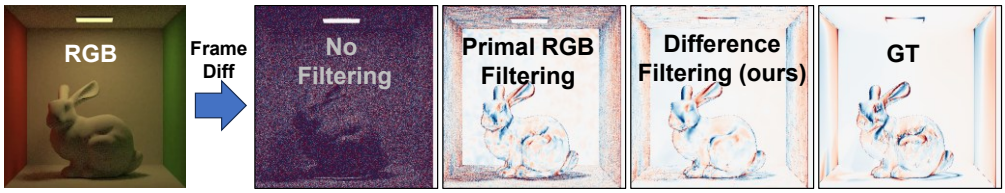
EGSR 2026

Difference-aware Filtering for Event Camera Simulation



Juhyeon Kim, Wojciech Jarosz, Adithya Pediredla

Dartmouth College

Standard Camera Measures Brightness

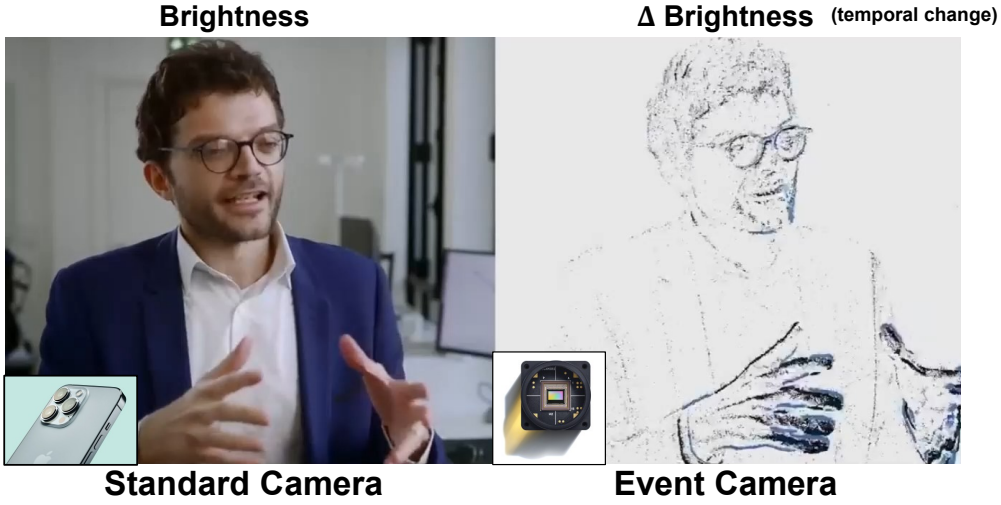
Standard Camera

Event Camera

2

When we think of a camera, we usually think of capturing images like this video. Each frame records the (click) brightness of the scene. But what if, instead of recording brightness itself, a camera only measures temporal changes in brightness? In fact, there is a special type of sensor called an event camera that exactly does that.

Event Camera Measures Brightness Change



3

Here is a side-to-side comparison. Notice that the event camera does not capture full images—it only reports brightness changes, emphasizing motion in the scene.

How does Event Camera Work?

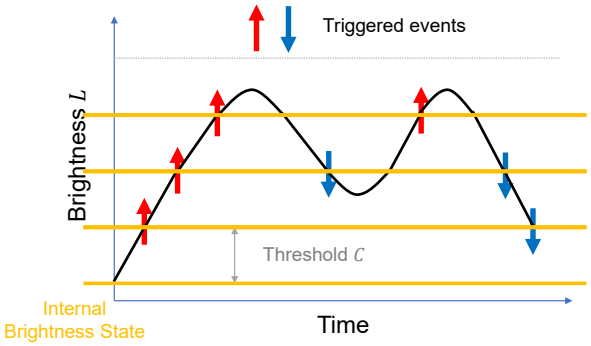
Dynamic Vision Sensor (DVS)

iniLabs

Brightness $L = \log(I + I_\epsilon)$

I : Intensity

I_ϵ: Intensity bias

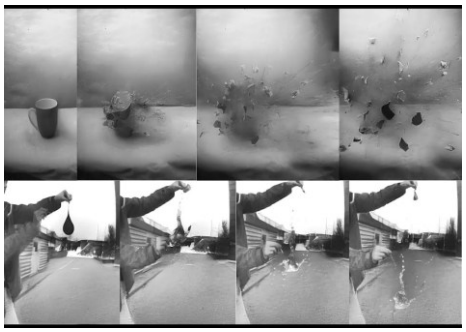


4

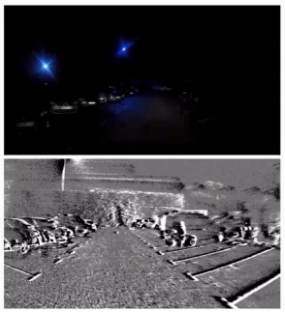
So, how does an event camera work? Event cameras use a specialized sensor called a Dynamic Vision Sensor. Each pixel continuously monitors the brightness L which is log of intensity and stores a reference value. When the brightness change exceeds a threshold, the pixel generates an event and update reference value. Brightness increase produces a positive event like this, while a brightness decrease produces a negative event and so on.

Advantage of Using Event Camera

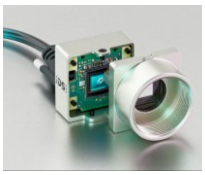
High FPS (~10000 FPS)



High Dynamic Range



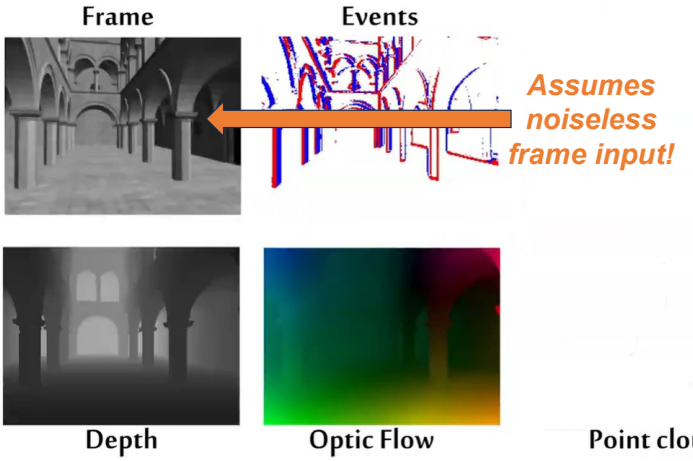
Low Power Consumption



5

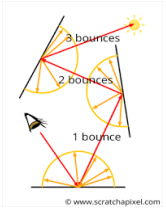
Event cameras offer several advantages over standard cameras: (1) extremely high temporal resolution for capturing fast motion, (2) very high dynamic range that works well under challenging lighting conditions, and (3) low power consumption because only brightness changes are measured. These advantages have led to growing interest in event cameras across many applications.

Event Camera Simulation : ESIM



Assumes noiseless frame input!

[Rebecq et al. 2018]

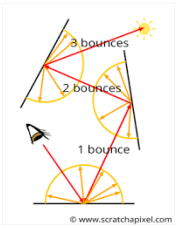


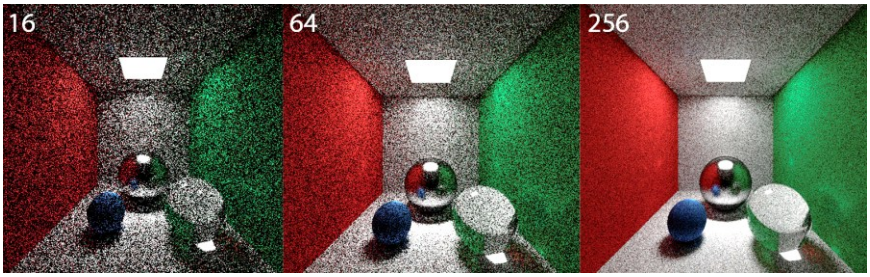
Monte Carlo Path Tracing

6

To develop and evaluate algorithms for these systems, we also need accurate simulators. The goal of event camera simulation is to render a sequence of images over time and reproduce the events triggered from these images. However, many existing simulators assume noiseless rendered inputs, often relying on rasterization, which is fast, but not physically accurate. Instead, our goal is to enable physically based event simulation using Monte Carlo rendering.

Event Camera Simulation : Monte Carlo Path Tracing



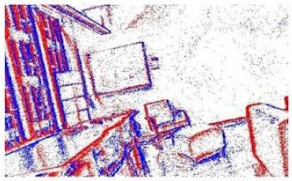


Requires Large SPP for Converge!

7

However, as you know, Monte Carlo method often requires a lot of samples to get a noiseless image.

Event Camera Simulation : Monte Carlo Path Tracing

	RGB	Event	
			
FPS	120	4800	**x 40**
Signal Magnitude	1 (primal)	0.01 (difference of primal)	**x 100**

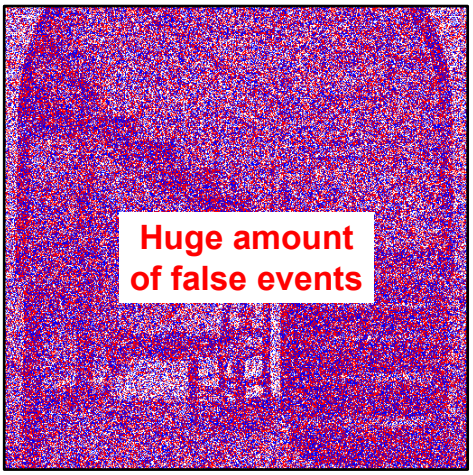
x 4000 SPP budget required than standard RGB rendering

8

This is even more challenging for event camera simulation, which requires much higher frame rates and measures small difference signals with inherently lower SNR, leading to a much higher sample budget than standard RGB rendering.

Event Camera Simulation : Monte Carlo with Low SPP

Input Image Sequence



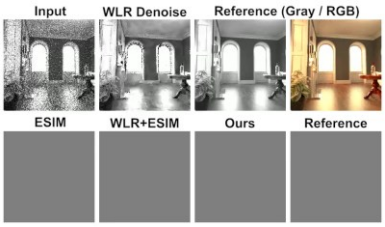
Event Generation

9

Therefore, making the simulation low spp is important, but if we naively use it, its temporal noise triggers a huge number of false events throughout the image.

