Event-based Robot Vision

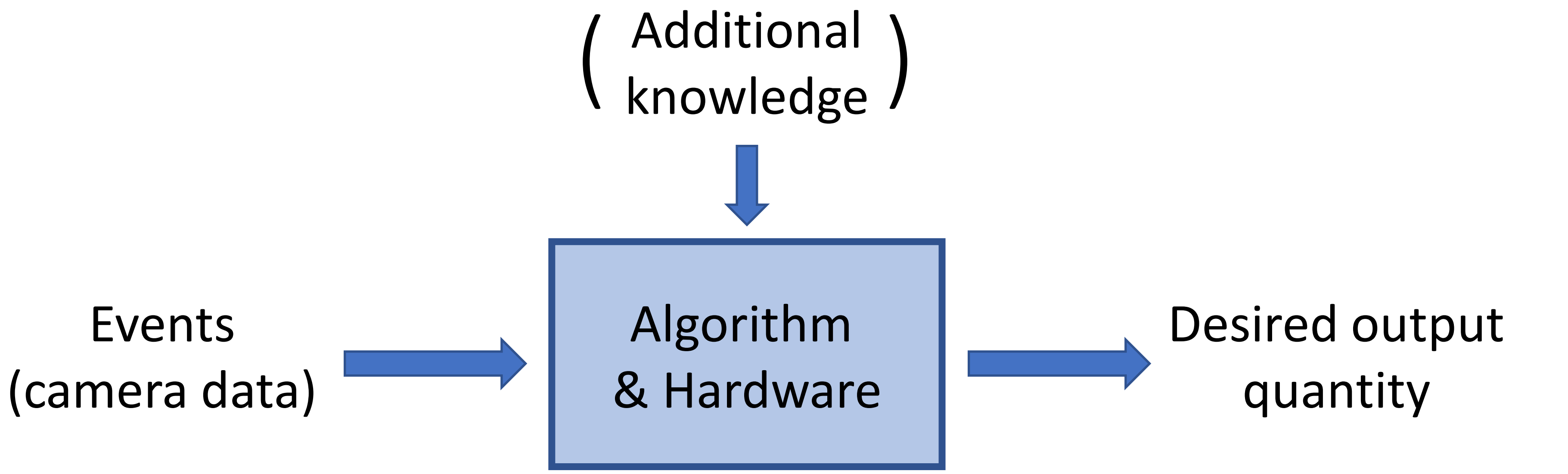
Prof. Dr. Guillermo Gallego
Chair: Robotic Interactive Perception

guillermo.gallego@tu-berlin.de

http://www.guillermogallego.es

How to Process the Events?

Overview

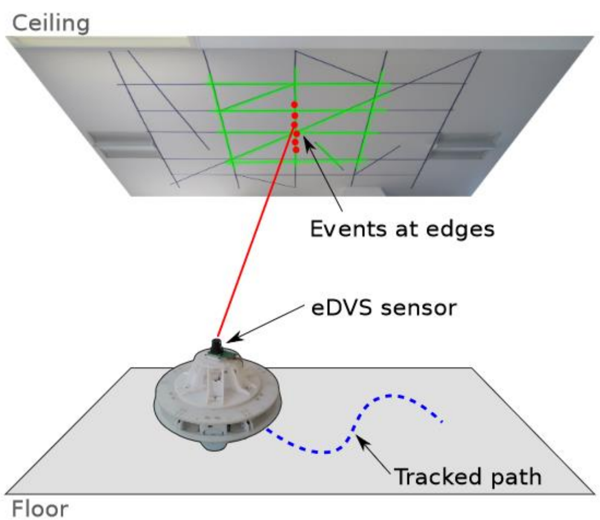


(Additional knowledge)
Events (camera data)
Algorithm & Hardware
Desired output quantity

The New Paradigm

- Process **one event at a time**
- How much information does an event carry?
- The vision system / algorithm needs to have **additional knowledge** (either from external information or built from past events) to assimilate (fuse with) each event.

- **Advantages**:
  - **Minimum latency**
  - Asynchronous & sparse
- **Disadvantages**:
  - Updating the system on a per-event basis may be expensive, depending on the task



Weikersdorfer et al. ICVS 2013

The Paradigm: Event-by-event Processing

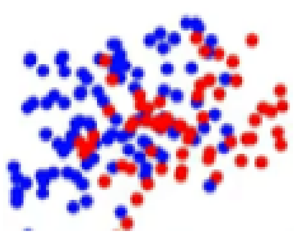
- **Case Study: Reconstruction of Intensity Image**

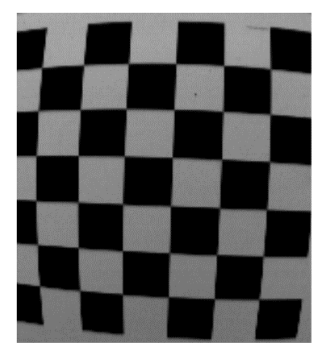
Additional knowledge:
Internal "state" $s(\boldsymbol{x}, t)$
built from past events

Event
$e_k = (\boldsymbol{x}_k, t_k, p_k)$

Algorithm
(& Hardware)

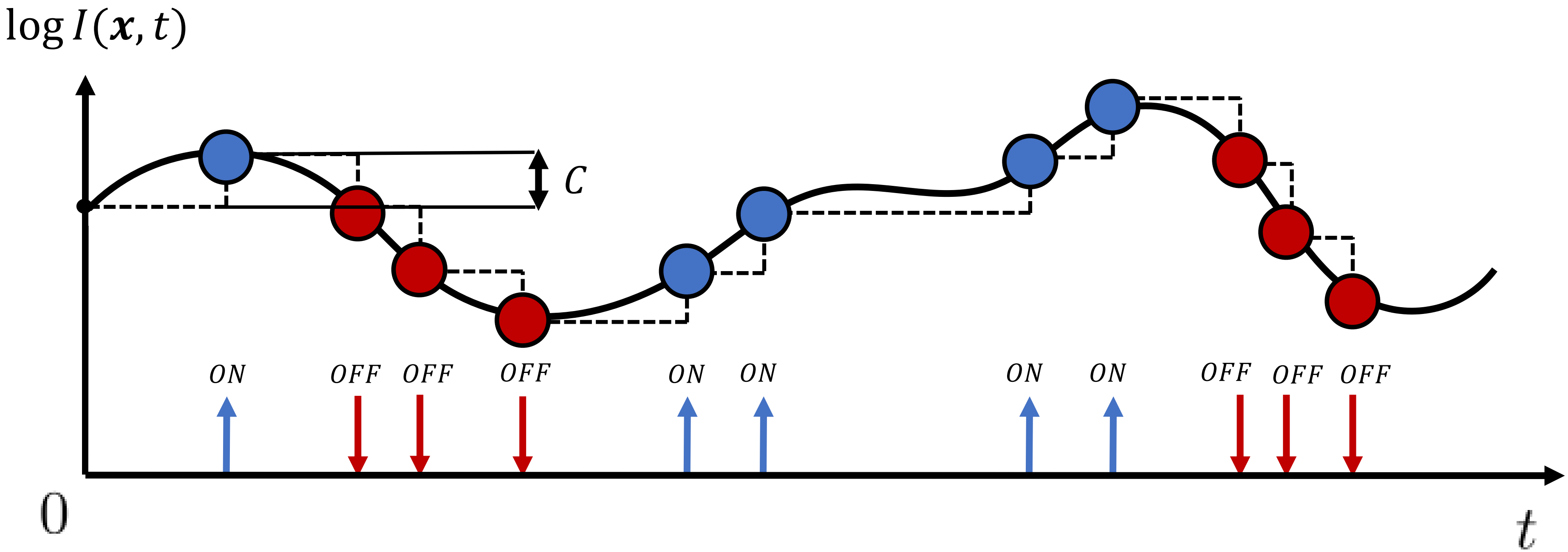
Reconstructed
intensity image
$I(\boldsymbol{x}, t_k)$





