Final Project Report
CMU Computational Photography Fall 2021

Gustavo Silvera
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213
gsilvera@andrew.cmu.edu

Abstract

Video frame interpolation is the process of synthesizing in-between frames of a video stream in post processing to artificially increase the framerate by inserting new generated frames. There are several approaches to synthesizing these frames, the most common of which utilize only video frame data and infer all the motion in a scene from just two boundary keyframes. TimeLens [7] presents a novel event-plus-frames technique that utilizes the high temporal resolution of an event camera in combination with a standard video camera to perform high quality and robust frame interpolation. In this work, I present a discussion of the TimeLens [7] work, its implementation and methods, challenges in running my own experiments, and empirical results from my data capture.

1. Introduction

Capturing high frame rate video typically requires inaccessible and expensive professional-grade hardware for decent image quality. Video frame interpolation (VFI) addresses this problem by upscaling the video stream of a lower framerate camera to a higher framerate in post processing. This technique has several applications ranging from slow-motion video [3], to video compression [8], to simply increasing the smoothness of video for one's viewing pleasure.

1.1. Frame-based approach

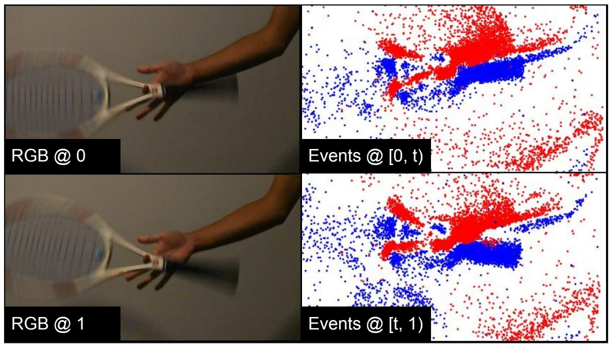
Most existing techniques used in practice for VFI utilize frame-based interpolation which relies solely on the frames output from a conventional video stream to perform interpolation. With assumptions of linear motion and brightness constancy between frames, these approaches estimate the optical flow and warp the provided keyframes to perform interpolation. Unfortunately, the assumptions laid out in these approaches might not hold when there are non-linear motions, changes in illumination, motion blur, or new objects appearing in the scene between keyframes. Modern approaches leverage additional information and might assume some nonlinear motions to improve their results, but they are still inherently limited in their ability to capture arbitrary motion. Other approaches such as phase-based [4] and kernel-based [6] VFI avoid explicit motion estimation and improve robustness to theoretically model arbitrary motions, but typically do not scale to large motions in practice due to their locality [7].

1.2. Event-based approach

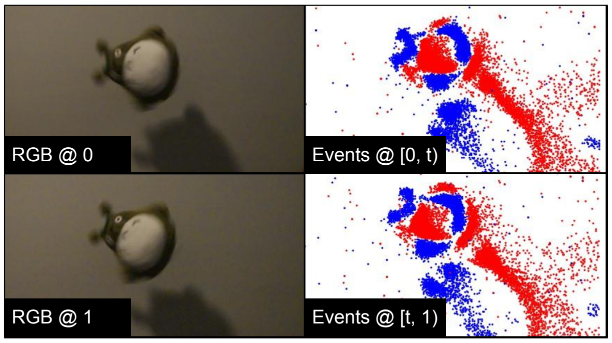
Event-based approaches utilizes event cameras to perform robust VFI even in highly dynamic scenarios. Event cameras are novel sensors that asynchronously report per-pixel intensity changes at a very high temporal resolution. The output stream contains binary "events" that highlight per-pixel intensity gradients of the scene. Event-only VFI approaches attempt to reconstruct video frames directly from the event stream by integrating the intensity gradients of accumulated events in a particular time-range. However, this integration approach is often miscalculated and results are dependent on the scene motion [7].

1.3. Event-plus-frames approach

TimeLens focuses on an event-plus-frames approach for VFI, where the combination of an event camera with a standard video camera is used to synthesize intermediate frames. The approach combines the advantages of frame-based and event-based interpolation by estimating motion from events and warping the frames accordingly. This has several advantages in that arbitrary motion can be modeled, and the interpolation can handle new objects appearing between keyframes. Additionally, the results are more robust to noise, motion blur, and illumination changes, and the results provides a higher quality interpolation compared to standalone frame-based approaches.



(a) Side-by-side of Tennis racquet being spun



(b) Side-by-side of Totoro plush tossed in the air

Figure 1. Side-by-side RGB and event "frames". The top rows illustrates I_0 and $E_{0\to\tau}$ and the bottom row illustrates I_1 and $E_{\tau\to 1}$. For this rendering, red points represent positive polarity (increasing intensity) and blue points represent negative polarity (decreasing intensity).