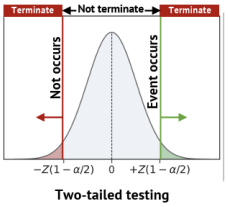
Previous Monte Carlo Event Camera Simulation

Image Denoising



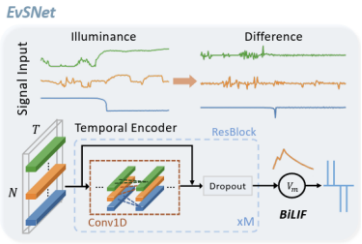
[Tsuji et al. 2023]

Adaptive Time Sampling based on Statistics



[Manabe et al. 2024]

Generate Event via Spiking Neural Network



[Li et al. 2025]

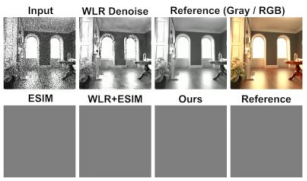
Still requires lots of samples (32-256 spp)

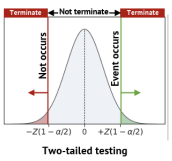
10

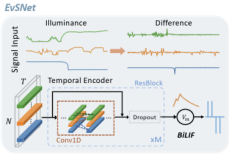
Recently, several methods have been proposed for low-spp input event simulation, including adaptive denoising, adaptive sampling, and spiking neural networks. However, they still require tens of samples per pixel, making efficient event simulation challenging.

Previous Monte Carlo Event Camera Simulation

Previous Works







Still requires lots of samples (32-256 spp)

→ Our method (*EventSVGF*) only uses 2 spp / frame (x10~100)

Based on real-time rendering technique (SVGF)

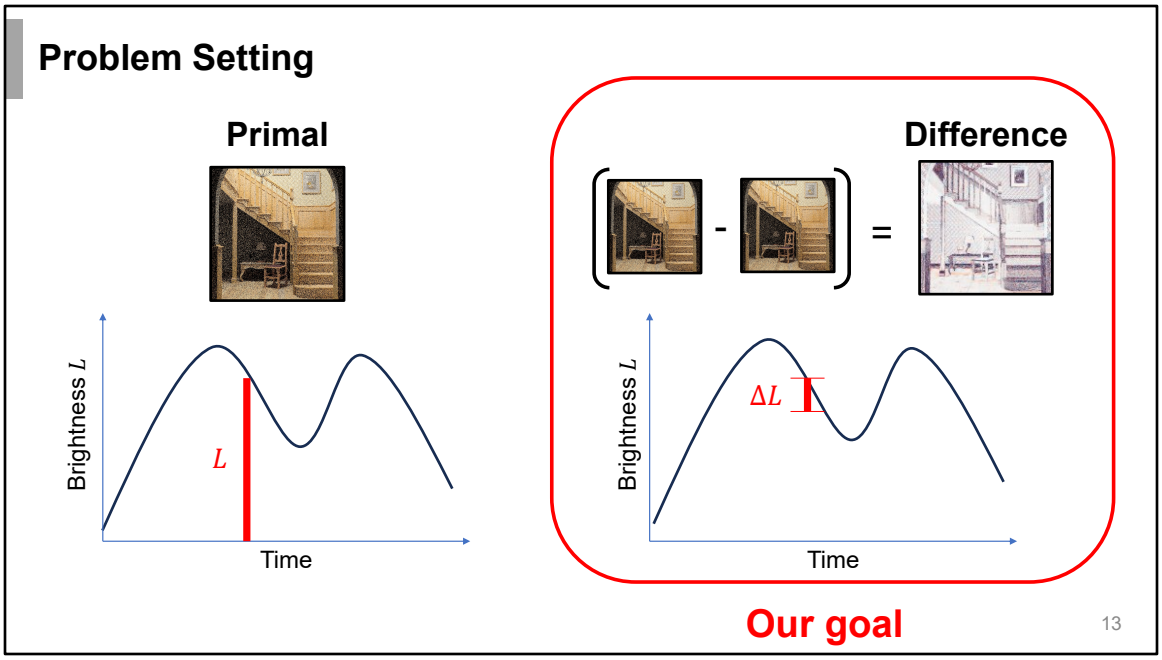
11

In contrast, our method namely EventSVGF based on real-time rendering technique operates with only 2 spp, allowing us to suppress false events while preserving true temporal changes even with very low-sample budgets.

Proposed Method

12

(3:20)



Before discussing the algorithm, let's first define the problem. Our goal is not to reconstruct primal brightness value itself, but to accurately estimate temporal brightness changes. One might ask: why not simply render brightness, like a standard camera, and generate events from it?

Event Camera is High-Frequency Sensor

Event camera is accurate in ΔL, not L

High-frequency Low-frequency

14

While this is possible, event cameras are primarily driven by high-frequency temporal signals, not guaranteeing low frequency accuracy. Let's look at this example of GT brightness change over time. If we reconstruct this brightness from occurred events, it often undergoes low frequency drift, which indicates low-frequency component is inherently ill-posed for event camera. So, event camera is accurate in capturing (click) \Delta L, high frequency region not (click) L, low frequency region which justifies simulating focusing on \Delta L.

High-Frequency Temporal Accuracy

$\sigma_{\Delta L}$: noise of ΔL

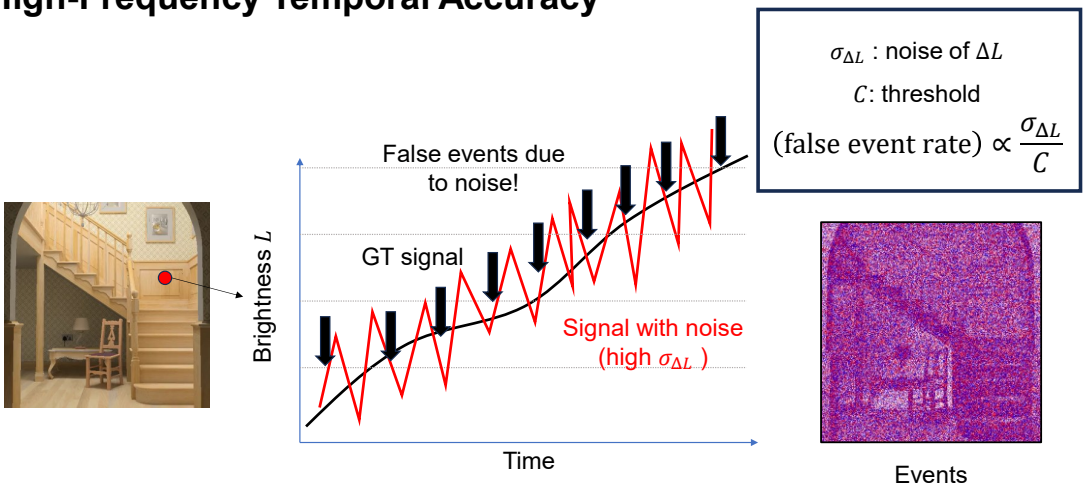
C: threshold

$$(\text{false event rate}) \propto \frac{\sigma_{\Delta L}}{C}$$

15

Theoretically, this high-frequency temporal accuracy is closely related to false event suppression in simulation.

High-Frequency Temporal Accuracy



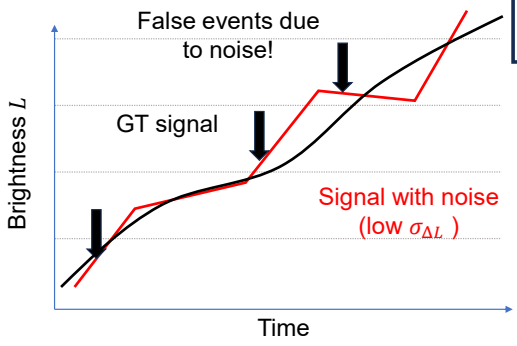
Think about this GT brightness change over time. And the red curve here is a Monte Carlo estimate which has high frequency noise. Due to its temporal instability, it causes many spurious threshold crossings and false events.

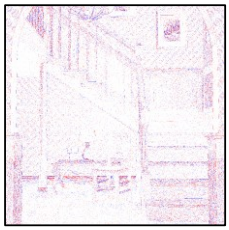
High-Frequency Temporal Accuracy

$\sigma_{\Delta L}$: noise of ΔL

C: threshold

$$(\text{false event rate}) \propto \frac{\sigma_{\Delta L}}{C}$$





Events

Suppressing $\boldsymbol{\sigma_{\Delta L}}$ is important to reduce MC-noise induced events

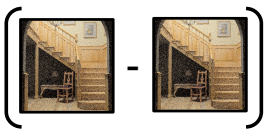
17

In contrast, the second signal has a similar brightness error but is much more temporally stable, resulting in far fewer false events. Therefore, our main objective is not to minimize brightness error itself, but to accurately estimate temporal brightness changes.

Overall Idea

EventSVGF

Goal : render brightness change ($\Delta \mathbf{L}$) effectively



(1) Correlated Sampling

(2) Difference-aware Denoising (based on SVGF)

18

So, EventSVGF is designed to accurately simulate brightness changes. The method consists of two key components: correlated sampling and difference-aware denoising. Next, I will explain each component in more detail.

Correlated Sampling

$$\Delta L_t = L_t - L_{t-1}$$

$$\mathrm{Var}[\Delta L_t] \propto (1 - \rho)$$

ρ : correlation coefficient

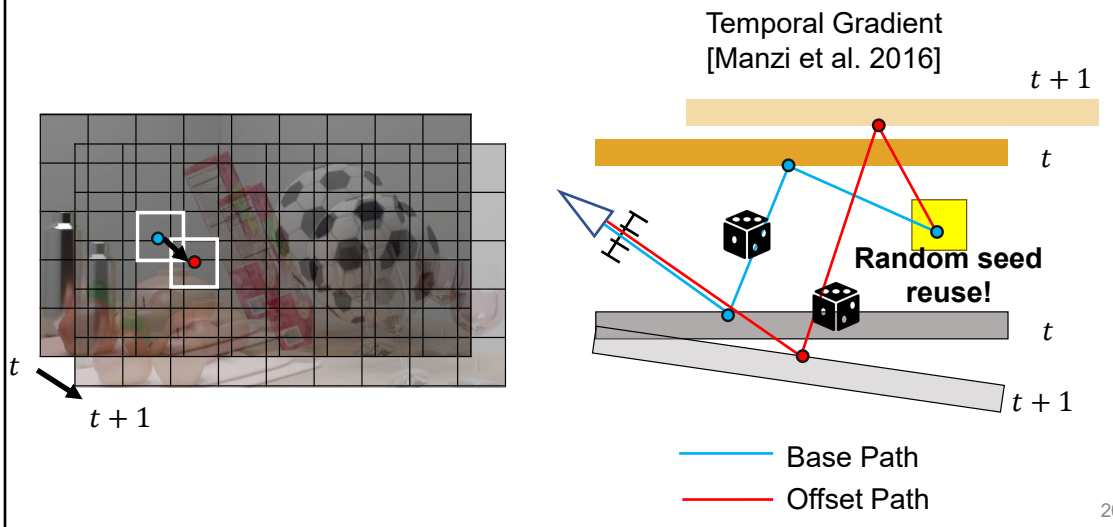
× **Small ρ → High variance!**

✓ **High ρ → Low variance!**

19

First, let's look at correlated sampling. Our goal is to estimate the temporal difference signal, and the variance of this difference depends on the correlation between the two frames. By increasing the correlation between consecutive frames, we can significantly reduce the variance of the difference signal.