Recall the Event Generation Model

$$\log I(\boldsymbol{x}, t) - \log I(\boldsymbol{x}, t - \Delta t) = \pm C$$

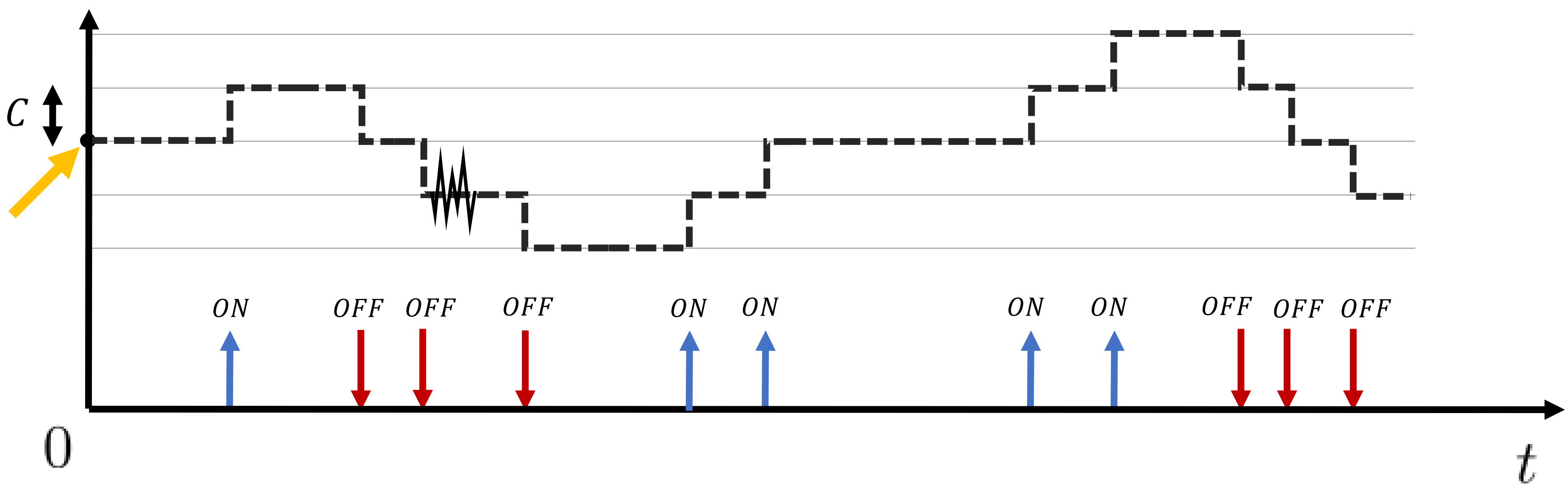


Light (log *I*) has been transduced into asynchronous events…

Given the events, can we recover the absolute intensity?

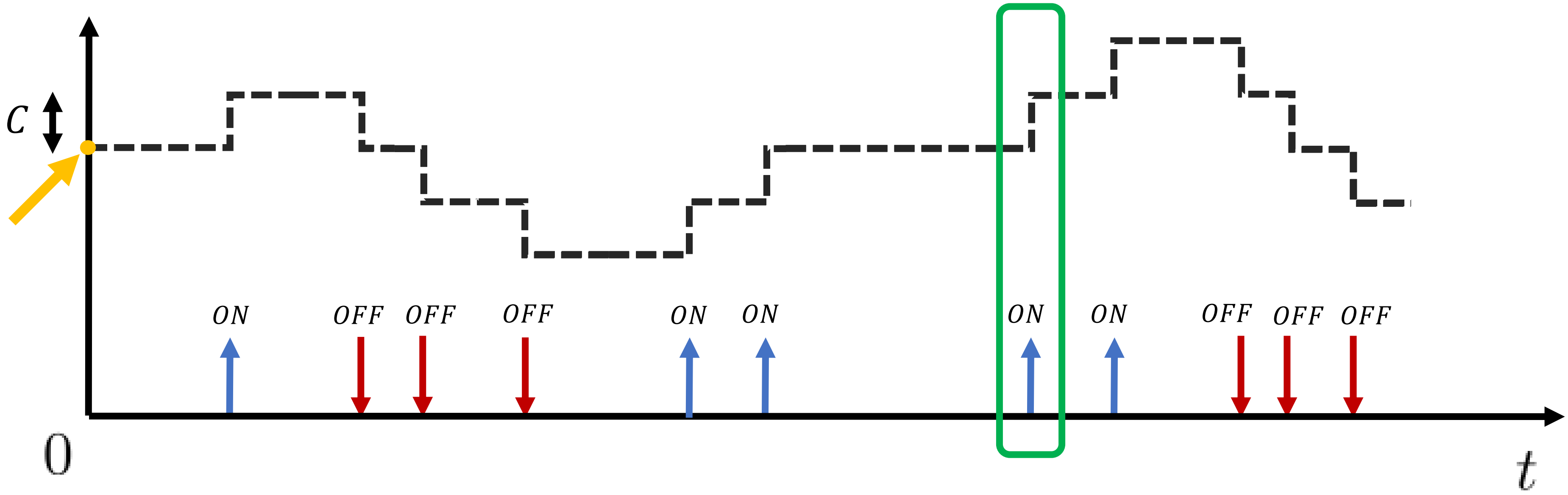
Let us try to recover the pixel's intensity

- Events represent **intensity changes** ⇒ **Integration** should provide absolute intensity



- The recovered signal approximates the original one
- And we cannot see oscillations within the step C (quantization error)
- Additionally, the offset (at $t = 0$) is typically unknown…