2. Method

2.1. Inputs

For the standard VFI setting in TimeLens, the interpolation occurs between two RGB keyframes, namely the "left" frame I_0 and "right" frame I_1, and an event stream $E_{0\to 1}$ that corresponds to the subset of all events that occurred between I_0 and I_1. Since the problem formulation is to synthesize a new frame I_τ at time $\tau \in (0, 1)$, the events can be split up into $E_{0\to\tau}$ and $E_{\tau\to 1}$ to represent the "left" events and "right" events respectively. This formulation is illustrated in Figure 1.

2.2. Module overview

TimeLens contains four complimentary modules to perform various interpolation schemes and combine them to a single frame I_τ. This approach is used to extract the advantages of various frame-based VFI schemes and combine them for optimal results. All modules use the same hourglass network and voxel grid representation of event sequences as described in [1, 3]. The first module performs warping-based interpolation to warp the boundary RGB keyframes (I_0 & I_1) using the optical flow estimation from the events ($E_{0\to\tau}$ & $E_{\tau\to 1}$). The second module refines the first module's estimates by computing residual flow. The third module performs synthesis-based interpolation to estimate a new frame by directly fusing event and keyframe information. Finally, the last module performs attention-based averaging to combine the warping-based and synthesis-based results.

2.3. Synthesis-based interpolation

Synthesis-based interpolation directly accumulates event data to the input keyframes as the events serve as gradients of the scene over time. This technique effectively handles illumination changes and sudden object appearances because unlike the warping-based approach, it does not rely on the brightness-constancy assumption. In practice, this formulation alone typically suffers when the event data is noisy or sparse as image edges and textures can get distorted in an undesirable manner. This is problematic for our event camera since its spatial resolution is much lower than the video camera ($\sim$30x less pixels) and hence accurate representations of per-pixel illumination are difficult to distinguish from neighbours.

2.4. Warping-based interpolation

Warping-based interpolation estimates the optical flow $F_{\tau\to 0}$ and $F_{\tau\to 1}$ between the desired target I_τ and the boundary keyframes I_0 and I_1 using events $E_{\tau\to 0}$ and $E_{\tau\to 1}$ respectively. Note that $E_{\tau\to 0}$ can simply be created by reversing the event sequence $E_{0\to\tau}$ and its intensities. By utilizing the dense event data, this module can effectively compute accurate optical flow of arbitrary captured motion instead of relying on the frames themselves and using a fixed-order motion assumption. By estimating optical flow in this fashion, this approach can naturally handle blur and non-linear motion encoded in the dense event stream. With the optical flow, the module warps the boundary keyframes to create $I_{0\to\tau}$ and $I_{1\to\tau}$ using differentiable interpolation [2]. While this approach typically performs better than the synthesis-based interpolation when event data is noisy or sparse, it functions on underlying assumptions of brightness constancy and fails when new objects appear in a scene.

2.5. Warping-refinement

The warping-refinement module improves the interpolation produced by the warping module by estimating residual optical flow $\Delta F_{\tau\to 0}$ and $\Delta F_{\tau\to 1}$ between the results $I_{0\to\tau}$ and $I_{1\to\tau}$ from the warping module, then proceeds to warp them again using the residual optical flow. This module also fills holes in the result with neighbouring values so occluded areas get inpainted.

2.6. Attention averaging

Finally, the attention averaging module blends the results of the synthesis and refine-warped modules in a per-pixel basis to produce the final interpolation result I_τ. The weights for the pixel blending are estimated from an attention network similar to [3, 5] to leverage the complementarity of the warping and synthesis-based approaches.

3. Experiments

A significant portion of this project included the data capture and corresponding results from my own experiments with the project. This section will describe the hardware setup used for the experiments and a discussion on additional preprocessing required for camera synchronization required by TimeLens.

3.1. Hardware Setup

The TimeLens [7] authors describe their hardware in section 2.1 as a "hybrid camera setup with a Prophesee Gen4 720p monochrome event camera (1280x720) and a FLIP BlackFly S RGB camera (1440x1080)". Additionally, their cameras are hardware synchronized with a small baseline of 2.5cm for minimal perspective difference. This dual camera setup works well for this event-plus-frames approach as the camera outputs are in usable conditions without additional preprocessing. Their cameras have a similar spatial resolution and the custom enclosure seen in Figure 5 of [7] allows for effective camera alignment, Additionally, the cameras follow a single timeline as they are hardware synchronized.