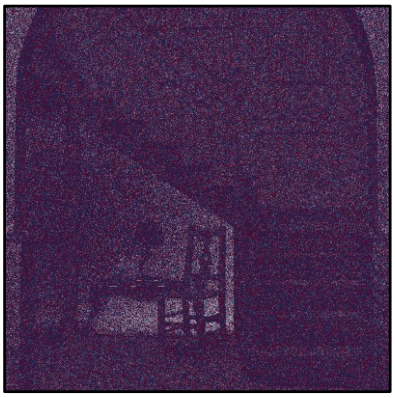
Gradient-Domain Rendering and Shift-Mapping

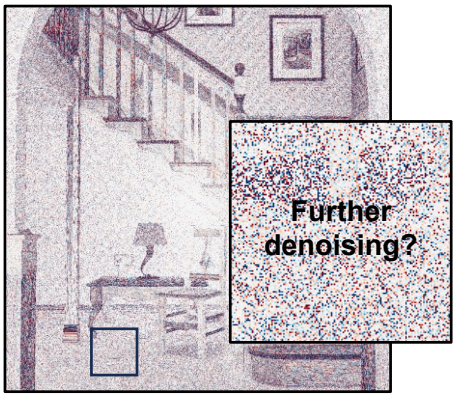


20

In fact, exploiting correlation between samples has been widely studied in gradient-domain rendering through a technique known as shift mapping. The key idea is to estimate image gradients by subtracting two highly correlated paths—a base path and an offset path—that have similar path throughput. Several methods are possible, but we use random replay, which simply reuses the same random seed across consecutive frames.

ΔL with 2-spp





Uncorrelated

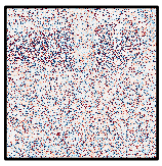
Correlated

21

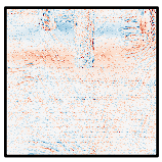
Here, we compare independent and correlated sampling at 1 spp. Correlated sampling substantially reduces noise in the brightness difference, but further denoising is still needed.

Denoising with SVGF

ΔL

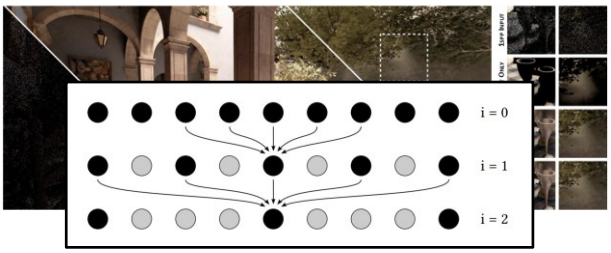


$\mathrm{D}[\Delta L]$



Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination

Christoph Schied, NVIDIA, Karlsruhe Institute of Technology
Anton Kaplanyan, Chris Wyman, Anjul Patney, NVIDIA
Chakravarty R. Alla Chaitanya, NVIDIA, University of Montreal, McGill University
John Burgess, Shiqiu Liu, NVIDIA
Carsten Dachsbacher, Karlsruhe Institute of Technology
Aaron Lefohn, Marco Salvi, NVIDIA



A-Trous Wavelet Filter

22

Now, let's discuss the denoising part. To further reduce noise in the brightness changes, we build upon the SVGF pipeline. We will skip the details, but SVGF is a real-time denoiser based on edge-avoiding à-trous wavelet filtering.

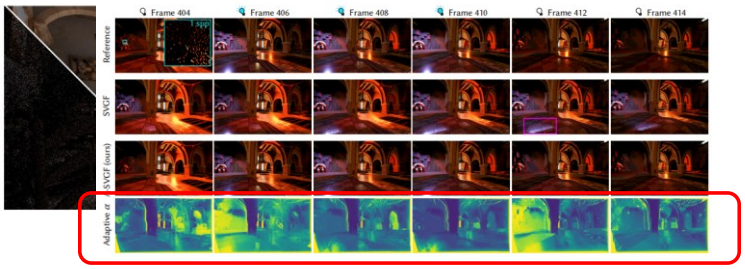
Denoising with SVGF

Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination

Christ … Karlsruhe Inst …

Gradient Estimation for Real-Time Adaptive Temporal Filtering

John … Shi … CHRISTOPH SCHIED, CHRISTOPH PETERS, and CARSTEN DACHSBACHER, Karlsruhe Institute of Technology, Germany



Temporal Gradient!

23

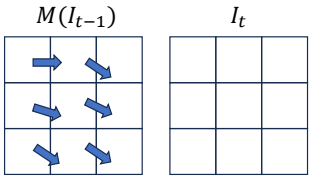
In fact, one of the previous work has already shown that SVGF can be used to estimate temporal gradients in Monte Carlo rendering.

Motion Alignment

Motion Vector Aligned

$$I_t - M(I_{t-1})$$

Prev gradient rendering focus on motion vector aligned gradient

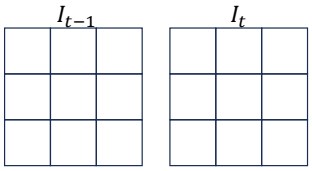


[Manzi et al. 2016]

Non-Motion Vector Aligned

$$I_t - I_{t-1}$$

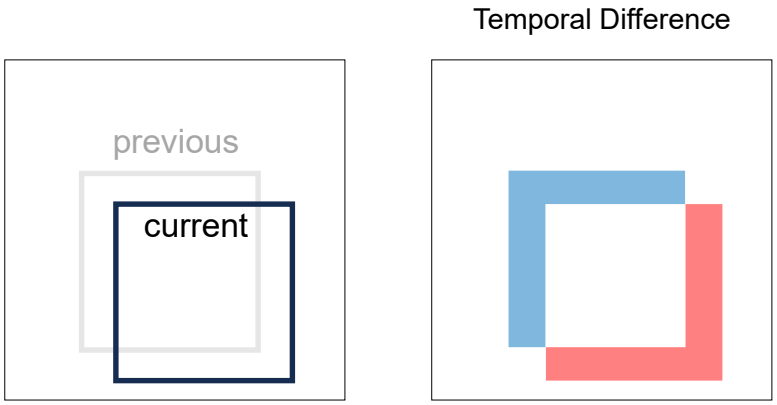
What we need for event-camera simulation



24

However, there is an important distinction in the context of event cameras. Previous methods typically consider motion-compensated temporal differences, where corresponding points are matched across frames using motion vectors. This reduces variance so is desirable in standard rendering. In contrast, event cameras do not work in this way. There is no motion-compensation. Event cameras simply respond to the change in brightness observed at each pixel. As a result, we must estimate the non-motion-compensated temporal difference, which requires a modified denoising strategy.

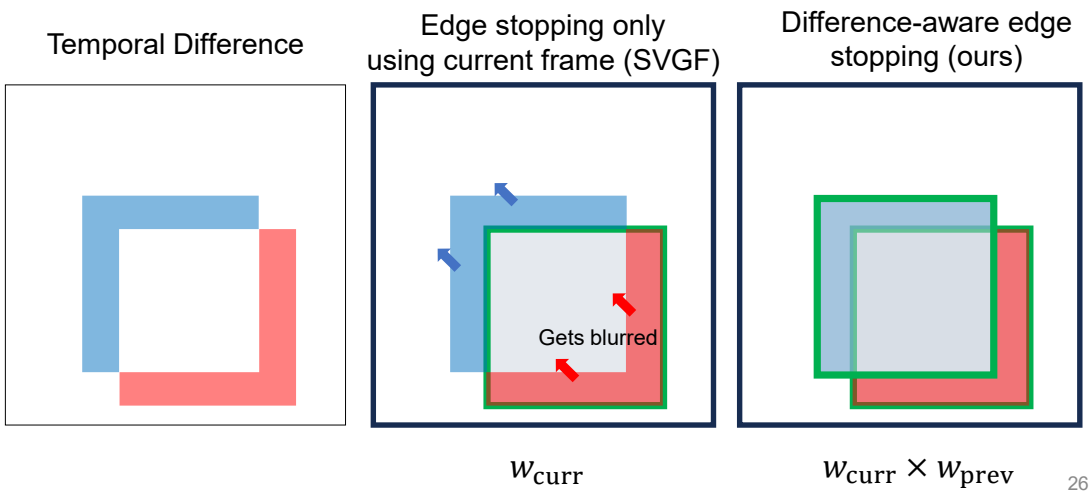
Difference-aware Edge Stopping Filter



25

To address this issue, we propose a difference-aware edge-stopping filter. Consider a simple example where a square moves toward the lower-right direction. Then resulting temporal difference will look like this.

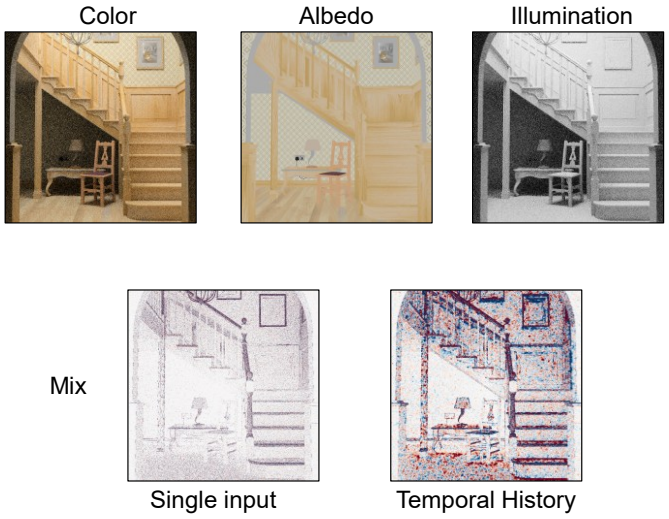
Difference-aware Edge Stopping Filter



26

If we denoise this signal using only information from the current frame, the difference regions tend to be blurred by the filtering process. Instead, we incorporate information from both the current and previous frames when computing the filter weights. Implementation is straightforward. We simply replace the original edge-stopping weight with the product of the weights computed from the current and previous frame features.