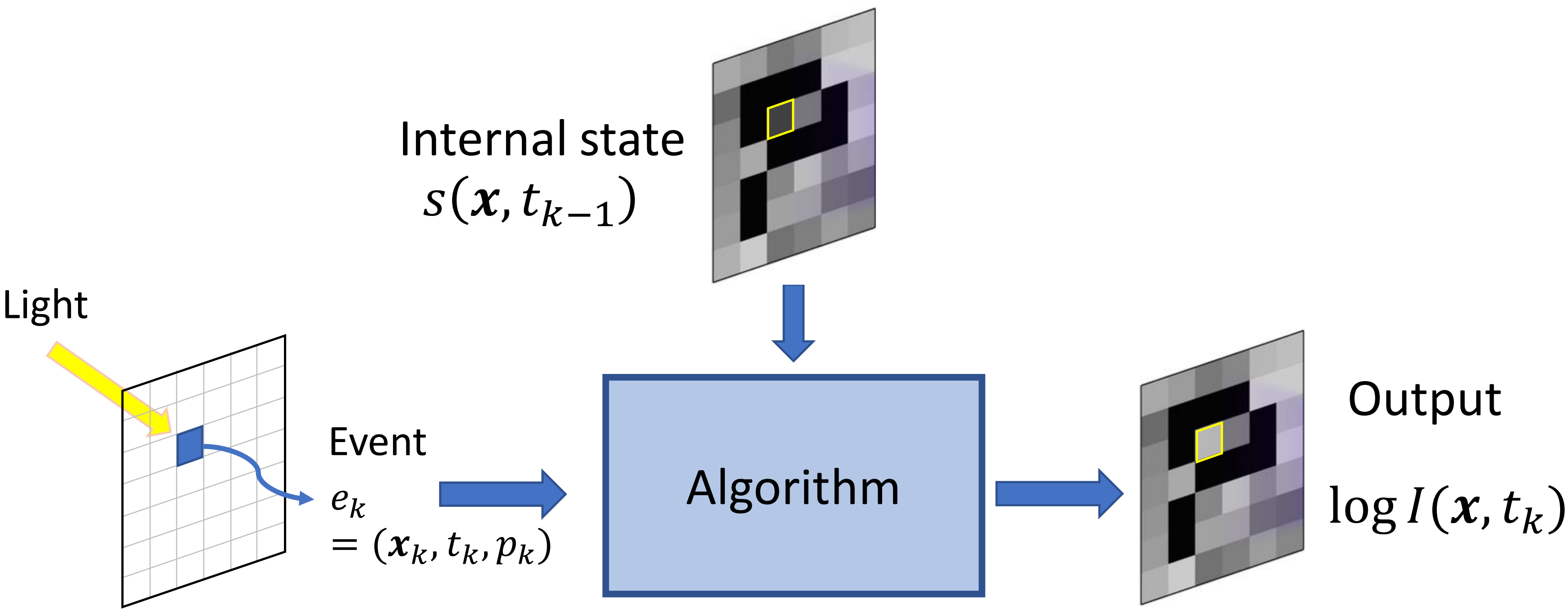
Let us try to recover the pixel's intensity



- Typically the offset (at $t = 0$) is unknown…
- That's what happens at one pixel… and we need offsets on all image pixels to make a good (coherent) image

The paradigm: Event-by-event Processing

- **Case Study: Reconstruction of Intensity Image**



Simplest rule: update one pixel

$$\log I(\boldsymbol{x}_k, t_k) = s(\boldsymbol{x}_k, t_{k-1}) + p_k C$$

Event Integration

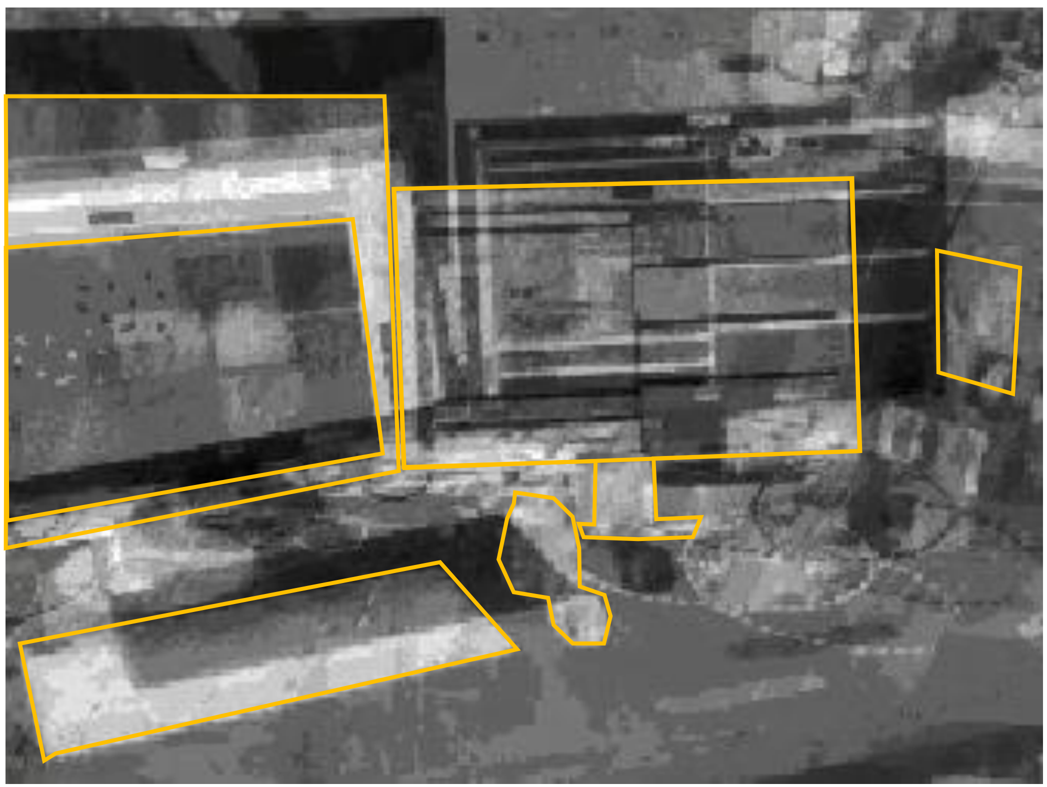
$$L(\boldsymbol{x}, t\ ) = L(\boldsymbol{x}, 0\ ) + \Delta L(\boldsymbol{x}; 0, t)$$

Log-intensity at time t

Log-intensity at time t=0

Increment intensity in [0,t] is $\int_0^t \frac{dL(v)}{dt} dv \approx \sum_k \Delta L_k$

$\tilde{L}(\boldsymbol{x}_k, t_k) = s(\boldsymbol{x}_k, t_{k-1}) + p_k$, starting from s=0



Direct integration of events, without starting from $L(\boldsymbol{x}, 0)$, **cannot recover absolute intensity**,
→Produces incremental intensity ΔL