The hardware setup used for my experiments was far less sophisticated than theirs. I used course-provided hardware including a Nikon D3500 DSLR (1920x1080) for RGB video and a DVXplorer Lite (320x240) event camera. By nature of having separate recording hardware, the outputs needed to be preprocessed to align spatially and temporally.

3.2. Temporal synchronization

Without hardware synchronization between the cameras, there is no straightforward way to begin recording on both at the same time. Instead, I devised a technique to synchronize the cameras' timeline in post regardless of their starting time using a clapperboard-like approach. Since my data capture was performed indoors, for my "clap" I simply toggled the lights off and on.

For the event camera, this causes a large spike in negative events (where many pixels go dark within a short period of time) when the lights go off and a large spike in positive events (where many pixels get brighter) when they come back on. This can be easily visualized by Figure 2 as the first major peak (red) signifies the lights going out and the subsequent (green) peak denotes the lights turning on again.

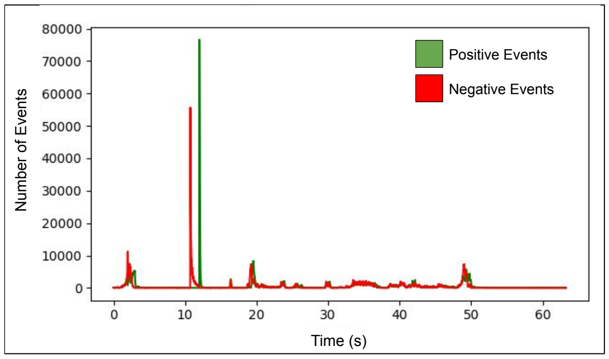


Figure 2. Graph illustrating number of events of positive (green) and negative (red) polarity over time. Datapoints were obtained by visualizing the event stream into accumulations of 10,000 bins of equal time.

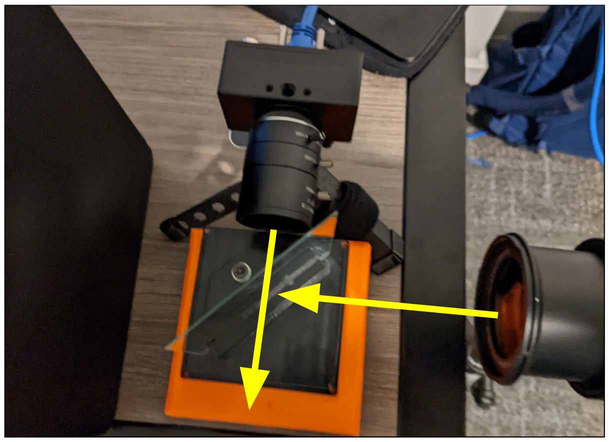


Figure 3. Camera setup with DVXplorer Lite event camera at the top, Nikon DSLR video camera to the right, beam-splitter in the middle such that the two cameras see scene photons traveling along the yellow arrows.

Note that the differences in the peaks are explained by the initial ambient light in the scene since the lights were initially on. For the RGB camera, it is trivial to find the first and last dark frames in post. With this information at hand, it is straightforward to compute the synchrony point $t_{\text{lights off}}$ where both cameras recorded the lights going out.

This "clapperboard" technique is is inherently flawed in that the (much faster) event camera could have detected the lights go out before the video camera captured it. This error is upper bounded by the inverse framerate of the video camera which for 60hz footage is less than 0.0167 seconds.

3.3. Spatial alignment

Without a mounting enclosure to physically align the cameras together as in [7] it is difficult to obtain spatial alignment required for event-pixel correspondences.