Difference-aware Adaptation



27

We will not go into the details here, but several other components of the SVGF pipeline also need to be adapted for temporal differences. In particular, we modify operations such as albedo demodulation and temporal accumulation to make them difference-aware.

Intensity to brightness

Intensity → Log (Intensity)

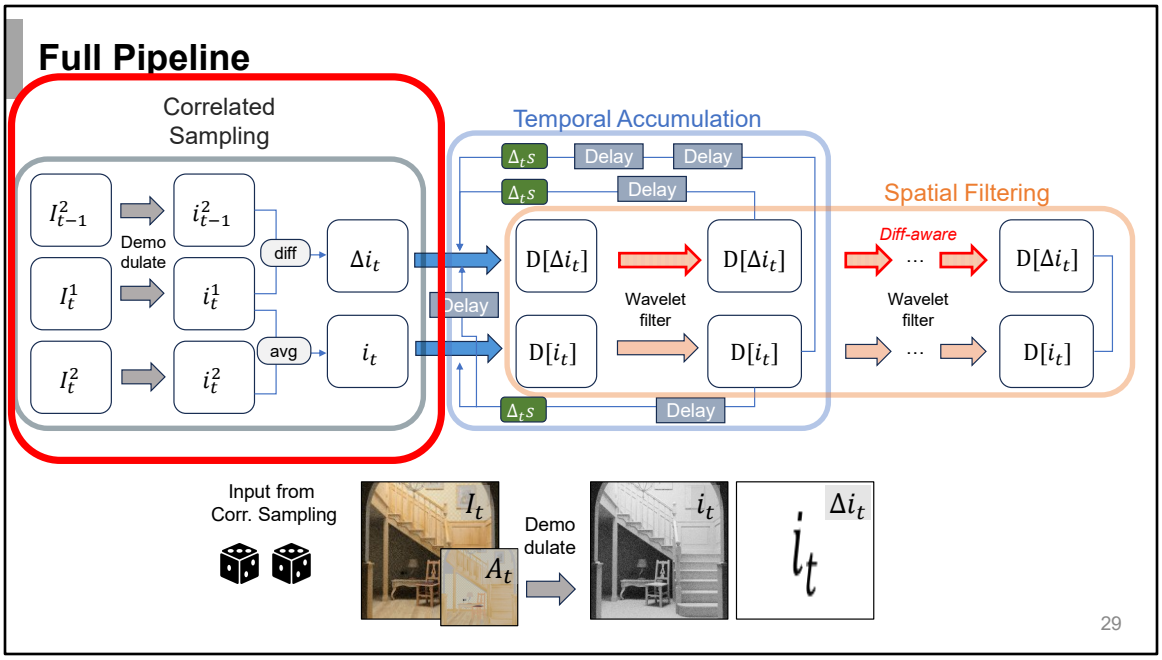
$$
\begin{aligned}
&\log(I_\epsilon + I_t) - \log(I_\epsilon + I_{t-1}) \\
&= \log(I_\epsilon + A_t i_t + E_t) - \log(I_\epsilon + A_{t-1} i_{t-1} + E_{t-1}) \\
&= \log\left(1 + \frac{(A_t i_t - A_{t-1} i_{t-1}) + (E_t - E_{t-1})}{I_\epsilon + A_{t-1} i_{t-1} + E_{t-1}}\right) \\
&= \log\left(1 + \frac{A_t(i_t - i_{t-1}) + (A_t - A_{t-1}) i_{t-1} + (E_t - E_{t-1})}{I_\epsilon + A_{t-1} i_{t-1} + E_{t-1}}\right)
\end{aligned}
$$

- I_ϵ: Intensity threshold
- I_t: Intensity at t

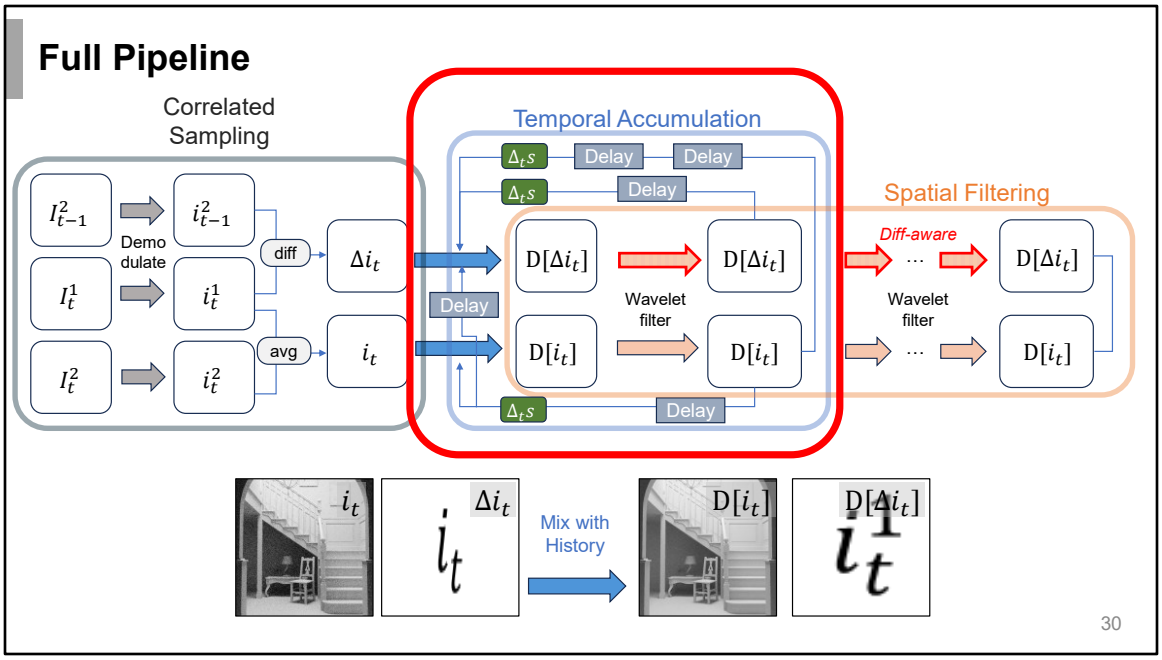
- A_t: Albedo at t
- i_t: illumination at t
- E_t: Emission at t

28

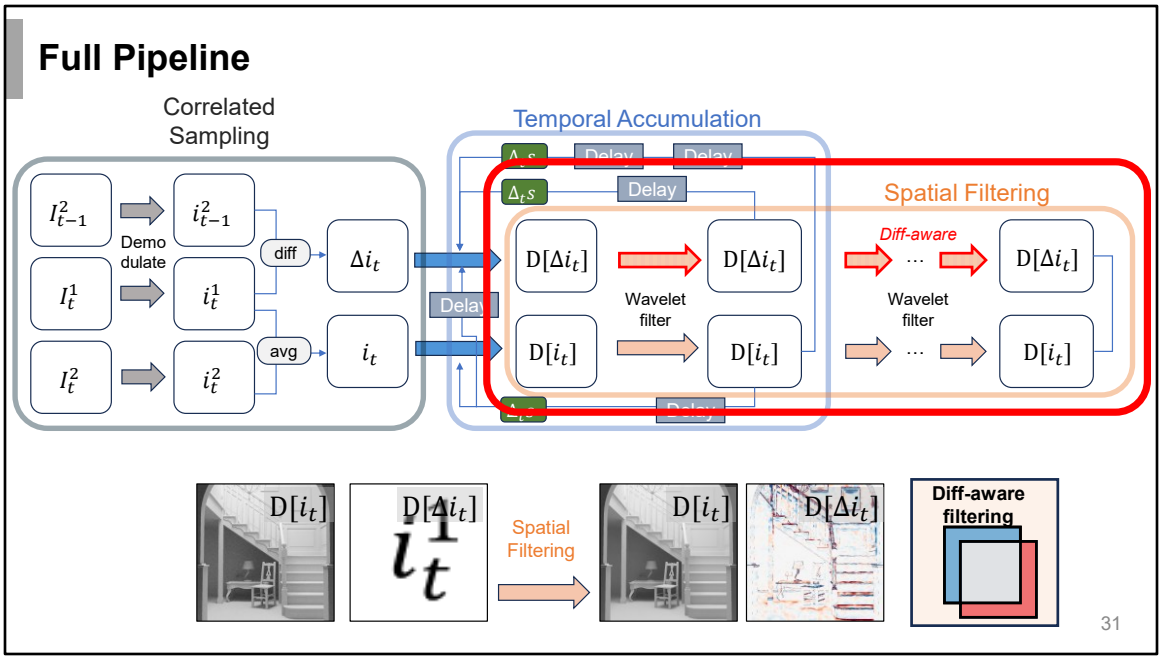
Finally, we need to adapt all components to operate on log-intensity rather than intensity itself, since event cameras respond to changes in log brightness. We will skip the details here.