Notice the **missing offset $L(x, 0)$**

Event Integration

- Estimated intensity. Intuition:

$$L(t) = L(0) + \underbrace{\int_0^t \frac{dL}{dt}(\tau)d\tau}_{\text{Increment } \Delta L := L(t) - L(0)}$$

$$\log \underbrace{\hat{I}(\boldsymbol{x}, t)}_{\text{pixel}} = \underbrace{\log I(\boldsymbol{x}, 0)}_{\substack{\text{Intensity at } t=0 \\ \text{(offset)}}} + \underbrace{\sum_{0<t_k\leq t} p_k\, C\, \delta(\boldsymbol{x} - \boldsymbol{x}_k)\delta(t - t_k)}_{\text{Intensity increment } \Delta \log I}$$

pixel

polarity

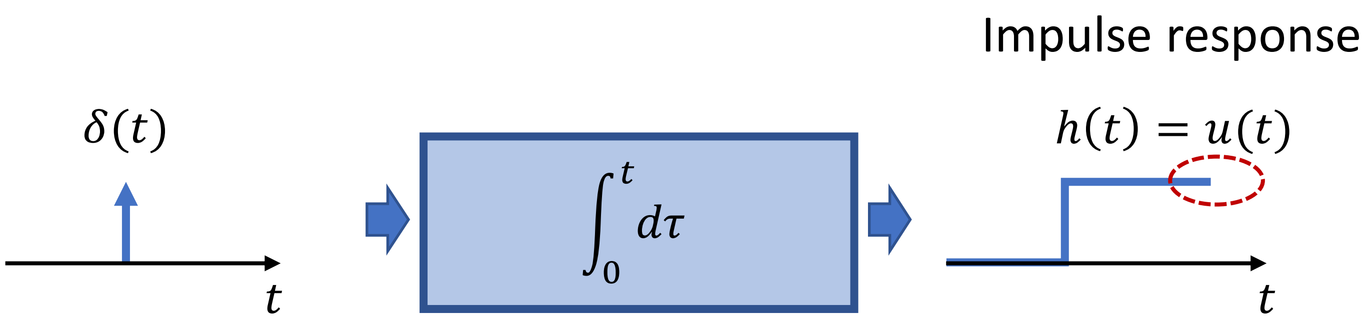
Kronecker delta (discrete variable)

Dirac delta (continuous variable)

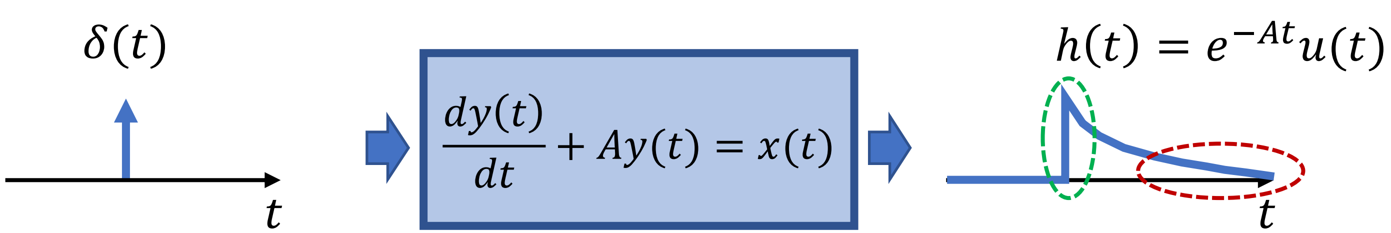
Mueggler et al., *The Event Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM*, IJRR 2017.

A Replacement for the Integrator?

- The (ordinary) integrator has a very long effect:



- Replace the integrator with a **leaky** one, so that its effect decays over time (i.e., it "forgets"):



Leaky integrator. A first-order filter

A so-called leaky integrator is a first-order filter with feedback. Let's find its transfer function, assuming that the input is $x(t)$ and the output $y(t)$:

$$\frac{dy(t)}{dt} + Ay(t) = x(t)$$

$$\mathcal{L}\left\{\frac{dy(t)}{dt} + Ay(t)\right\} = \mathcal{L}\left\{x(t)\right\}$$

where $\mathcal{L}$ denotes application of the Laplace transform. Moving forward:

$$sY(s) + AY(s) = X(s)$$

$$H(s) = \frac{Y(s)}{X(s)} = \frac{1}{s+A}$$

(taking advantage of the Laplace transform's property that $\frac{dy(t)}{dt} \Leftrightarrow sY(s)$, assuming that $y(0) = 0$).

This system, with transfer function $H(s)$, has a single pole at $s = -A$. Remember that its frequency response at frequency ω can be found by letting $s = j\omega$:

$$H(j\omega) = \frac{1}{j\omega + A}$$

To get a rough view of this response, first let $\omega \to 0$:

$$\lim_{\omega \to 0} H(\omega) = \frac{1}{A}$$

So the system's DC gain is inversely proportional to the feedback factor A. Next, let $w \to \infty$:

$$\lim_{\omega \to \infty} H(\omega) = 0$$

The system's frequency response therefore goes to zero for high frequencies. **This follows the rough prototype of a lowpass filter.** To answer your other question with respect to its time constant, it's worth checking out the system's time-domain response. Its impulse response can be found by inverse-transforming the transfer function:

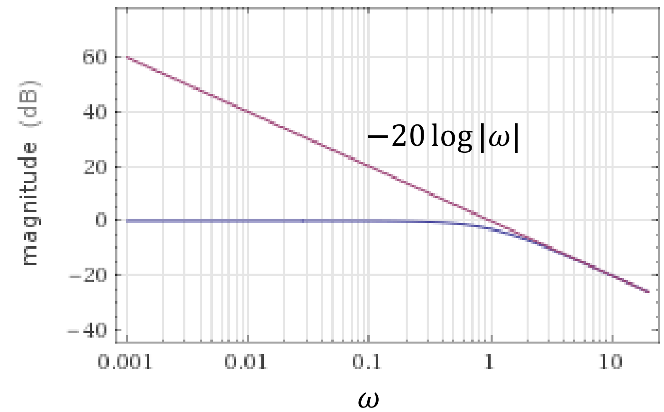
$$H(s) = \frac{1}{s+A} \Leftrightarrow e^{-At}u(t) = h(t)$$

where $u(t)$ is the Heaviside step function. This is a very common transform that can often be found in tables of Laplace transforms. This impulse response is an exponential decay function, which is usually written in the following format:

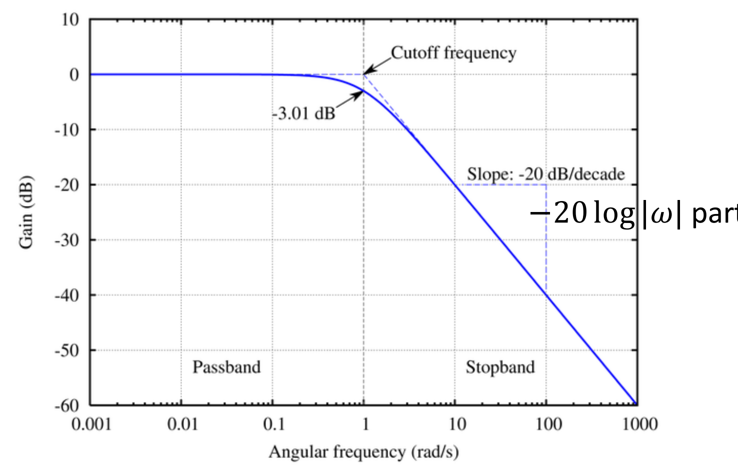
$$h(t) = e^{-\frac{t}{\tau}}u(t)$$

where τ is defined to be the function's time constant. **So, in your example, the system's time constant is $\tau = \frac{1}{A}$.**

Ordinary integration $\int_0^t f(v)dv \xleftrightarrow{Laplace} \frac{F(s)}{s}$ has a Fourier response $\frac{1}{s} = \frac{1}{j\omega}$ that blows up at $\omega = 0$.

