between consecutive frames, a task that remains challenging, especially in the presence of large displacements and non-linear motion. Most existing VFI methods rely on the information contained in the original video to estimate these changes. However, at low frame rates, image-based motion estimation does not accurately capture inter-frame motion, especially in presence of large and non-linear motion. While using high frame-rate cameras can alleviate this problem, they are typically expensive and produce excessive amounts of data, which cannot be recorded for an extended amount of time. For example, the Huawei P40 can record only 0.5s of 720p video with 1920 fps, which fills up its 2GB frame buffer.

Event cameras address both of these problems. Instead of measuring synchronous frames of absolute brightness like standard cameras, they only measure asynchronous brightness changes at each pixel, which results in a sparse and asynchronous stream of *events*, each encoding the position, time, and polarity (sign) of the measured change. This stream of events is simultaneously sparse, provides a low data bandwidth, and has a high temporal resolution on the order of microseconds capable of capturing high-speed phenomena, such as gunshots and popping water balloons[1].

The recently introduced Time Lens [24] leverages this compressed stream of visual information by combining a traditional camera with an event camera to perform video frame interpolation. Its key innovation lies in combining the benefits of *warping-based* and *synthesis-based* interpolation approaches through an attention mechanism. Warping-based interpolation produces intermediate frames by warping frames of the original video using nonlinear motion estimated from events, while synthesis-based interpolation produces intermediate frames by "adding" intensity changes captured by inter-frame events to the frames of the original video. Time Lens combines these two approaches because they are complementary: while warping-based interpolation usually produces high-quality results, it suffers where motion estimation is unreliable due to violation of brightness constancy assumption. By contrast, the synthesis-based interpolation does not rely on brightness constancy and can

easily handle objects with illumination changes such as, for example, fire, and water, however, it distorts fine texture due to the sparsity of events.

Motivation. Despite its impressive performance compared with pure image-based methods,[2] previous work suffers from several drawbacks.First, to combine warping- and synthesis-based interpolation it relies on image-level fusion which can fail in the presence of artifact in one of the inputs, as shown in Figure 1(a), where "after-image" artifacts from synthesis interpolation are propagated to the final results. Second, it relies on non-parametric motion estimation, that runs independently for each inserted frame with a computational cost of $O(N)$, where N is the number of inserted frames and produces potentially temporally inconsistent motion estimates. Third, to leverage information about non-linear motion it relies on events-only motion estimation, which leads to artifacts in low contrast areas without events (see Figure 1(c)).

This work addresses all of these open challenges. Our method makes a key innovation in terms of motion estimation, visualized in Fig. 2. Instead of using linear (b) or chunked linear flow from events (c), we use both images and intermediate events (a) to predict a continuous flow field (d). By doing so our flow method is inherently temporally consistent and can be efficiently reused for multi-frame insertion. Additionally, we introduce a novel multi-scale fusion module that fuses event and image features on feature-level instead of image-level, thereby limiting ghosting artifacts.

We make the following contributions in this work:

1. We introduce a novel *motion spline estimator*, which produces non-linear continuous flow from events and frames. It is temporally consistent and can be efficiently sampled, enabling the interpolation of N intermediate frames with $O(1)$ instead of $O(N)$ computation. Moreover, leveraging images also produces accurate flow in the absence of events.

2. We introduce a *multi-scale feature fusion* module with multiple encoders and joint decoder with a *gated compression* mechanism that selects the most informative features from each encoder at each scale and improves fusion of warping- and synthesis-based interpolation results.

3. We compare our approach on an existing dataset and a new *large scale hybrid dataset* containing 123 videos collected with a beamsplitter setup that has temporally synchronized and aligned events and frames. We compare our method on multiple benchmarks including this new dataset and found an up to 0.2 dB improvement in terms of PSNR and up to 15% improvement in perceptual score [32] over the prior art, across datasets.

---

[1] https://youtu.be/eomALySSGVU

[2] https://youtu.be/dVLyia-ezvo

2. Related Work

Frame-based video interpolation is a well studied topic with an abundance of prior work [10, 16–21, 29]. It aims at reconstructing intermediate latent frames at arbitrary or fixed timestamps using consecutive frames of the original video, called keyframes. Most image-based frame interpolation methods adopt one of four approaches: While *direct* approaches [11], regress intermediate frames directly from keyframes, *kernel-based* approach [18, 19], apply convolutional kernels to keyframes to produce the latent frame, and *phase-based* [15] approaches, estimate the phase decomposition of the latent frame. The most popular approach is *warping-based* [10, 16, 17, 20, 21, 29] and it explicitly estimates the motion between keyframes and then warps and fuses keyframes to produce the latent frame. This fusion is usually done on the image level, using visibility masks [10] or, more recently on the feature-level [1, 17]. The majority of works tackling VFI are image-based and thus suffer from two major limitations. First, they rely on image-based motion estimation, which is only well defined when brightness constancy is satisfied. Secondly, they can not capture precise motion dynamics in the blind time between keyframes and often resort to a simplistic linear motion assumption.

Motion Estimation: Indeed, motion estimation, has predominantly been studied in the linear case, where correspondences between pixels are assumed to follow linear trajectories. However, in the case of rotational camera ego-motion and non-rigid object motion, this assumption is usually violated. Only few works use more complex motion models, such as quadratic [28] or cubic [2, 3]. While [2, 28] directly regress polynomial coefficients, [3] parametrize the pixel trajectories in terms of 2-D knots, using B-Splines. Fitting these non-linear models requires multiple frames and long time windows, and thus they still fail to model high-speed and non-linear motion between keyframes.

Use of Additional Sensors: To capture this motion, information from additional sensors with high temporal resolution can be used. In particular [7, 20] uses an auxiliary low resolution, high-speed camera to provide these additional cues, and combine them with high-resolution images. However, additional high-frame-rate image sensors increase data rate requirements. This is a fundamental limitation of frame cameras to capture high-speed motions, since they oversample the image, leading to wasteful data acquisition.

Event Cameras: are sensors that ideally address this limitation since they mitigate this oversampling by only providing data at locations with intensity changes, and do this for each pixel independently. The works in [8, 14, 24, 27, 31] have used an auxiliary event camera for VFI, demonstrating high accuracy and low bandwidth. Time Lens [24], generates frame interpolations from a warping-based and synthesis-based module and fuses them on the image-level using learned alpha blending parameters. By combining the

advantages of both, it can handle both regions with brightness constancy and those with illumination changes where optical flow is ill-defined. [31] improves the fusion part by performing progressive multi-scale, feature-level fusion.