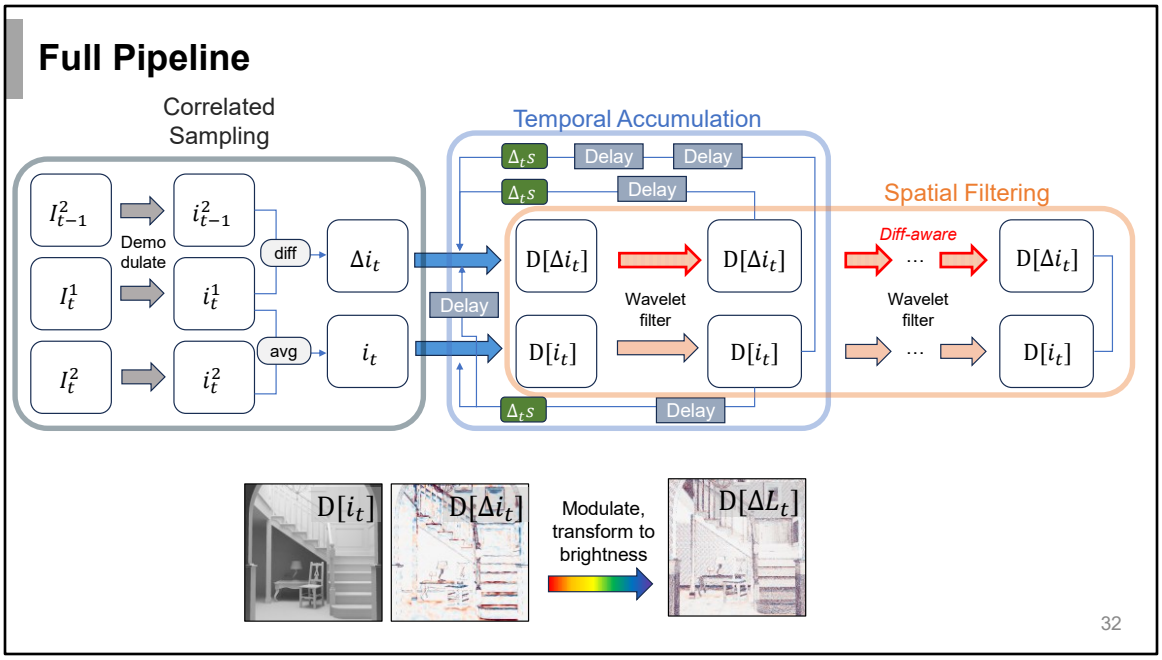
Here is the full pipeline. We first generate demodulated difference image with correlated sampling.



Then, we perform temporal accumulation, mix with history difference data.



We also perform spatial filtering using proposed difference aware filtering.

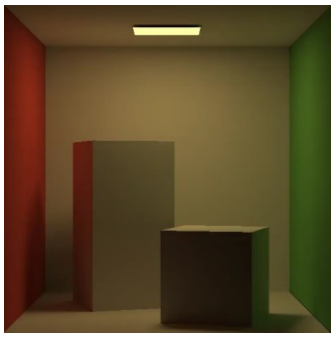
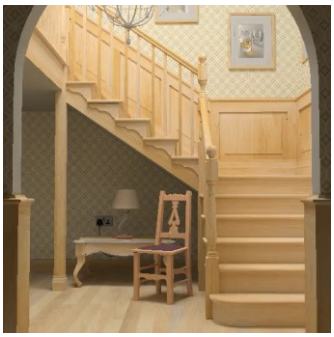


Finally, we reconstruct brightness change and generate event accordingly.

Result

Now results.

Scene Setting

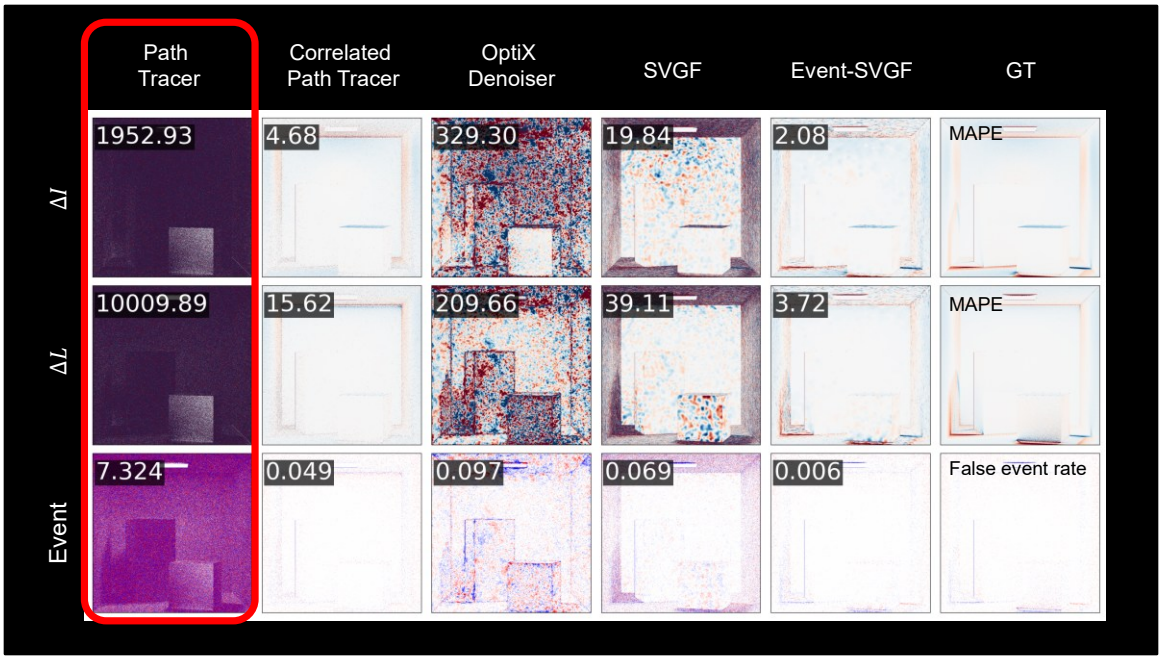




ΔI : intensity / ΔL : brightness (log-intensity) / Event

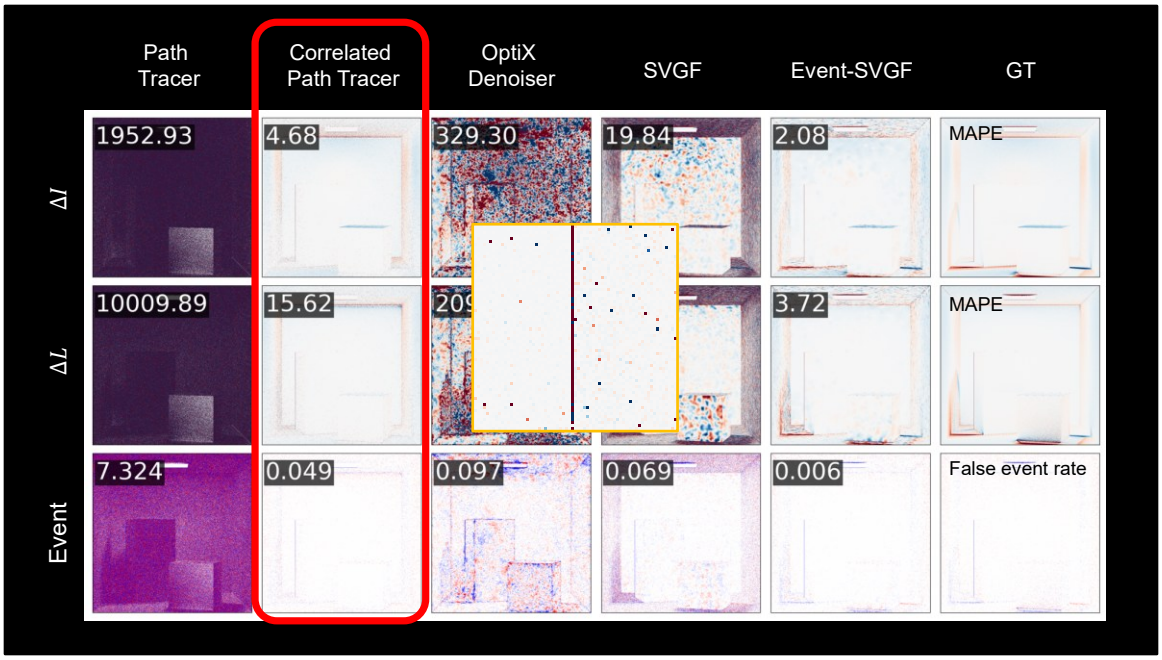
(implemented with NVIDIA Falcor engine)

34

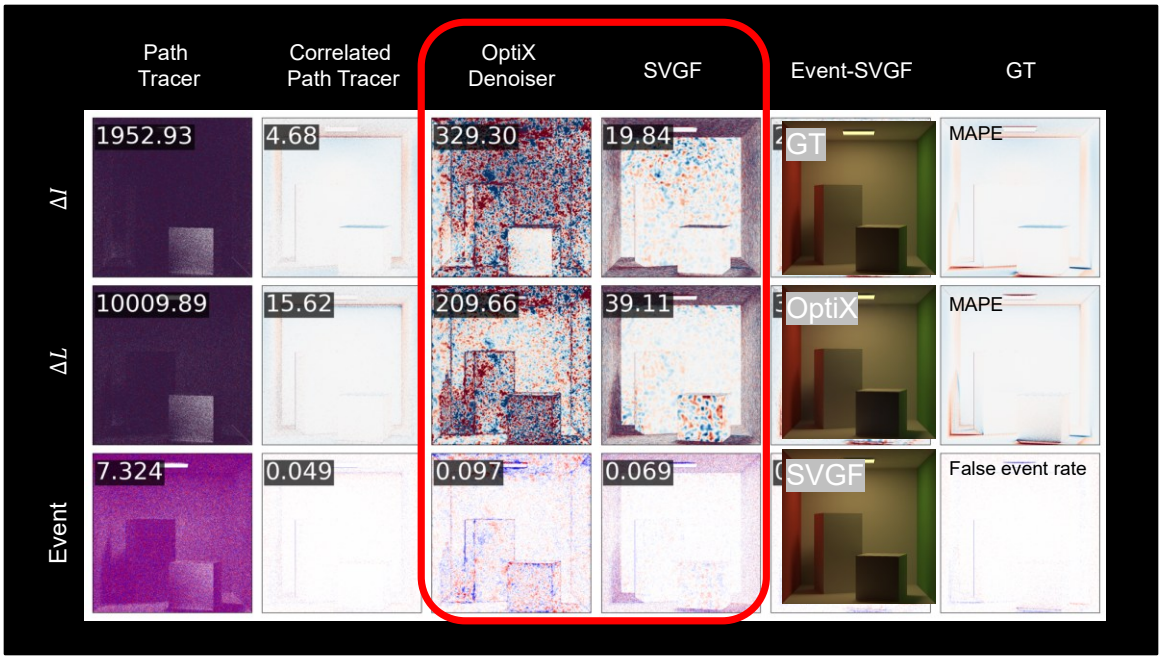
We simulate these scenes with camera moving forward and visualize temporal change of intensity, brightness and resulting event.