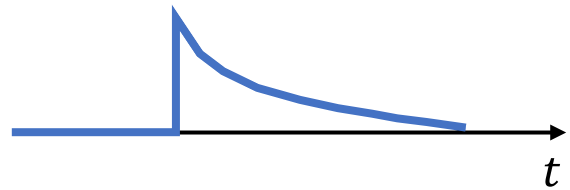


Leaky integration smooths off the infinite value at $\omega = 0$. Thus the amplitude spectrum $20\log\frac{1}{|\omega|} = -20\log|\omega|$, except that it is not ∞ at $\omega = 0$.

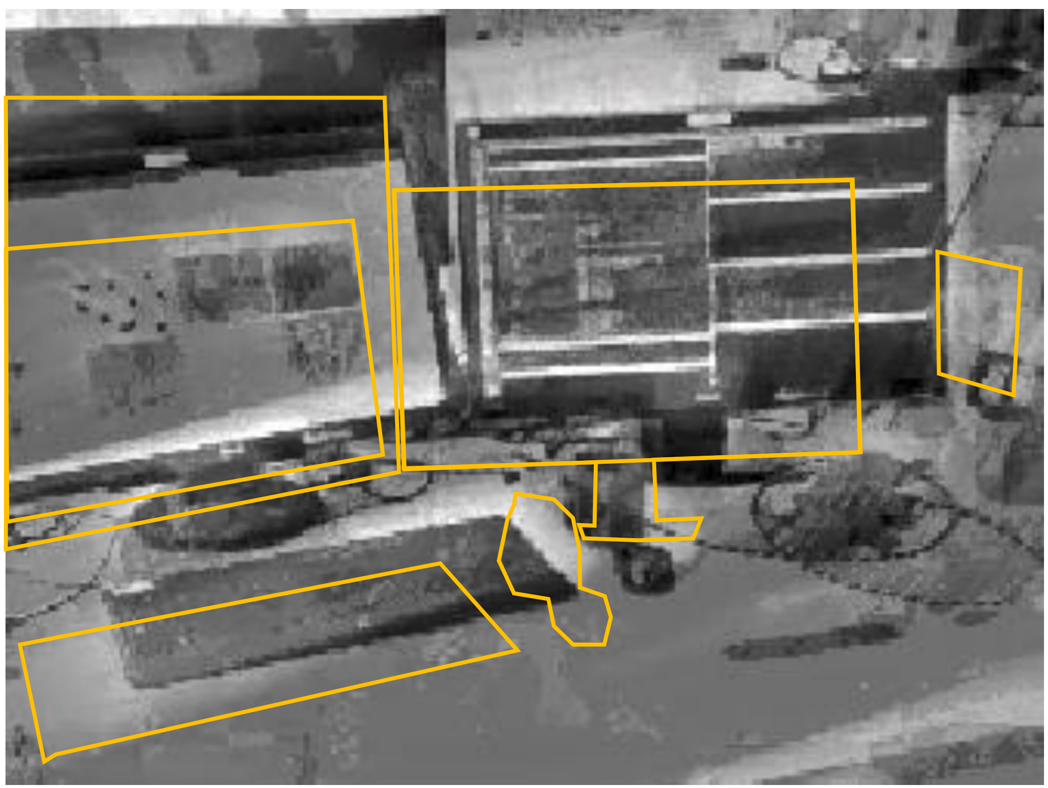


$\alpha = 0.3$

Leaky integration



$$\tilde{L}(\boldsymbol{x}_k, t_k) = e^{-\alpha \Delta t_k} s(\boldsymbol{x}_k, t_{k-1}) + p_k$$, starting from s=0



Leaky integration of events can recover ≈ absolute intensity, **even without knowing the offset $L(x, 0)$**!

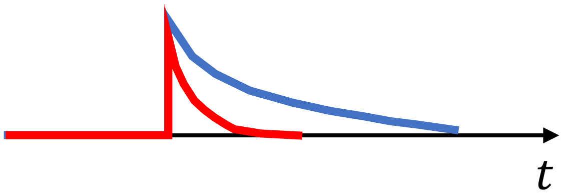
Isn't it magic?

Temporal leakage "forgets" the initial intensity.

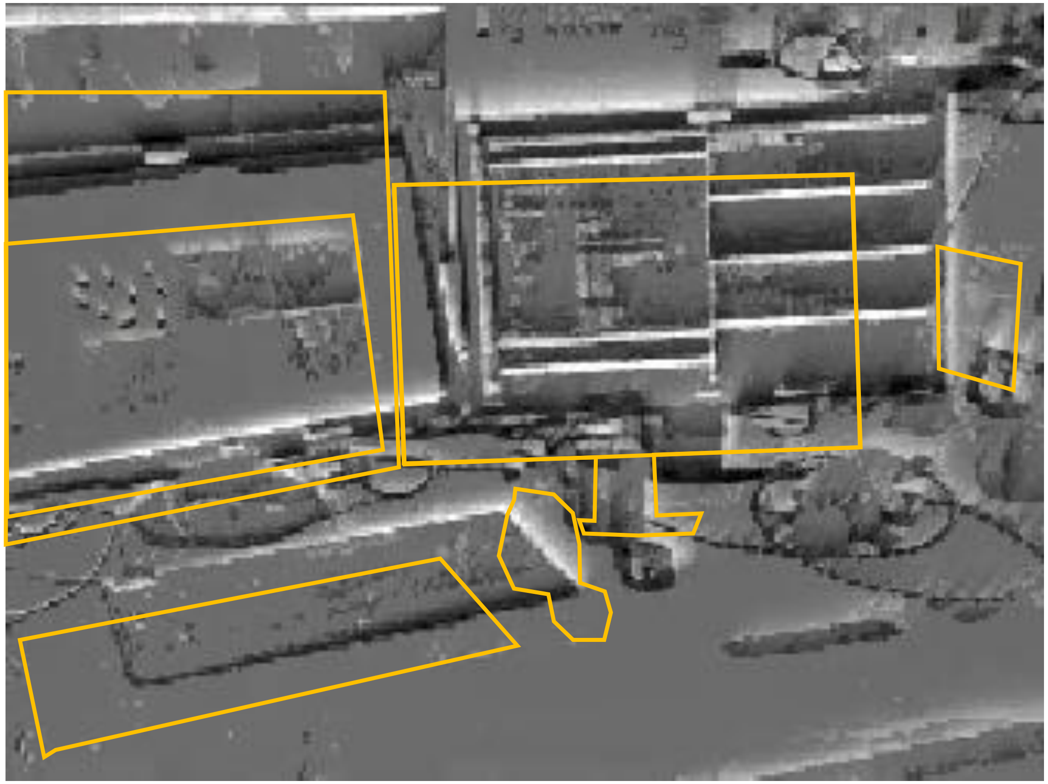
Spatial filtering is **not** needed.