However, [31] aligns keyframes to the latent frame using flow, computed from original keyframes, and thus can not capture non-linear inter-frame dynamic. Instead, [24] computes flow from events, capturing non-linear inter-frame dynamic, and directly predicts a series of non-parametric linear flow between keyframes and the latent frame. However, this model does not take into account the continuous nature of events, and since flow is non-parametric it cannot be reused and must be recomputed. While this leads to a significant run-time increase, it also leads to temporal inconsistency, which manifests in wobbling textures. Finally, since flow is computed from events, it is sparse and inaccurate in low contrast regions that do not trigger events.

In this work, we combine the advantages of [24] and [31] while addressing their limitations. Firstly, we make a key innovation in terms of motion estimation, visualized in Fig. 2. Instead of using linear (b) or chunked linear flow from events (c), we use both images and intermediate events (a) to predict a continuous flow field (d), based on cubic splines, similar to [3]. See Fig. 2 for a visualization of their differences. The resulting flow has several advantages: *(i)* it is non-linear, capturing high-speed and highly non-linear dynamics with microsecond resolution, *(ii)* it is dense, thus producing flow, even where few events are present and *(iii)* it can be efficiently reused during multi-frame insertion, resulting in low inference time and high temporal consistency.

3. Method

Problem formulation. We are given as input proceeding I_0 and following I_1 key frames acquired at time 0 and 1, and *event sequence* $E_{0 \to 1}$ consisting of events triggered during the time interval $t \in [0, 1]$, and our goal is to insert one or more *latent frame(s)* $\hat{I}_t$ at some time $t \in [0, 1]$ between the key frames. Similar to previous works, we represent events as a *voxel grid* [34]. We will use $V_{a \to b}$ to denote the voxel grid formed by converting events between times t_a and t_b.

System overview. The overall system is shown in Figure 3 and key contributions are highlighted with a thick contour. Our system first generates multi-scale *warping interpolation features* in two steps. First image I_0 is encoded using a warping encoder, resulting in multi-scale features. These features are then remapped to the time of the latent frame using splines $S_{0 \to 1}$, derived from the voxel grid $V_{0 \to 1}$ and images I_0, I_1, resulting in $\{\mathbf{C}_{0 \to t}^w, \mathbf{C}_{1 \to t}^w\}$. The *warping interpolation features* are combined with *synthesis interpolation features* $\{\mathbf{C}_{0 \to t}^s, \mathbf{C}_{1 \to t}^s\}$ computed from I_0 and I_1, using a newly proposed *multi-scale feature fusion* module and produces latent frame $\hat{I}_t$ at time t. The system performs symmetric processing for I_0 and I_1, therefore we
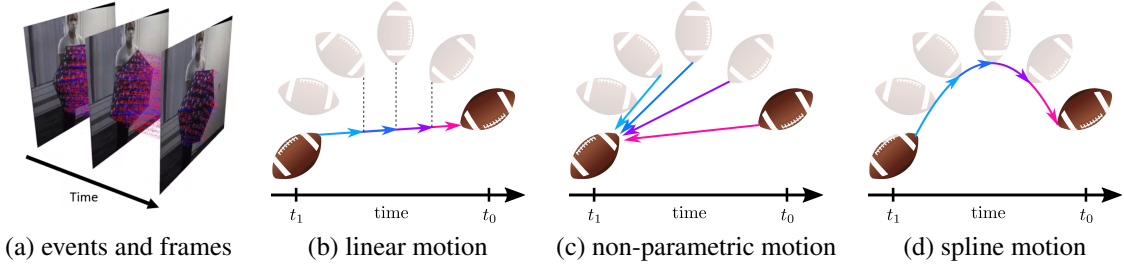
Figure 2. Different motion estimation types for video-frame interpolation (VFI). More details are in Section 2, "motion estimation".
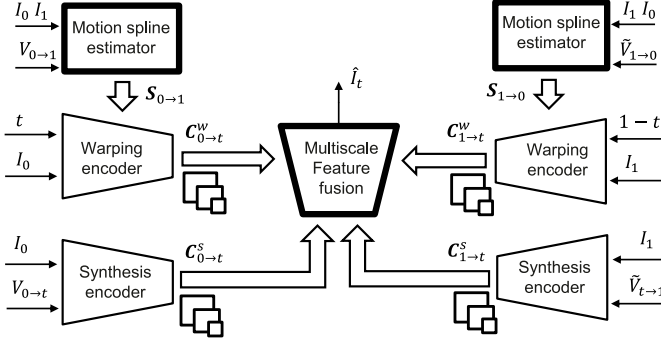

Figure 3. System overview. Main contribution of this work, shown with a thick contour, are: *multi-scale feature fusion* that combines synthesis and warping-based interpolations on a feature level on multiple scales and *motion spline estimation* that computes parametric motion model from boundary frames and inter-frame events. See "system overview" in Section 3 for more details.

only explain processing for I_0.

The *synthesis interpolation features* $\mathbf{C}_{0 \rightarrow t}^{s}$ are computed by a *synthesis encoder* from the preceding keyframe I_0 and voxel grid $V_{0 \rightarrow t}$ for events between preceding and latent frames. Intuitively, this encoder "adds" intensity changes registered by events to the keyframe and thus can interpolate non-rigid objects with illumination changes, such as fire and water. In contrast to prior work, we found it beneficial to encode I_0 and I_1 separately using an encoder with shared weights, which we call *shared synthesis encoder*.

The *warping interpolation features* $\mathbf{C}_{0 \rightarrow t}^{w}$ are computed by a *warping encoder* which warps features extracted from the preceding key frame I_0 to time t using pixel-wise *spline model* of inter-frame motion $\mathbf{S}_{0 \rightarrow 1} = \{S_{0 \rightarrow t}^{\Delta x}, S_{0 \rightarrow t}^{\Delta y}, S_{0 \rightarrow t}^{p}\}$.

The motion splines $\mathbf{S}_{0 \rightarrow 1}$ are computed once per interframe interval using boundary frames $\{I_0, I_1\}$ and voxel grid $V_{0 \rightarrow 1}$ for inter-frame events using a newly introduced *motion spline estimator*. Note, that in contrast to previous works, our motion estimator in addition to events uses boundary images to ensure interpolation robustness in low-contrast areas without events. In the next paragraphs, we explain the key modules of our system in detail.

Multi-scale feature fusion. To improve fusion of synthesis and warping interpolation result, we use a *multi-scale*

feature fusion decoder shown in Figure 4. The decoder progressively combines warping and synthesis interpolation features and features from the previous processing stage performed on a coarser scale. It relies on a novel *gated compression* module, which attenuates features before combining them and thus, intuitively selects informative features from each source. For example, in Fig. 7, the gated compression decides to use synthesis-based interpolation for the flame, for which optical flow fails (colder colors in image 1, and warmer in image 2 of the figure). The gating idea was inspired by recent work on HDR fusion [30], where it was used for combining multiple exposures.