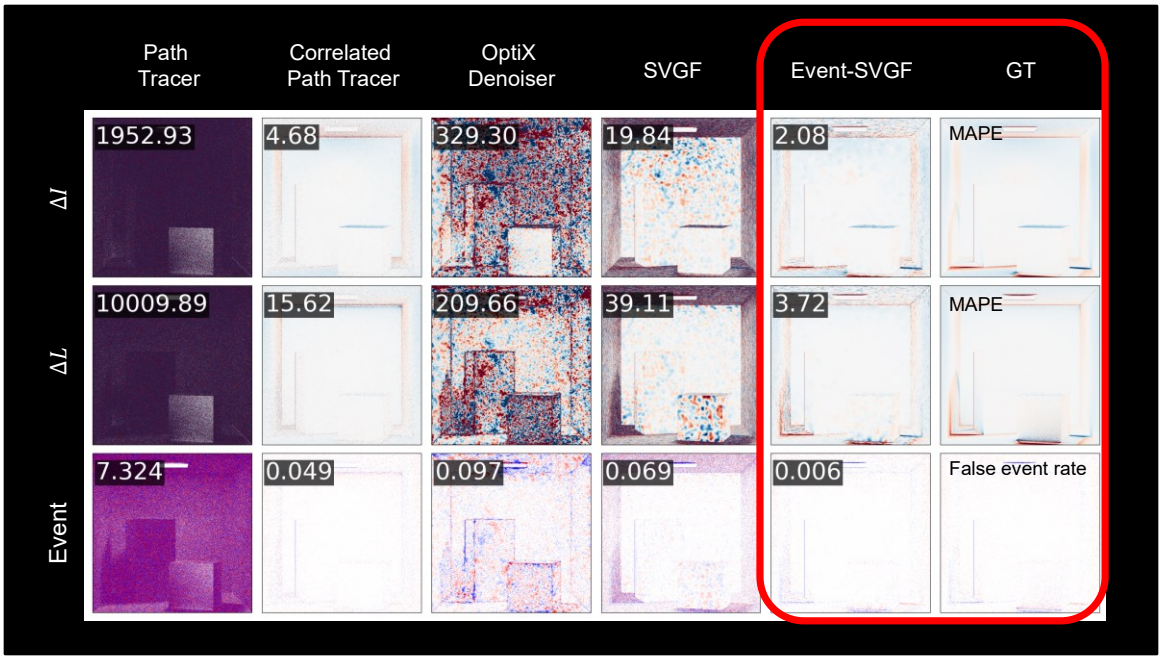
Here, we compare several different rendering methods under equal time budget. For naïve path tracer, it ends up with huge amount of noise.



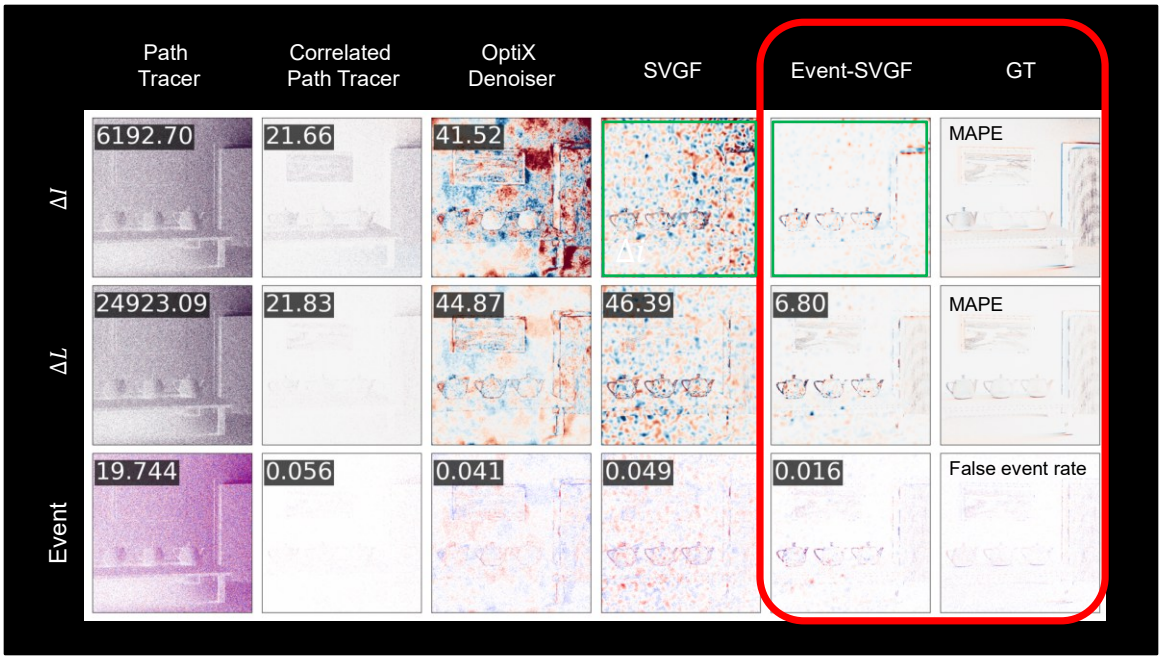
Correlated path tracer improves the result, but it often suffers from abrupt change between two samples which causes fireflies.



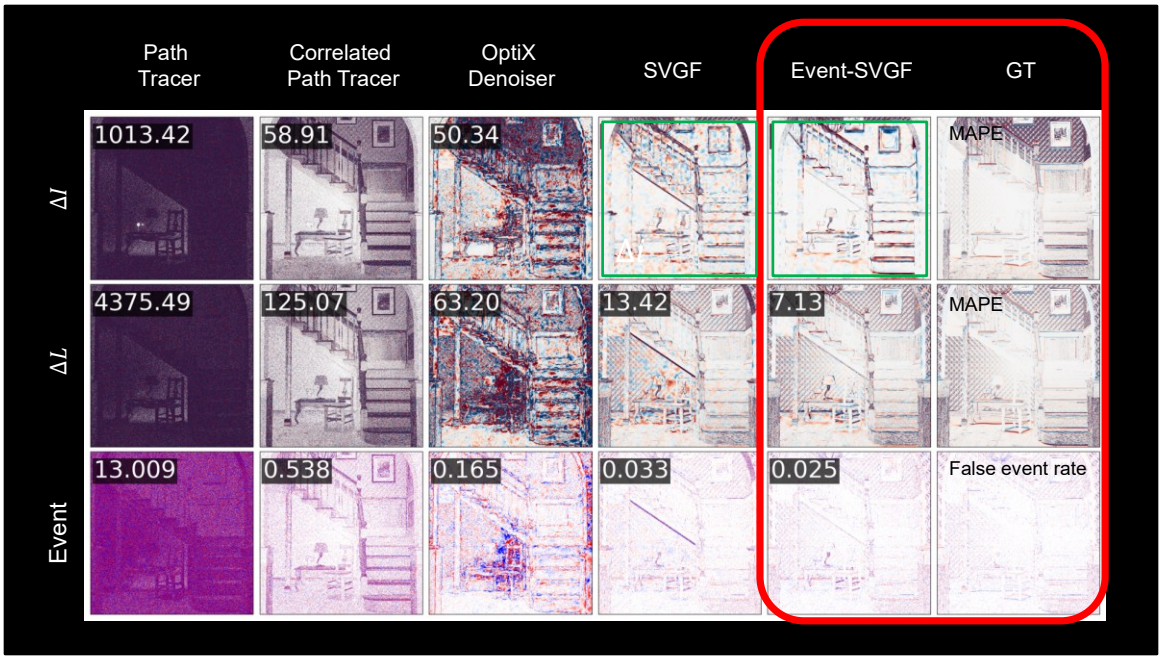
Primal RGB image denoising method gives satisfactory result in primal domain, but when we evaluate different, we can see lots of artifacts, due to their incremental independent sample update.



On the other hand, our proposed method shows much better result in terms of brightness difference error and suppressing false event.



We show result for other scene. We also show demodulated difference, excluding the affect of texture.

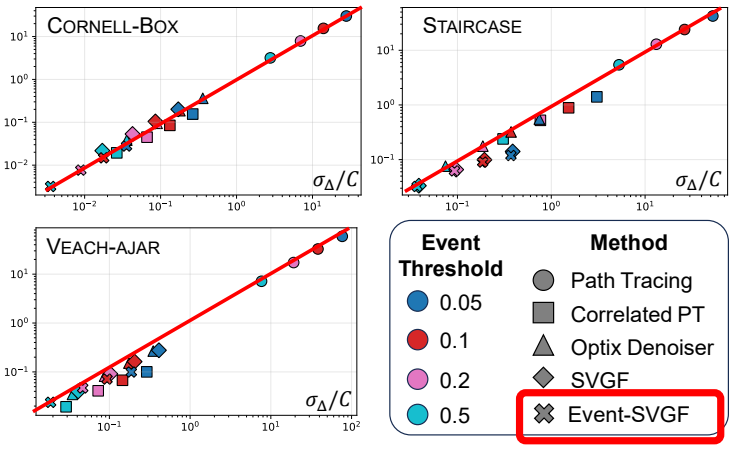

Path Tracer
Correlated Path Tracer
OptiX Denoiser
SVGF
Event-SVGF
GT
ΔI
ΔL
Event
1013.42
58.91
50.34
MAPE
4375.49
125.07
63.20
13.42
7.13
MAPE
13.009
0.538
0.165
0.033
0.025
False event rate