Leaky integration. Short decay

$\alpha = 2$



$\tilde{L}(\boldsymbol{x}_k, t_k) = e^{-\alpha \Delta t_k} s(\boldsymbol{x}_k, t_{k-1}) + p_k$, starting from *s*=0

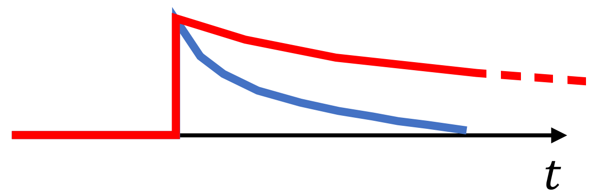


Too much leakage (forgetting quickly)
Faster decay → not integrating well

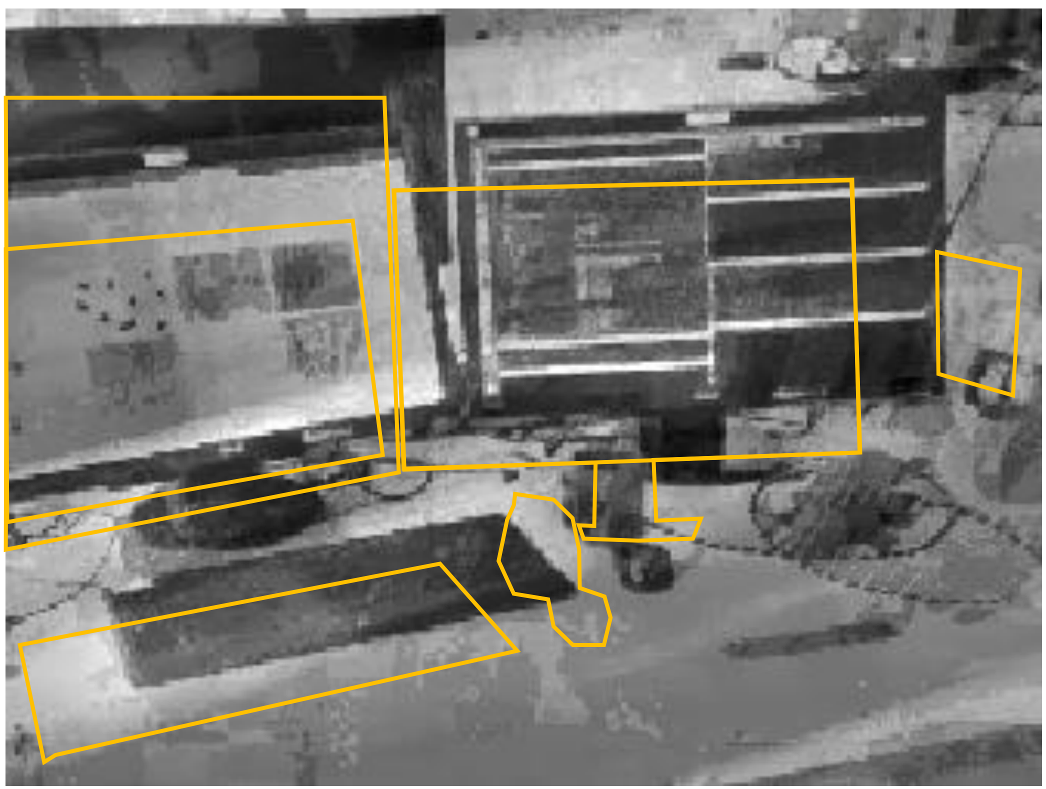
(even if offset $L(x, 0)$ is washed out)

$\alpha = 0.1$

Leaky integration. Long decay



$\tilde{L}(\boldsymbol{x}_k, t_k) = e^{-\alpha \Delta t_k} s(\boldsymbol{x}_k, t_{k-1}) + p_k$, starting from s=0



Too little leakage (forgetting slowly)
Slower decay → integration artefacts

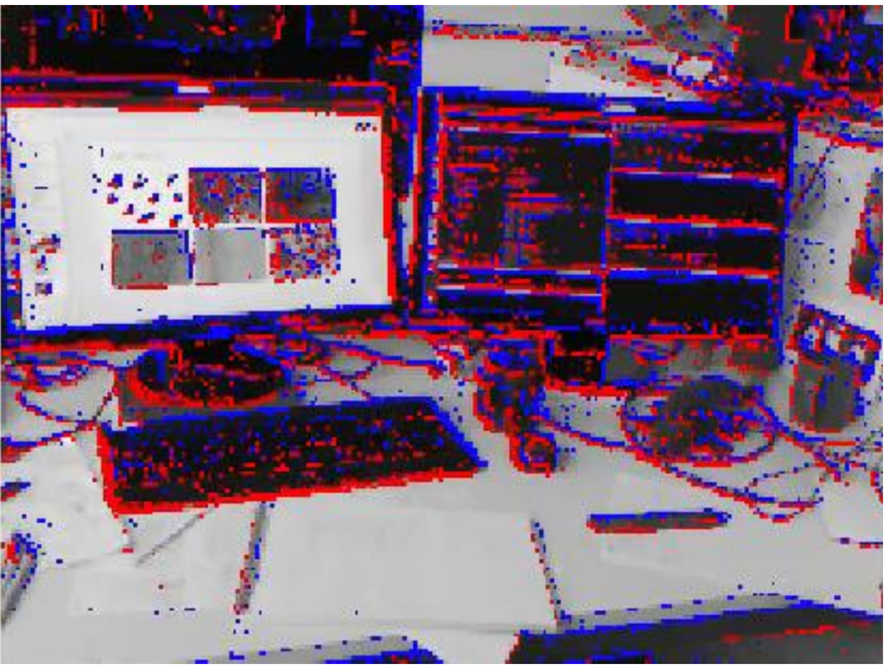
(even if offset $L(x, 0)$ is washed out)