In [31], the authors also use a network with multiple encoders for combining images and events, however in contrast to their approach we *(i)* estimate motion not only from keyframes but also from events; *(ii)* move features of keyframes instead of keyframes; *(iii)* use gating instead of location-wise alpha blending for selecting most informative source; *(iv)* use not all inter-frame events in the synthesis encoder, but only events between keyframe and latent frame and *(v)* combine features from the previous coarse-scale processing stage with other features. In the supplementary materials and ablations studies, we show the benefits of these design choices.

Motion spline estimator. Previous work [24] computes motion independently from each latent frame to boundary frame by re-partitioning the events as shown in Figure 2(c). This approach leverages information about non-linear interframe motion contained in events, however, its computational complexity scales linearly $O(N)$ with the number of interpolated frames N and its motion estimates are independent and, thus, potentially temporally inconsistent.

In contrast, this work relies on a *spline motion estimator* shown in Figure 5 which infers nonlinear interframe motion (see Figure 2(d)) from events and key frames and approximates it with splines. This enables efficient sampling of motion between key frames and arbitrary latent frame and ensures temporal consistency of the motion. Our motion estimator computes 3 cubic splines $\{S_{0 \rightarrow 1}^{\Delta x}, S_{0 \rightarrow 1}^{\Delta y}, S_{0 \rightarrow 1}^{p}\}$ for each location. These splines model horizontal, vertical displacement and warp-
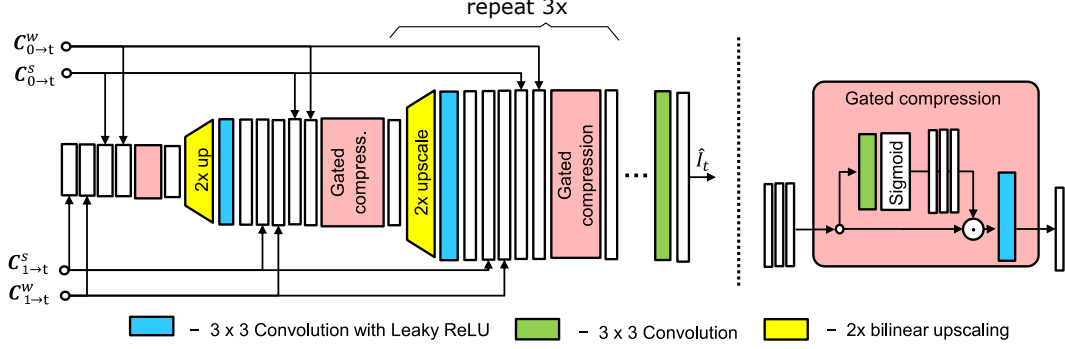
Figure 4. *Multi-scale fusion* decoder progressively combines warping and synthesis interpolation features and features from the previous processing stage performed on a coarser scale using novel *gated compression* module, that selects most informative features from each source. For detailed explanation please refer "multi-scale feature fusion" minisection in Section 3. Blank white boxes represents feature maps. $\odot$ represents the element-wise product.
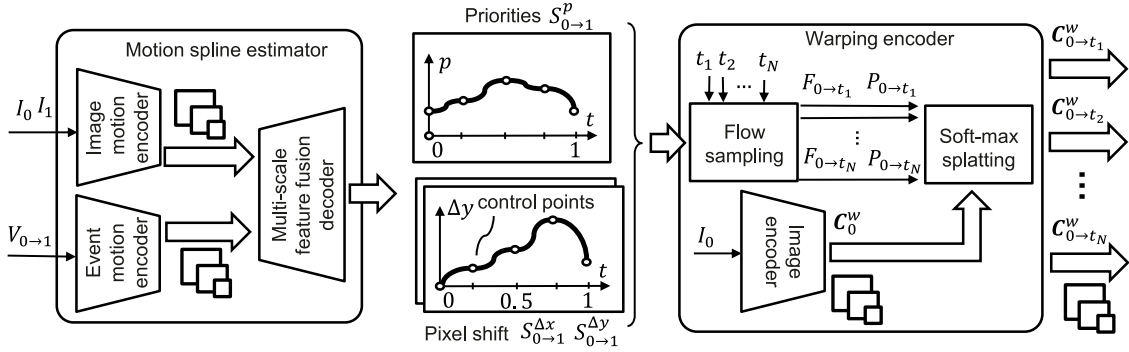

Figure 5. *Motion spline estimator* uses boundary images in addition to inter-frame events to compute *cubic motion splines* even in areas without events. Motion splines allow computing warping interpolation features for any inter-frame time, *e.g.* $\mathbf{C}_{0 \to t_1}^w$, $\mathbf{C}_{0 \to t_2}^w$, ... $\mathbf{C}_{0 \to t_N}^w$ for time $t_1, t_2, \ldots, t_N$ with minimum computational cost. For detailed explanation please refer to "motion spline estimator" in Section 3.

ing priority of each pixel of the preceding key frame as a function of time, correspondingly. Each spline we represent by K control points, *e.g.*, horizontal displacement spline $S_{0 \to 1}^{\Delta x}$ we represent as displacements $(\Delta x_0, \Delta x_{\frac{1}{K-1}}, \Delta x_{\frac{2}{K-1}}, \ldots, \Delta x_1)$ for uniformly sampled timestamps $(0, \frac{1}{K-1}, \frac{2}{K-1}, \ldots, 1)$.

Using these splines, we can compute multi-scale warping interpolation features $\mathbf{C}_{0 \to t}^w$ for any time $t \in [0,1]$ with minimum additional computational cost using the *warping encoder*. To this end, for a given time t the *flow sampling* module firstly samples flow $F_{0 \to t}$ and priority $P_{0 \to t}$ from the splines using the *cubic convolution method* [12]. Then, using the flow and the priority, the *softmax splatting* module projects multi-scale image features $\mathbf{C}_0^w$, extracted from the preceding boundary image I_0 by *image encoder* to time t. Note, that we experimented with different representations of inter-frame motion, including polynomials, Bezier splines, Natural B-Splines and they all performed very similar. We adopted the representation described above since cubic convolution methods allow to sample motion efficiently.

Using images and events for motion estimation: Time

lens only uses events for motion estimation because they contain information about non-linear inter-frame motion. However, in low contrast areas, where temporal brightness changes are below the contrast threshold of a camera, events are not triggered [5]. This causes interpolation artifacts such as shown in Figure 1(c). These artifacts could be avoided if Time Lens used images in addition to events. However, since it relies on bi-linear interpolation [9] for image warping, it requires motion from the "non-existing" latent frame to boundary frames shown in Figure 2(c), which can only be approximated from the keyframes.