False Event Rate

$$(\text{false event rate}) \propto \frac{\sigma_{\Delta L}}{C}$$

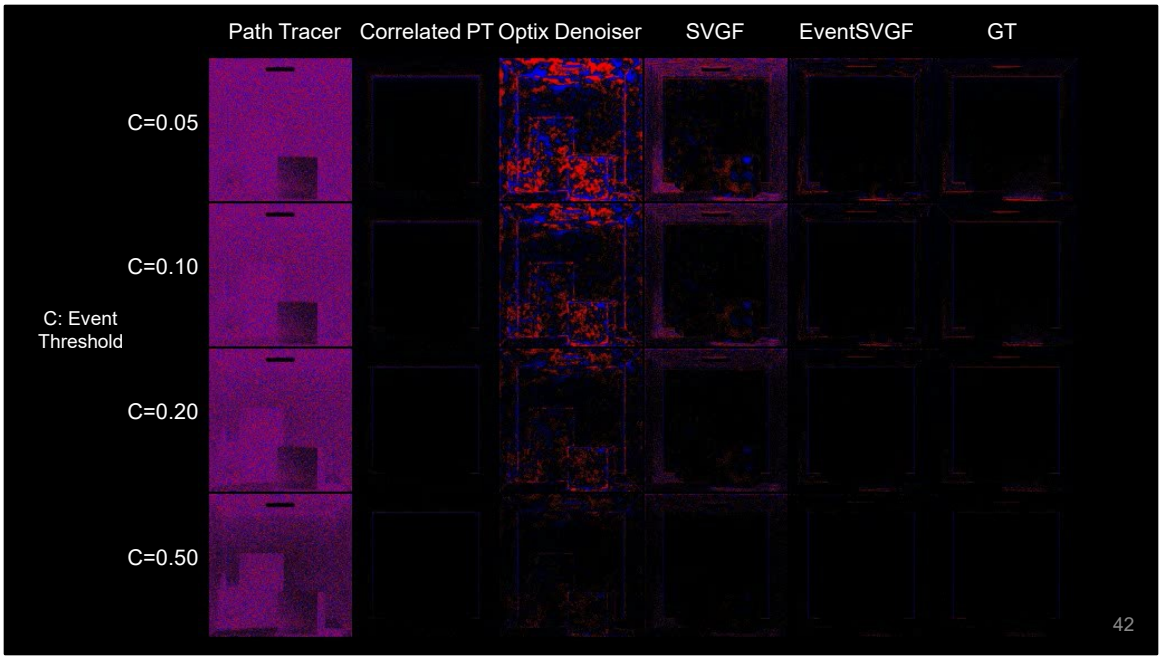
$\sigma_{\Delta L}$: noise of ΔL

C: threshold

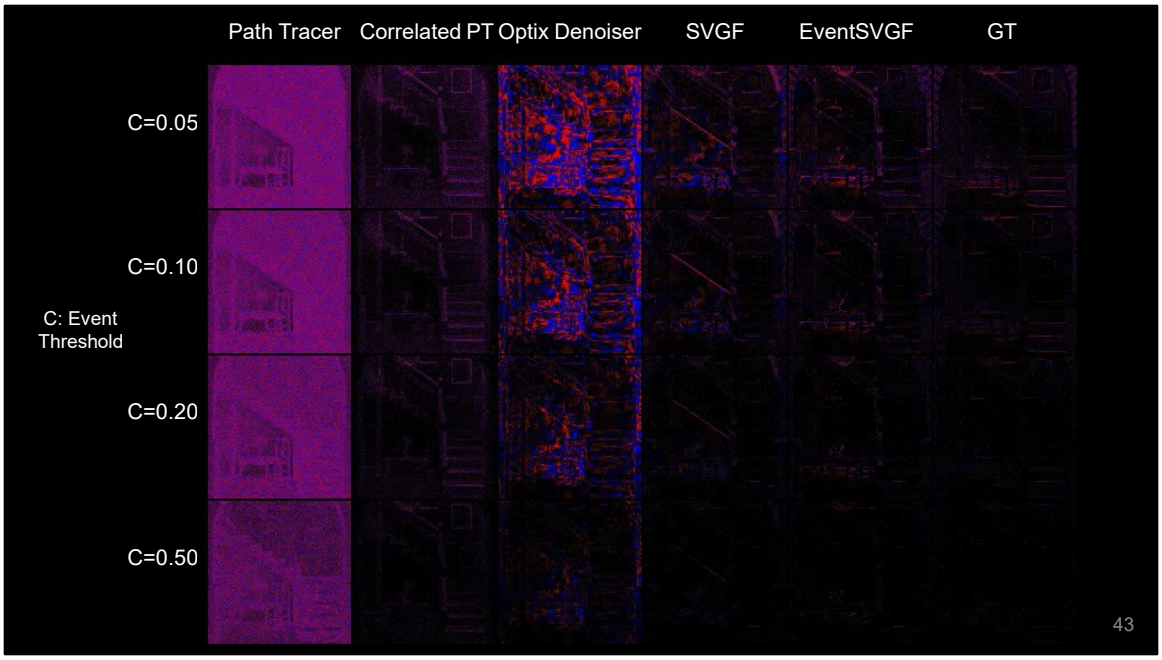


41

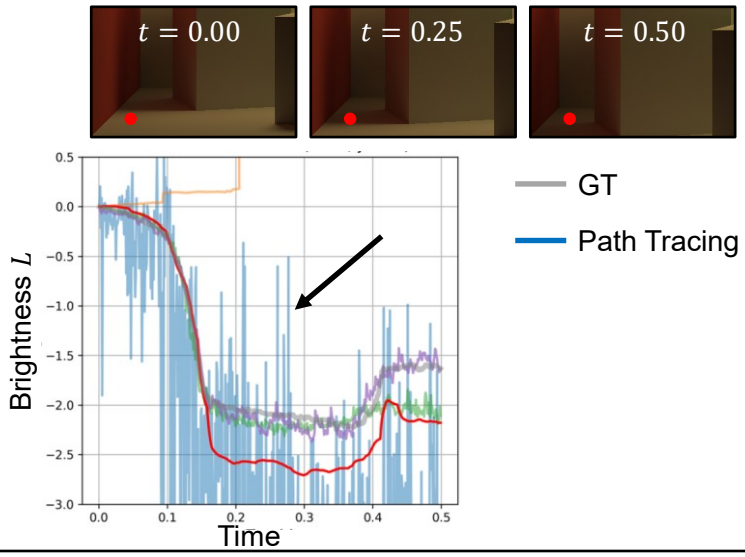
We also analyze the false event rate under different event thresholds. We observe a clear relationship between the false event rate and temporal stability relative to the threshold size, highlighting the importance of reducing noise in temporal differences for event camera simulation. Across all settings, our proposed EventSVGF consistently achieves the best performance.



We also show video result. As you can see, proposed method works most reliably.

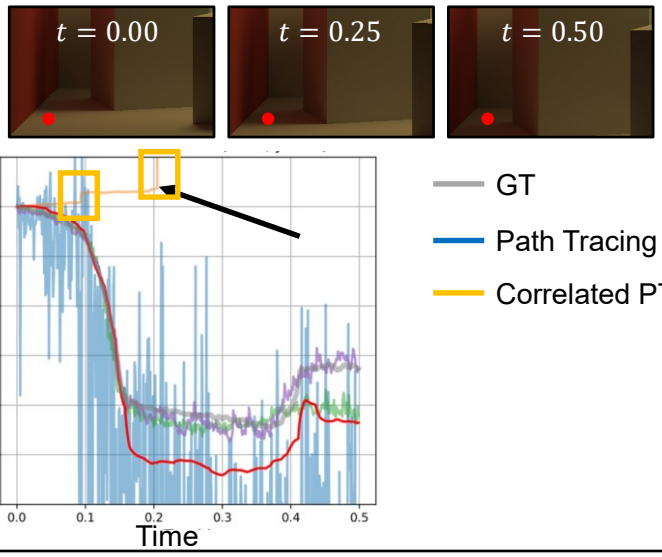

Path Tracer
Correlated PT
Optix Denoiser
SVGF
EventSVGF
GT
C=0.05
C=0.10
C=0.20
C=0.50
C: Event Threshold
43

Temporal and Frequency-Domain Analysis



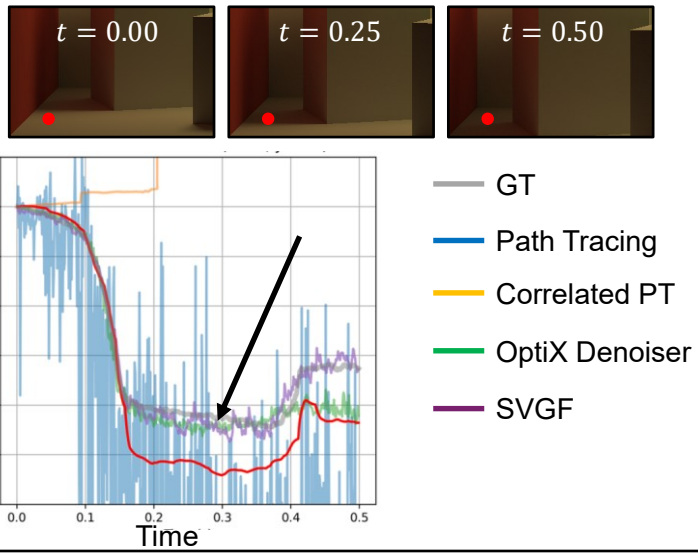
We then perform temporal analysis by plotting brightness over time at certain pixel. First, naïve path tracing ends up with extreme temporal noise.

Temporal and Frequency-Domain Analysis



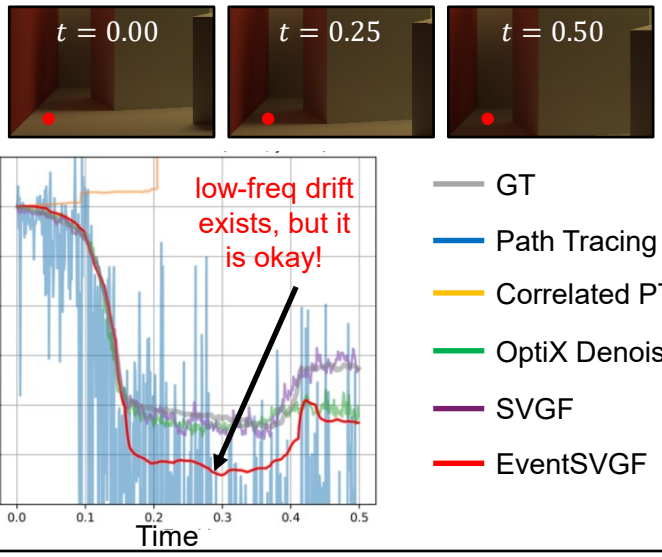
Correlated path tracing improves it, but suffers from some abrupt changes.

Temporal and Frequency-Domain Analysis



Primal domain denoiser gives move accurate brightness values, but it is still temporally noisy.