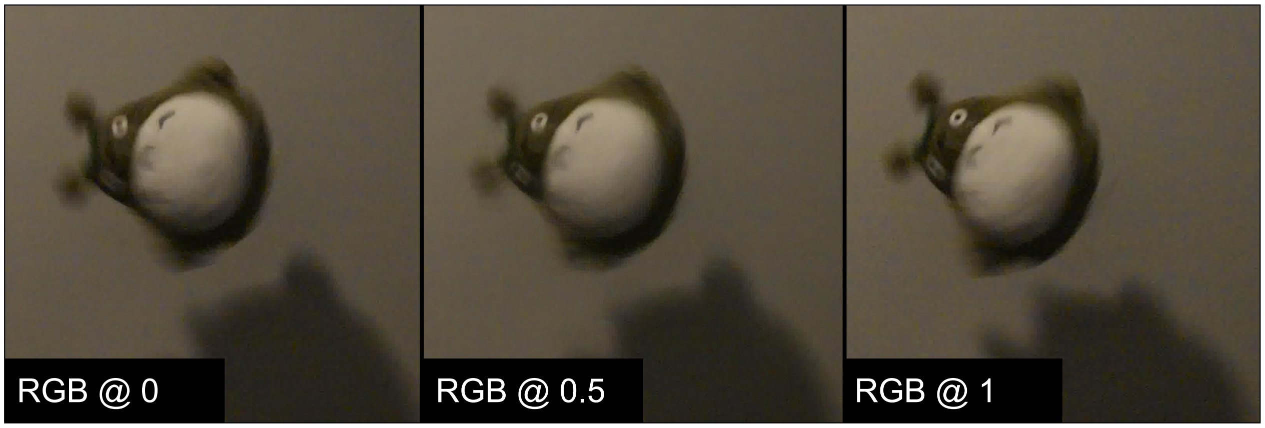


Figure 4. Result of a single synthesized RGB frame between the I_0 and I_1 shown earlier in Figure 1b. Notice the center frame I_τ is completely synthesized and lies perfectly between the boundary RGB keyframes, hence in this case $\tau = 0.5$.

One approach to spatially align the cameras is to use a beam-splitter for both cameras to view the same scene. This approach is reasonably simple to set up as shown in Figure 3 and ensures there will be no visual perspective differences between the camera views. However, by nature of the beam-splitter optics, half of the light in the scene will be lost by splitting the incoming light to two cameras.

Another approach tested is to simply position both cameras similarly to [7] and minimize the baseline difference between the two lenses. This has the advantage of not requiring further hardware but may require perspective-correction transformations in postprocessing before producing viable data.

Regardless of the approach, I still performed some visual transformations in post-processing to align the events with the RGB image. These mostly consisted of cropping, scaling, flipping (when using a beam-splitter), and translating one event stream to align with the other. While this is a manual process it was not very tedious because the aligning a single frame with time-synchronous events would be enough for the transformations to propagate across the entire data stream. There is inherent accuracy error in performing these manual modifications, but for purposes of this project both approaches along with transformations were sufficiently accurate to produce reasonable results.

3.4. Results

Several results are provided in the provided source code repository linked in the conclusion. Individual frames of the RGB camera and events corresponding to that frame can be seen in Figure 4.

With my data capture, I found the best results occured when only inserting one or two new frames. This is in comparison to the example data provided by the TimeLens [7] authors where I could get good results even when inserting seven new frames between keyframes. I believe this is primarily because their event data was far more dense (I counted $\sim$10x more data) than the stream provided from the DVXplorer Lite, as they had a higher quality event camera. The added data allows for a much finer granularity when estimating optical flow from events, a critical step in the warping based modules.

Unfortunately my lighting conditions were also not ideal for the project. While in possession of the imaging hardware, I was only able to capture data when it was already dark out and my room itself has poor interior lighting. This led to both problems in increased noise as well as blurry frames from the video. An ideal capture for this data would be outdoors with a very high shutter speed so every frame is sharp and the situation is very similar to the examples provided by [7].

Additionally, using a beam-splitter effectively halves the number of photons reaching the sensor from the scene. The low light scenario also likely contributed to the added noise in my event stream compared to the example data in the TimeLens [7] dataset. While the warping based modules are mostly robust to event noise, the synthesis modules are not, placing a higher burden on the attention averaging to prioritize pixels from the warping modules.

4. Conclusion

In this work, I present a minimal implementation of TimeLens, a project on event based video frame interpolation. My contributions are the TimeLens implementation itself (compatible with their pre-trained model) and additional source code for preprocessing and integration with the camera hardware setup described in this paper. The source code is available at this https url. These contributions should widen accessibility for users who have a DVXplorer camera and are interested in using TimeLens to performing high quality video frame interpolation.

5. Acknowledgement

This project is based on the work done in TimeLens: Event-based Video Frame Interpolation [7], from the Huawei Technologies Zurich Research Center and Department of Informatics and Neuroinformatics at University of Zurich and ETH Zurich. Guidance and hardware for this project was provided by the course professor Dr. Ioannis Gkioulekas.

References

[1] Daniel Gehrig, Antonio Loquercio, Konstantinos G Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5633–5643, 2019. 2

[2] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015. 2

[3] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018. 1, 2, 3

[4] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. Phasenet for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 498–507, 2018. 1

[5] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5437–5446, 2020. 3

[6] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–270, 2017. 1

[7] 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. 1, 3, 4, 5

[8] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 416–431, 2018. 1