We use *softmax splatting* [17] interpolation for warping which requires motion from boundary frame to latent frame (see Figure 2(d)) and, thus, allows combining events and image-based motion estimation in the proposed *motion spline estimator*. This motion estimator has two encoders: *image-based motion encoder* for estimating motion features from the boundary images $\{I_0, I_1\}$, and *event-based motion encoder* for estimating motion features from voxel grid $V_{0 \to 1}$ of inter-frame events. These motion features are then combined in the joint decoder similar to Figure 4. We use an architecture with two encoders, because it is easier for

Method	BS-ERGB				HS-ERGB [24]	
	1 skip		3 skips		7 skips	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
FLAVR [11]	25.95	0.086	20.90	0.151	27.42	0.031
DAIN [1]	25.20	0.067	21.40	0.113	29.82	0.022
Super Slowmo [10]	-	-	22.48	0.115	30.05	0.103
QVI [28]	-	-	23.20	0.110	26.28	0.143
Time Lens [24]	28.36	0.026	27.58	0.031	**33.48**	0.017
Ours	**28.56**	**0.0222**	**27.63**	**0.026**	33.09	**0.016**

Table 1. Quantitative comparison of our method with frame- and hybrid frame+event-based methods in terms of PSNR (higher is better) and LPIPS (lower is better). For on the HS-ERGB dataset [24] we average the score over the *close* and *far* subsets.

the network to learn motion from events than images, and therefore in presence of a single encoder network simply converges to a local minimum and mostly ignores images.

4. Experiments

Next, we justify our design for multi-scale fusion and spline motion estimation modules with a series of ablation studies. Then, we compare our VFI method to other state-of-the-art image and event-based methods on several benchmark datasets, including our newly introduced *Beam Splitter Events & RGB (BS-ERGB)* dataset.

All experiments are done using the *PyTorch* framework [22]. We use the Adam optimizer [13], batch size 4 and learning rate 10^{-4}, which we decrease by a factor of 10 every 12 epochs. We train each module for 27 epochs. For training, we use a large-scale dataset with synthetic events generated from the *Vimeo90k* septuplet dataset [29] using the video to events conversion method [6].

We train the motion spline and multi-scale fusion module separately and then fine-tune them together. Firstly, we train the motion module with L_1 and SSIM losses with weights 0.15 and 0.85. Then, we freeze the motion network and train the fusion network with LPIPS [32] and L^1 losses with weights 1.0 and 2.0. For training each module we also use multistage training, which firstly trains each encoder separately with dummy decoders and then freezes encoders and training decoders, and finally trains the module. This training procedure significantly boosts the performance of our method. Due to the time limitations, we don't use this method in ablations, if not stated explicitly. For real data, we fine-tune our entire network with the losses that we use for training the fusion module. To measure the quality of interpolated images we use peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [25] metrics and LPIPS.

4.1. Ablation studies

Next, we ablate the various components of our method. For ablation studies, we randomly sample 4k training examples and 500 validation examples from the Vimeo90k [29] dataset. We first ablate the motion estimation module, and then the fusion module. For the motion module we calculate errors in non-occluded areas of warped frames. For fusion,

we compute the error of final interpolated frames. Results are summarized in Table 2 for the motion module, and in Table 3 for the fusion module.

Method	15 frames[ms]	SSIM	PSNR [dB]
Importance of Spline Flow			
Linear	200	0.856	26.83
Non-parametric	2700	**0.877**	**28.20**
Spline (ours)	**220**	0.863	27.41
Importance of Images			
Images		0.808	24.51
Events		0.853	26.82
Images and Events (ours)		**0.863**	**27.41**
Comparison with State-of-the-art			
EV-FlowNet		0.756	22.31
Time Lens Flow		0.866	27.22
Ours		**0.879**	**28.10**

Table 2. Ablation for motion estimation module: while spline motion benefits run-time performance for multiple frame interpolation, using images and events boosts accuracy. Compared to methods Time lens flow [24] and EV-FlowNet [33], our method achieves superior performance.

Method	SSIM	PSNR [dB]
Importance of Fusion		
Warping	0.886	29.42
Synthesis	0.868	29.77
Synth. & warping (ours)	**0.912**	**31.87**
Importance of Gating		
No gating	0.907	31.67
Gating (ours)	**0.912**	**31.87**
Comparison with State-of-the-art		
Time Lens Fusion	0.906	31.25
Ours	**0.919**	**32.73**

Table 3. Ablation studies for fusion module: combining synthesis and warping features boosts performance, as does gated compression mechanism. Compared to image-level fusion in [24], our method achieves superior performance.

4.1.1 Motion Estimation Module

Visualization of Spline Motion. We visualize the output of our spline estimator in Figure 6 (right, in yellow). By combining images with events, it can model the highly nonlinear trajectory of a soccer ball. Events are an integral part of modeling non-linearity since when they are removed (blue, right), the flow module defaults to using images alone and predicts linear motion.

Importance of Spline Motion. We compare against linear motion and non-parametric motion in Table 2. The results suggest that non-parametric motion estimation achieves higher accuracy (28.20 dB vs. 27.41 dB) but has a much higher run time (2700 ms vs. 220 ms) for 15 flow predictions since it needs to run for each inserted frame, while spline motion can be efficiently re-sampled once computed.
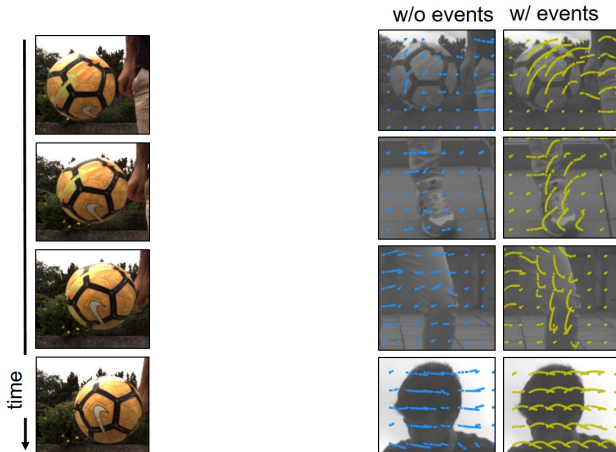
Figure 6. Spline motion visualization. The left image shows ground truth image sequence with close-ups (animation can be viewed in Adobe). The right image shows motion estimated by proposed motion spline estimator from images & events (green) and from images only (blue).

This difference becomes especially noticeable when inserting multiple frames. By contrast, linear motion estimation has a low run time but also low performance.