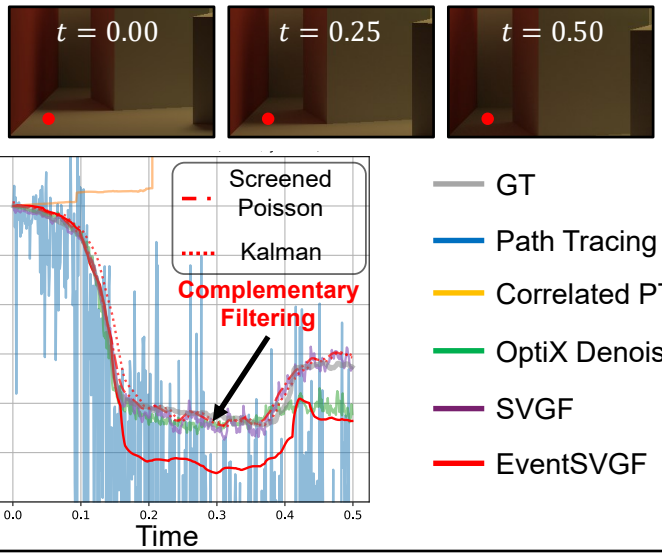
Temporal and Frequency-Domain Analysis



47

On the other hand, our event svgf achieves the most temporally stable result. Of course, there clearly exists the low frequency drift, but as I said before, real sensor also suffers from low-frequency drift and low-frequency accuracy was not in our interest.

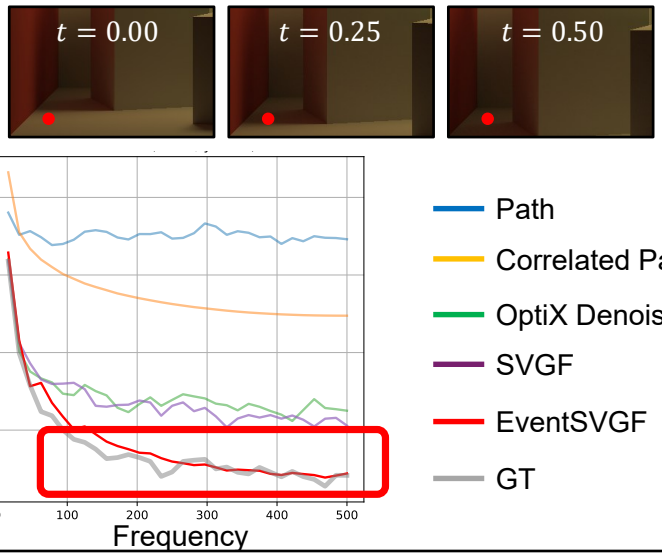
Temporal and Frequency-Domain Analysis



48

In practice, this low-frequency accuracy can be improved via complementary filtering with RGB sensor, with algorithm such as screened Poisson of Kalman filter. But accuracy in high-frequency region of our algorithm is still helpful for such filtering.

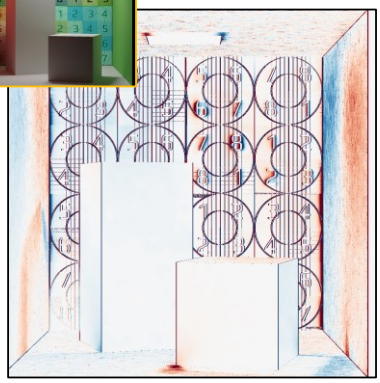
Temporal and Frequency-Domain Analysis



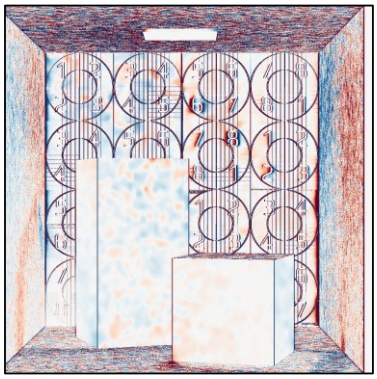
We also show frequency analysis of the brightness signal. As you can see, the proposed method achieves the best result in high frequency noise.

Ablation Study

Full Model



w/o Correlated Sampling

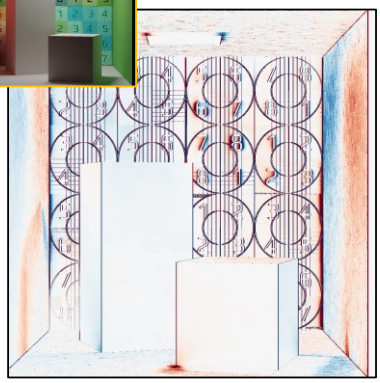


50

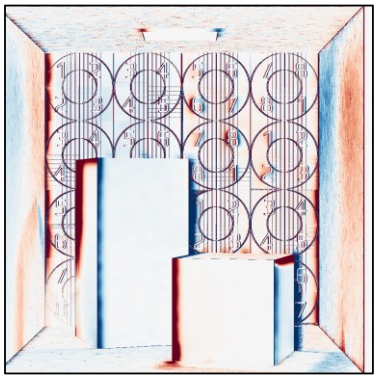
To show importance of each components in denoisng pipeline, we perform abalation study. First we disable correlated sampling and this gives increased noise with some ripple like artifact.

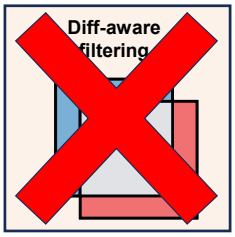
Ablation Study

Full Model



w/o Diff-aware Filtering

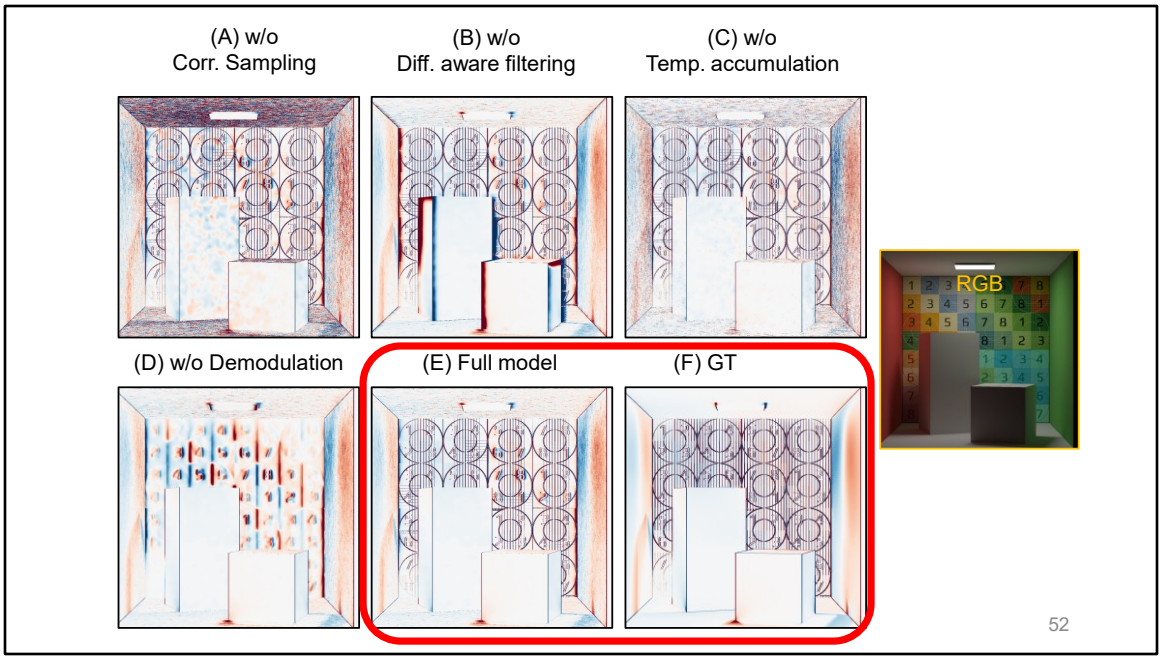




$w_{\text{curr}} \times w_{\text{prev}}$

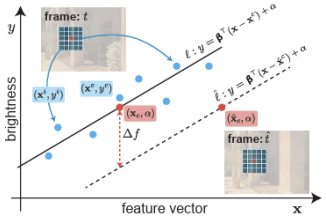
51

Then we disable difference-aware filtering which gives blur over edges.

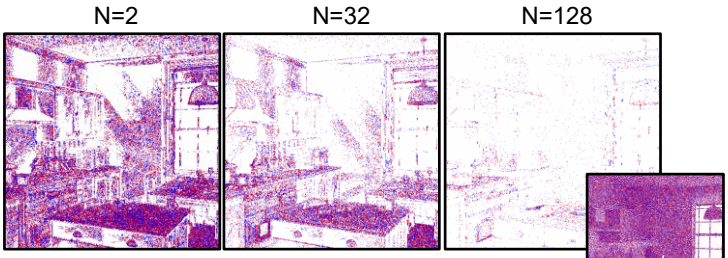


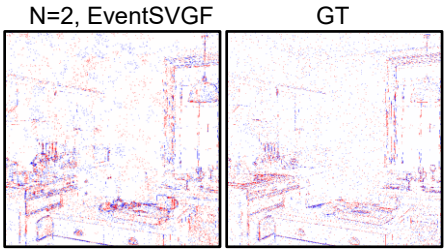
We also show result without temporal accumulation or demodulation. Overall, the proposed full model gives the best result.

Comparison with Adaptive Filtering [Tsuji et al. 2023]



Adaptive Filtering
[Tsuji et al. 2023]





53

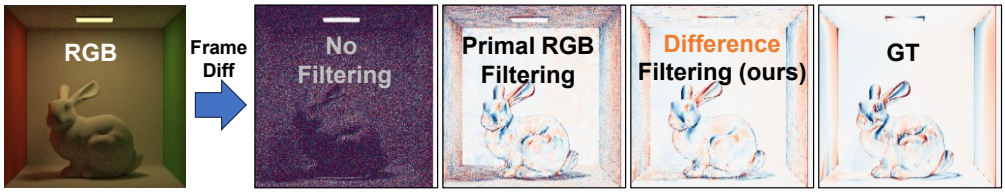
Finally, We also compare with one of previous event rendering method with adaptive filtering. We found that their method clearly gives better result than naïve path tracer, but still requires lots of samples to converge. On the other hand, our method gives converged result only using 2spp thanks to correlated sampling and our tailored difference-aware filtering.

Conclusion

Conclusion & Future Work

Thank you!

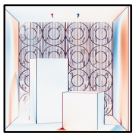
Difference denoiser for Event-camera simulation



Imaging Side

- Hardware noise
- Downstream Applications

Rendering Side



- Better shift-mapping / antialiasing
- Better Denoiser (e.g., REBLUR)
- Connection to Differentiable rendering

55

In conclusion, we proposed a difference-aware denoiser for efficient low-spp event camera simulation. There are several promising directions for future work. For imaging side, modeling hardware noise or developing downstream applications would be interesting, and for rendering side, using improved shift mapping, anti-aliasing, denoising or even searching connections to differentiable rendering can be exciting future works.

Backup Slides