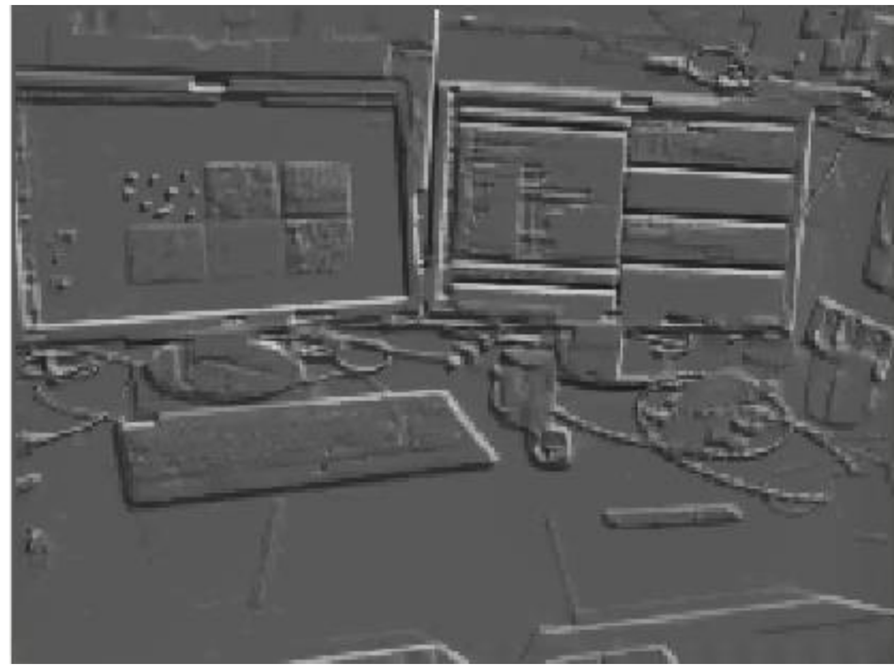
Noise-Free, Simulated Events

Desktop Scene

$\alpha = 0$

Direct (ordinary) Integration





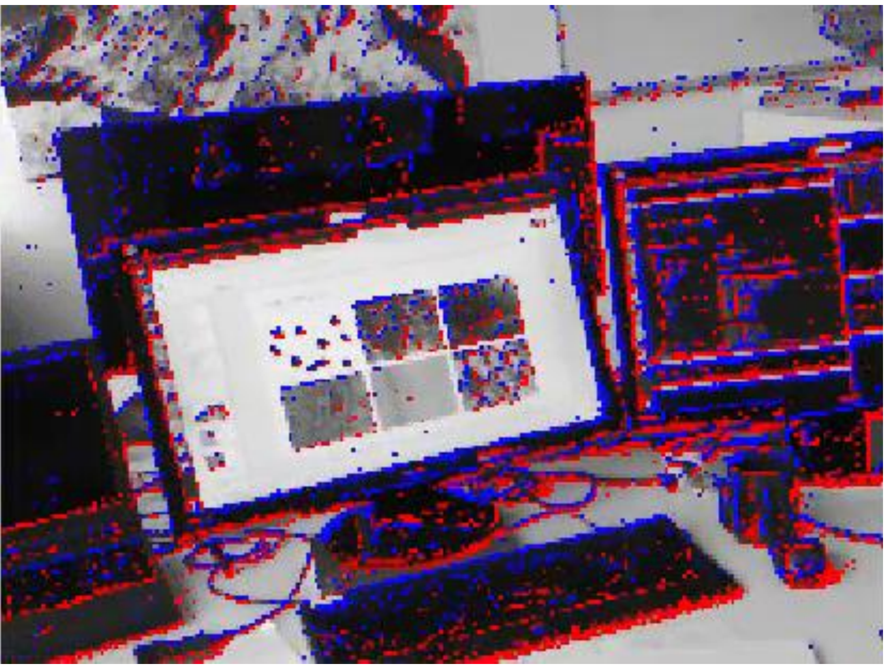
Input: **events** (ON, OFF)
Grayscale frames are just used for visualization.

$$\tilde{L}(\boldsymbol{x}_k, t_k) = e^{-\alpha \Delta t_k} s(\boldsymbol{x}_k, t_{k-1}) + p_k$$

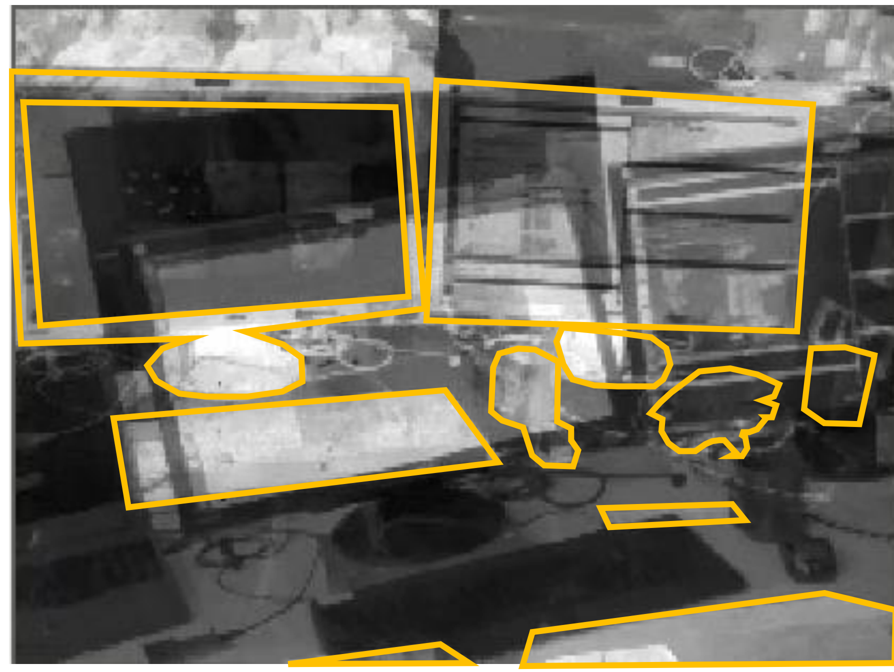
starting from *s*=0

$\alpha = 0$

Direct (ordinary) Integration



Input: **events** (ON, OFF)
Grayscale frames are just used for visualization.