Importance of Images. We train two variations of our motion module: one using only events, and one using only frames and report results in Table 2. We note that combining both sensor inputs achieves the best results, boosting performance by 0.6 dB or 2.9 dB compared to single-sensor inputs. This underlines the complementarity of their information: while events provide non-linear motion cues, images provide information where events are missing.

Comparison to State-of-the-art. We compare against optical flow methods EV-FlowNet [33] and the optical flow module from Time Lens [24], which both predict non-parametric flow only from events. The resulting warping error in terms of PSNR is in Table 2. Our method outperforms the runner up [24] by 0.88 dB in terms of PSNR. Note that here we use multistage training explained in Section 4.

4.1.2 Multi-scale Fusion

Gated Compression. We firstly confirm the importance of gated compression by training the fusion network without gated compression. As shown in Table 3 (*Importance of Gating*), gated compression improves the fusion module. Next, in Figure 7 we show channel-wise averaged weights predicted by the gated compression module shown in Figure 4 for synthesis and warping features on each scale for a specific example. We conclude that: *(i)* synthesis features are used for non-rigid objects, such as fire, while warping features are used for rigid objects, such as the bottle (compare 1 & 2); *(ii)* areas occluded in the closer left frame I_0 are filled from the right frame I_1 (e.g. see 3 & 4); *(iii)* on
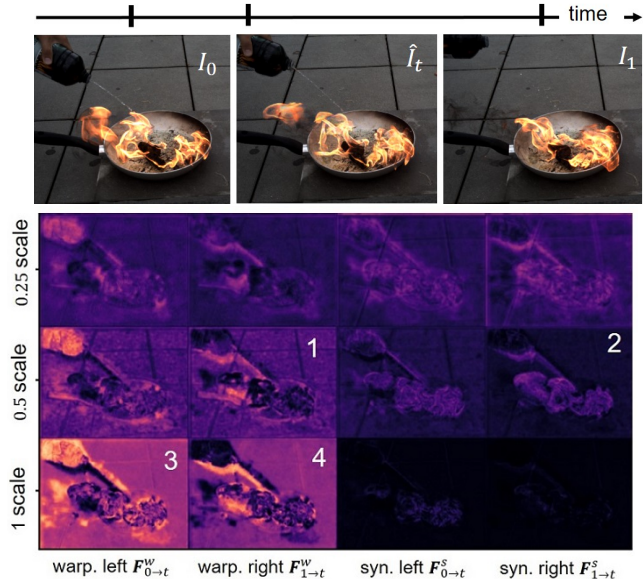

Figure 7. Gated compression weights visualization. The top figure shows key frames and latent interpolated frame. The bottom figure shows average weight prediction by gated compression for synthesis and warping features on each scale (smaller weight shown in colder colors).For details, please refer to "gated compression weight" mini-section in Section 4.1.2

Table 4. Details of proposed BS-ERGB dataset compared to similar GEF dataset. [26]

	BS-ERGB Ours)	GEF [26]
Event Camera	970×625	190×180
№ sequences	123	20
Scene dynamic	high-speed	low-speed
RGB Camera	970×625, 28 fps	1520×1440, 20 fps
Camera motion	moving & static	moving & static
Seq. length	100-600 frames	200-250 frames

a finer scale warping interpolation features are preferred to synthesis features.

Importance of Fusion: We first study the effect of combining synthesis and warping features during fusion, which we show in Table 3. We see that combining features from both modules boosts performance by 2.1 dB.

Comparison to State-of-the-art: We also compare against the image-level fusion module from [24]. As shown in Table 3 multi-scale feature-level fusion performs better by 1.48 dB. Note that for this final comparison we use multistage training explained in Section 4.

4.1.3 Beamsplitter Events and RGB dataset

We built a new hybrid setup, that uses a FLIR 4096×2196 RGB global shutter camera and a Prophesee Gen4 1280×720 Event camera mounted on a rigid case with a 50/50 one-way mirror to share incoming light. With this
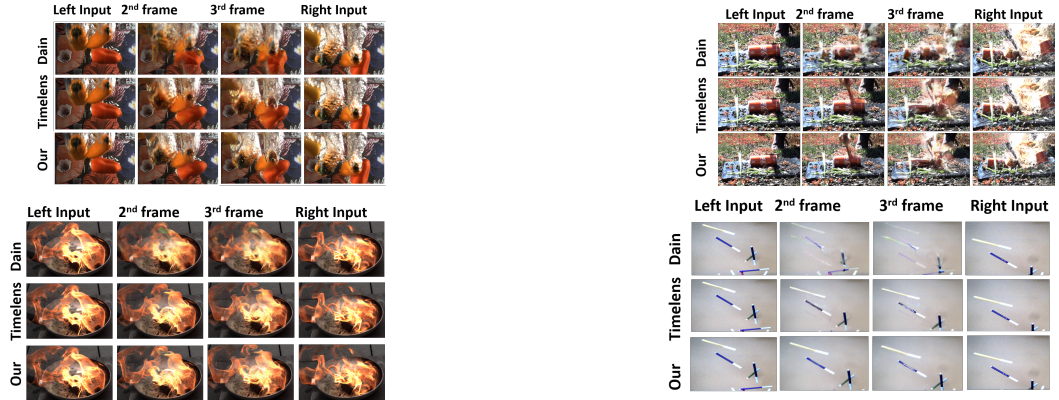
Figure 8. Qualitative results for the proposed method and its closes competitor on test sequences from the BS-ERGB: "Water" (top-left), "Fire" (bottom-left) "Fast motion" (top-right) and "Thin objects" (bottom-right). For each sequence, the figure shows our interpolation results (the animation can be viewed in Acrobat Reader) and close-up interpolation results on the right.

setup, we have collected BS-ERGB dataset, summarized in Tab. 4. As shown in the table, compared to the GEF dataset [26], proposed dataset is larger, has higher event resolution and contains high-speed scenes, with a moving and static camera. Also, in contrast to GEF, the proposed datasets include challenging scenarios with fire, water, shadows, transparent objects, high-frequency patterns, small and colorful objects, thin structures, close and far away scenes, capturing both linear and highly non-linear motion.

4.1.4 Benchmarking

We quantitatively compare our method to state-of-the-art image-based and image-and-event-based VFI methods HS-ERGB [24] and newly introduced BS-ERGB dataset, featuring realistic events and high-quality images. As shown in Table 1, the proposed method greatly outperforms not only the best image-based methods, but also the event-and-image-based Time Lens by up to 0.2 dB in PSNR and improves its LPIPS score by up to 15%.

In Figure 8 we also show qualitative comparison of our method to state-of-the-art image-based DAIN [1] and image-and-event-based Time Lens [24] methods. From the examples, it is clear that our method outperforms DAIN on the scenes with deformable objects and illumination change, such as scenes with fire and water splashes. The proposed method also outperforms Time Lens on especially complex scenes with thin and low contrast objects.

Timing Experiments: We highlight the computational efficiency of our method by comparing the per-frame computation time at different upsampling factors, for both Time-Lens [24] and our method. The results are reported in Tab. 5. Especially at high upsampling factors (> 10) our method uses less computation per frame, leveraging the continuous flow, while TimeLens requires a constant com-

putation time.

| | Time per frame [ms] | | | | № param |
Upscale Factor	1×	3×	10×	20×	[M]
Ours	634	303	**193**	**167**	**53.9**
Time Lens [22]	**273**	**273**	273	273	72.2

Table 5. Per-frame computation at different upsampling factors.

5. Conclusion

Producing stunning slow-motion video from a low framerate video has been recently made possible through the advent of frame-based and event-based methods. Event-based methods, in particular, manage to capture the non-linear motion between keyframes, especially in high-speed and highly dynamic scenarios, by using events in the blind-time between frames. Existing event-based approaches rely on simple fusion and chunked flow which causes ghosting, temporal inconsistencies, and high computational latencies. In this work, we presented a novel method that addresses all of these limitations. By leveraging continuous flow, our method produces a temporally consistent flow that can be efficiently queried at several timesteps for multi-frame insertion. Finally, our novel multi-scale fusion module significantly reduces ghosting artifacts. We demonstrate an up to 0.2 dB improvement in PSNR and 15% improvement in LPIPS score over state-of-the-art methods while being significantly faster.

6. Acknowledgment

7. Dataset Licenses

In this work we use the HS-ERGB dataset [22] which is published under the "TimeLens Evaluation License" and can be found under the URL http://rpg.ifi.uzh.ch/timelens. The Vimeo90k dataset [26] is published under the following URL http://toflow.csail.mit.edu/.

8. Network structure

In this section, we provide a detailed description of our network. As explained in the paper, our network consist of *spline motion estimator* and *multi-scale feature fusion* modules. The spline motion estimator consists of 2 encoders and one joint decoder and the multi-scale fusion network consists of 4 encoders and one joint decoder. In both networks, encoders and decoders have the same structure shown in Fig. 10 and Fig. 9.

All decoder and encoder are defined by three parameters: depth D, the maximum number of features C_{max} and the base number of features C. We provide these parameters for the motion estimator module and multi-scale fusion module in Tab. 6.

Module	D	C_{max}	C
Fusion encoders & decoder	3	128	32
Motion estimator encoders & decoder	4	256	64

Table 6. Parameters of encoder and decoder for motion estimator and multi-scale fusion modules.

9. Training Details

In this section, we provide additional details about the training procedure.

Thresholded losses. As we mentioned in the paper, we use SSIM and L^1 losses to train the motion estimator. We found that it is beneficial to use these losses only in the areas where they are beyond a certain threshold (0.06 for L^1 and 0.4 for SSIM). Intuitively, this helps to exclude occluded areas and areas with brightness constancy violation from the loss computation.

Multi-stage training. We developed a multi-stage training procedure that improves the performance of the motion estimator (see Tab. 7) as well as the multi-scale fusion module (see Tab. 8). We use the same procedure for the motion encoder and the multi-stage fusion module since they share a similar high-level network structure. Firstly, we train the image and event-based encoders separately, each with its own "dummy" decoder. Secondly, we select similar performing checkpoints for each encoder, we remove the dummy encoders and train joint decoder while freezing parameters of the encoders. Finally, we unfreeze all
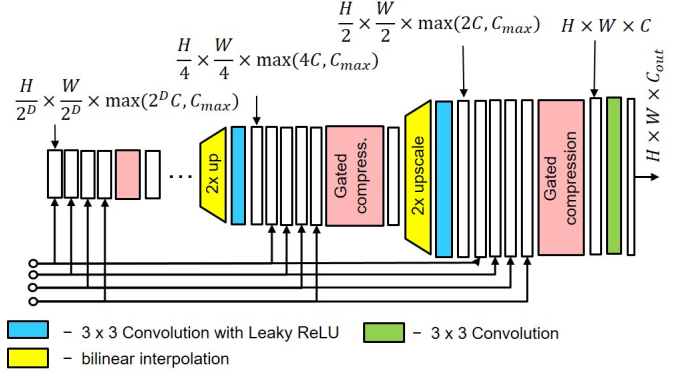

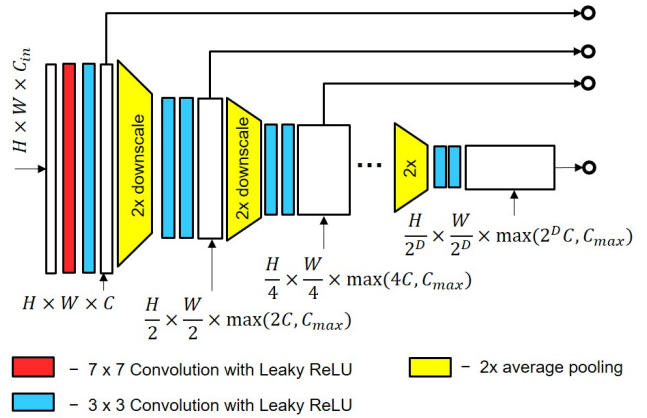Figure 9. Architecture of joint decoder.

Figure 10. Architecture overview of encoder.

weights and train the entire network. This procedure helps when encoders have different training speeds and forces the network to avoid a local minimum where the decoder uses features only from the "faster" training encoder, ignoring the other. For example, the event-based motion encoder in the motion estimation module trains much faster than the image-based motion encoder, and thus using a single-stage training procedure would learn to simply ignore images and only use events. By contrast, our multi-stage training procedure forces the network to use both inputs.

10. Additional Ablations

Here we present ablations that we omitted from the main paper due to space limitations.

Motion estimator. Here we summarize additional ablations for the motion estimator in Table 7. We found that using two separate encoders in the motion module improves results by 0.23 dB, by leaving the encoders more freedom to compute separate features for events and frames. Additionally, we found that by conducting multi-stage training, we can further improve the performance of our module by 0.69 dB. Multistage training consists of firstly training each encoder separately, when freezing encoders and train decoder,

Method	SSIM	PSNR [dB]
Importance of Double Encoder		
U-Net	0.858	27.13
Double Enc. U-Net (ours)	**0.863**	**27.41**
Importance of Multistage Training		
Single stage	0.863	27.41
Multistage (ours)	**0.879**	**28.10**

Table 7. Additional ablation for motion estimation module.

Method	SSIM	PSNR [dB]
Importance of shared synthesis encoder		
Joined	0.911	31.74
Shared (ours)	**0.912**	**31.87**
Importance of Multistage Training		
Single stage	0.912	31.87
Multistage (ours)	**0.919**	**32.73**

Table 8. Additional ablation studies for fusion module.

and finally training the entire motion module.

Multi-scale feature fusion. We found that, by using shared synthesis encoders, we can get a 0.13 dB improvement, and, by applying multi-stage training, we can further boost performance by 1.06 dB in Tab. 8. Additionally, in Fig. 11 we show the average attention weights predicted by our fusion network for synthesis and warping interpolation features on each scale. It is clear that the fusion network uses both synthesis and warping interpolation features, but prefers synthesis features on coarse scales and warping features on fine scales. This is consistent with observations in Time Lens [27].
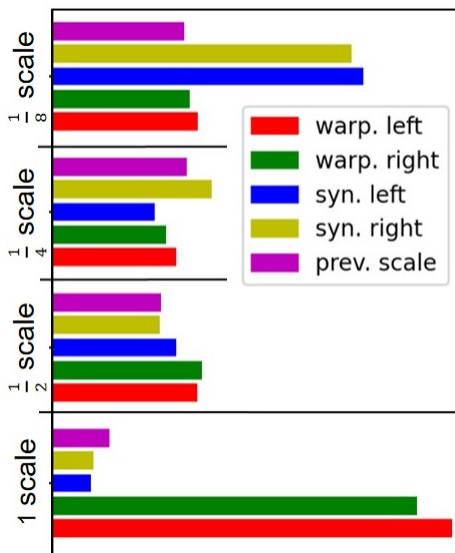

Figure 11. Average attention weights predicted by fusion network for synthesis and warping interpolation features on each scale.

11. Beamsplitter Setup and Dataset

We built an experimental setup with a global shutter RGB Flir 4096×2196 camera and a Prophesee Gen4M 1280×720 event camera [4] arranged with the beam splitter as shown in Fig. 13.

In our setup, the two cameras are hardware synchronized through the use of external triggers. Each time the standard camera starts and ends exposure, a trigger is sent to the event camera which records an *external trigger event* with precise timestamp information. This information allows us to assign accurate timestamps to the standard frames.

We use a standard stereo rectification procedure using images and event reconstructions from E2VID [23] to physically align the event and frame cameras and achieve a baseline of around 0.6mm. Next, we set the same focus for the event and frame camera and match the resolution of the RGB camera to the event camera by downsampling the images to a resolution of 1280×720. Next, we calibrate each camera separately to estimate the lens distortion parameters and focal lengths. After removing the lens distortion, we estimate a homography that compensates for the misalignment between events and images. While this procedure removes misalignment for most of the scenes, small pixel misalignment still occurs for very close scenes due to the residual baseline. We automatically compensate for this misalignment for each sequence using a small global x-y shift, computed by maximizing the cross-correlation of event integrals and temporal image difference. Finally, after image acquisition, we adjust the brightness and contrast of the images. Using the procedure above, we collected 123 diverse and challenging scenes with varying depth, some of which we show in Fig. 12. The dataset is divided into 78 training scenes, 19 validation scenes, and 26 test scenes.

References

[1] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 3703–3712, 2019. 2, 3, 6, 8

[2] Zhixiang Chi, Rasoul Mohammadi Nasiri, Zheng Liu, Juwei Lu, Jin Tang, and Konstantinos N Plataniotis. All at once: Temporally adaptive multi-frame interpolation with advanced motion modeling. *Eur. Conf. Comput. Vis. (ECCV)*, 2020. 3

[3] S. Y. Chun, T. G. Reese, J. Ouyang, B. Guerin, C. Catana, X. Zhu, N. M. Alpert, and G. El Fakhri. MRI-based non-rigid motion correction in simultaneous PET/MRI. *Journal of Nuclear Medicine*, 2012. 3

[4] Thomas Finateu, Atsumi Niwa, Daniel Matolin, Koya Tsuchimoto, Andrea Mascheroni, Etienne Reynaud, Pooria Mostafalu, Frederick Brady, Ludovic Chotard, Florian LeGoff, Hirotsugu Takahashi, Hayato Wakabayashi, Yusuke Oike, and Christoph Posch. A 1280x720 back-illuminated stacked temporal contrast event-based vision sensor with
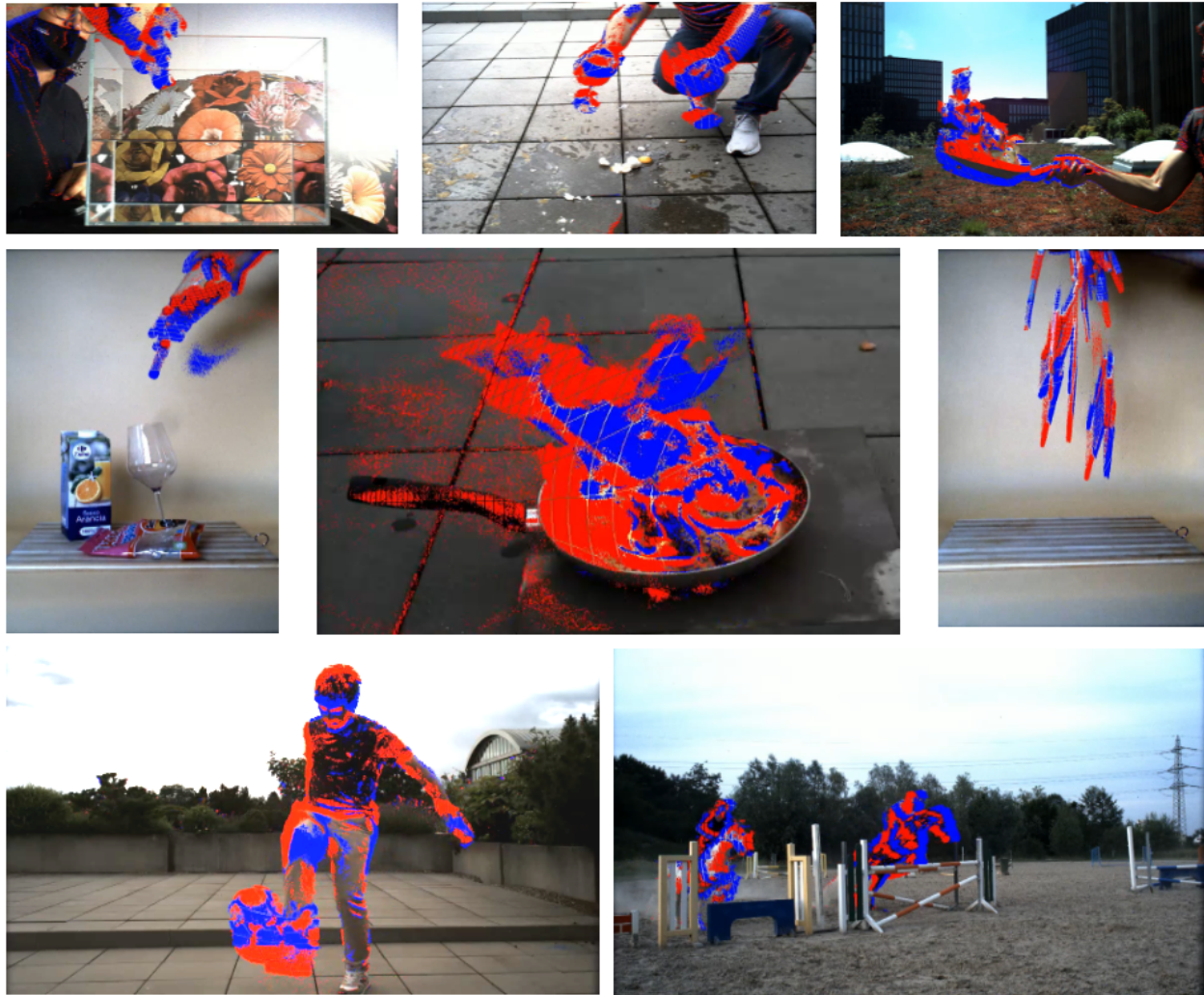
Figure 12. The dataset includes challenging scene with water, color liquid, objects and liquid, small colorful objects, fire, thin colorful objects, rotating objects on a moving background, eye catching scene (from top-left to bottom-right)
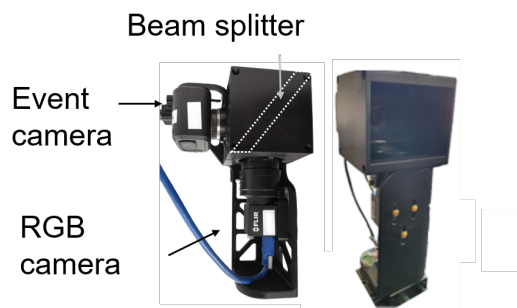

Figure 13. The beamsplitter setup mounts a FLIR RGB global shutter camera and a Prophesee Gen4 event camera [4] mounted on a case with a 50R/50T beam splitter mirror that allows the sensors to share a spatially aligned field of view. view

4.86μm pixels, 1.066geps readout, programmable event-rate controller and compressive data-formatting pipeline. In *IEEE Intl. Solid-State Circuits Conf. (ISSCC)*, 2020. 10, 11

[5] Guillermo Gallego, Tobi Delbruck, Garrick Michael Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jorg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *PAMI*, page 1, 2020. 5

[6] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, June 2020. 6

[7] Ankit Gupta, Pravin Bhat, Mira Dontcheva, Brian Curless, Oliver Deussen, and Michael Cohen. Enhancing and experiencing spacetime resolution with videos and stills. In *ICCP*, pages 1–9. IEEE, 2009. 3

[8] Jin Han, Yixin Yang, Chu Zhou, Chao Xu, and Boxin Shi. Evintsr-net: Event guided multiple latent frames reconstruction and super-resolution. In *Int. Conf. Comput. Vis. (ICCV)*, pages 4882–4891, October 2021. 3

[9] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Conf. Neural Inf. Process. Syst. (NIPS)*, pages 2017–2025, 2015. 5

[10] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 9000–9008, 2018. 3, 6

[11] Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. FLAVR: Flow-agnostic video representations for fast frame interpolation. 2021. 2, 3, 6

[12] Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, 29(6):1153–1160, 1981. 5

[13] Diederik P. Kingma and Jimmy L. Ba. Adam: A method for stochastic optimization. *Int. Conf. Learn. Representations (ICLR)*, 2015. 6

[14] Songnan Lin, Jiawei Zhang, Jinshan Pan, Zhe Jiang, Dongqing Zou, Yongtian Wang, Jing Chen, and Jimmy Ren. Learning event-driven video deblurring and interpolation. *Eur. Conf. Comput. Vis. (ECCV)*, 2020. 3

[15] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. Phasenet for video frame interpolation. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 3

[16] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 1701–1710, 2018. 3

[17] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 5437–5446, 2020. 3, 5

[18] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2017. 3

[19] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Int. Conf. Comput. Vis. (ICCV)*, 2017. 3

[20] Avinash Paliwal and Nima Khademi Kalantari. Deep slow motion video reconstruction with hybrid imaging system. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020. 3

[21] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. *Eur. Conf. Comput. Vis. (ECCV)*, 2020. 3

[22] Pytorch web site. https://pytorch.org/ Accessed: 28 March 2022. 6

[23] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. High speed and high dynamic range video with an event camera. *TPAMI*, 2019. 10

[24] Stepan Tulyakov, Daniel Gehrig, Stamatios Georgoulis, Julius Erbach, Mathias Gehrig, Yuanyou Li, and Davide Scaramuzza. Time lens: Event-based video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16155–16164, 2021. 1, 2, 3, 4, 6, 7, 8

[25] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6

[26] Zihao Wang, Peiqi Duan, Oliver Cossairt, Aggelos Katsaggelos, Tiejun Huang, and Boxin Shi. Joint filtering of intensity images and neuromorphic events for high-resolution noise-robust imaging. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2020. 7, 8

[27] Ziwei Wang, Yonhon Ng, Cedric Scheerlinck, and Robert Mahony. An asynchronous kalman filter for hybrid event cameras. In *Int. Conf. Comput. Vis. (ICCV)*, pages 448–457, October 2021. 3

[28] Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic video interpolation. In *Conf. Neural Inf. Process. Syst. (NIPS)*, pages 1647–1656, 2019. 3, 6

[29] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *Int. J. Comput. Vis.*, 127(8):1106–1125, 2019. 3, 6

[30] Qingsen Yan, Dong Gong, Qinfeng Shi, Anton van den Hengel, Chunhua Shen, Ian Reid, and Yanning Zhang. Attention-guided network for ghost-free high dynamic range imaging. In *CVPR*, pages 1751–1760, 2019. 4

[31] Zhiyang Yu, Yu Zhang, Deyuan Liu, Dongqing Zou, Xijun Chen, Yebin Liu, and Jimmy S Ren. Training weakly supervised video frame interpolation with events. In *ICCV*, pages 14589–14598, 2021. 3, 4

[32] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 586–595, 2018. 2, 6

[33] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. EV-FlowNet: Self-supervised optical flow estimation for event-based cameras. In *Robotics: Science and Systems (RSS)*, 2018. 6, 7

[34] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based optical flow using motion compensation. In *Eur. Conf. Comput. Vis. Workshops (ECCVW)*, 2018. 3