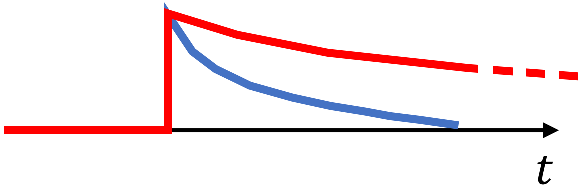


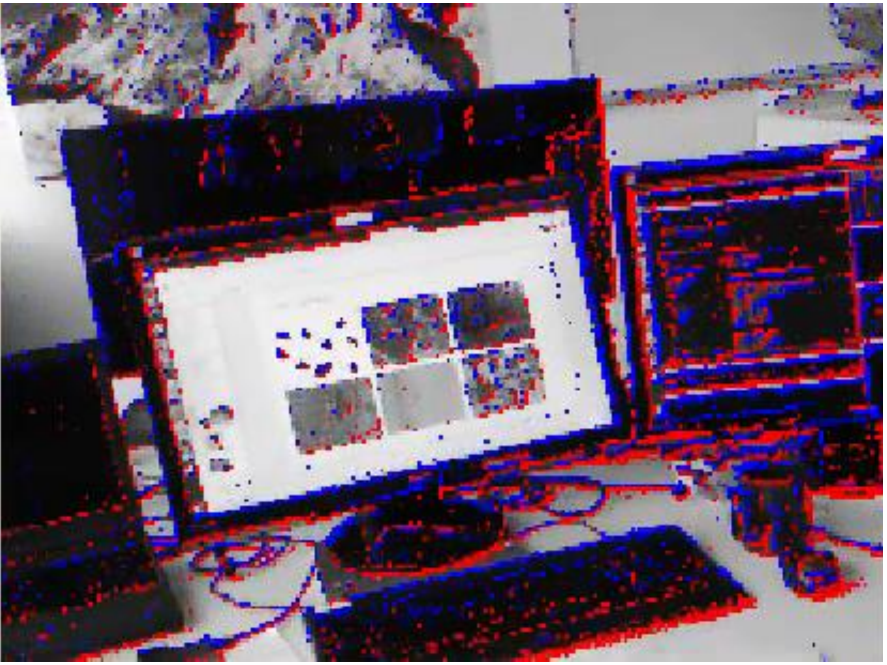
$$\tilde{L}(\boldsymbol{x}_k, t_k) = e^{-\alpha \Delta t_k} s(\boldsymbol{x}_k, t_{k-1}) + p_k$$

starting from s=0

$\alpha = 0.1$

Leaky integration. Long decay





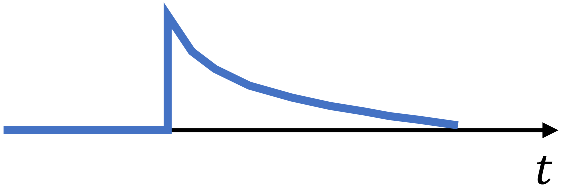
Input: **events** (ON, OFF)
Grayscale frames are just used for visualization.

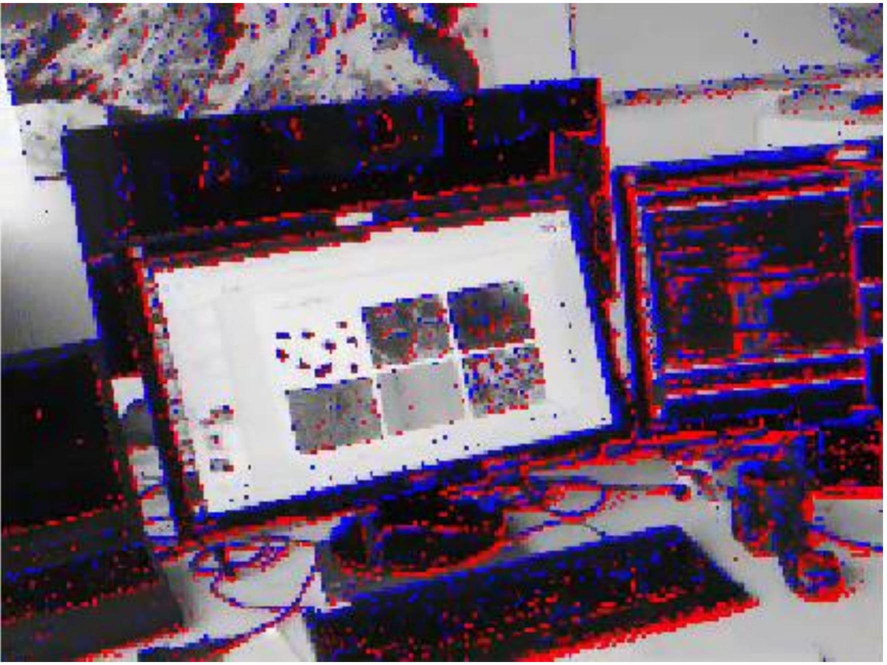
$$\tilde{L}(\boldsymbol{x}_k, t_k) = e^{-\alpha \Delta t_k} s(\boldsymbol{x}_k, t_{k-1}) + p_k$$

starting from *s*=0

Leaky integration

$\alpha = 0.3$





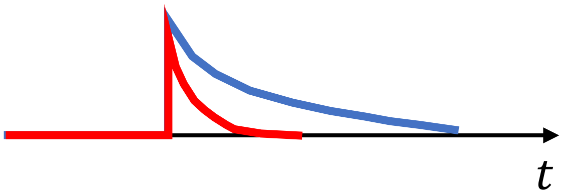
Input: **events** (ON, OFF)
Grayscale frames are just used for visualization.

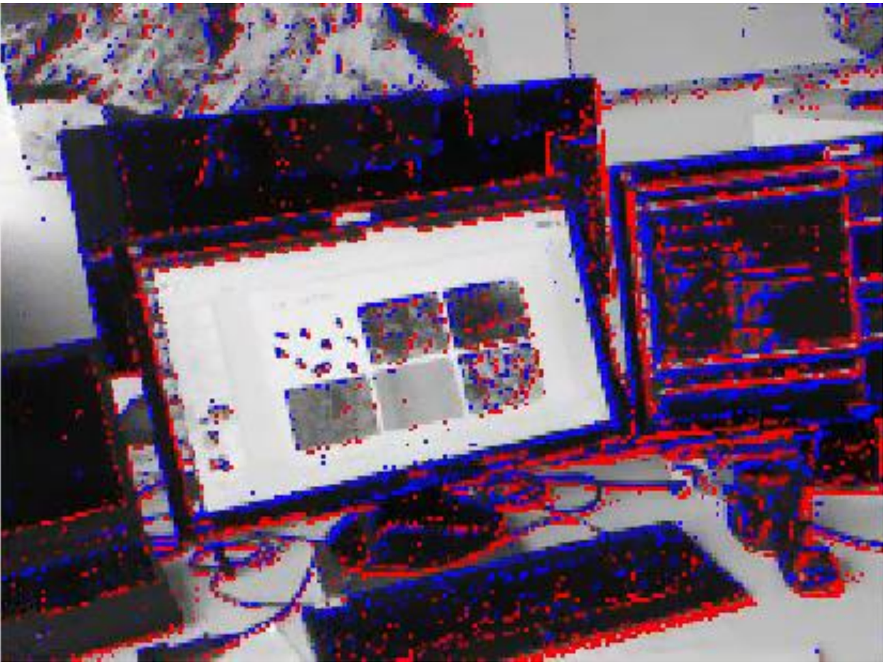
$$\tilde{L}(\boldsymbol{x}_k, t_k) = e^{-\alpha \Delta t_k} s(\boldsymbol{x}_k, t_{k-1}) + p_k$$

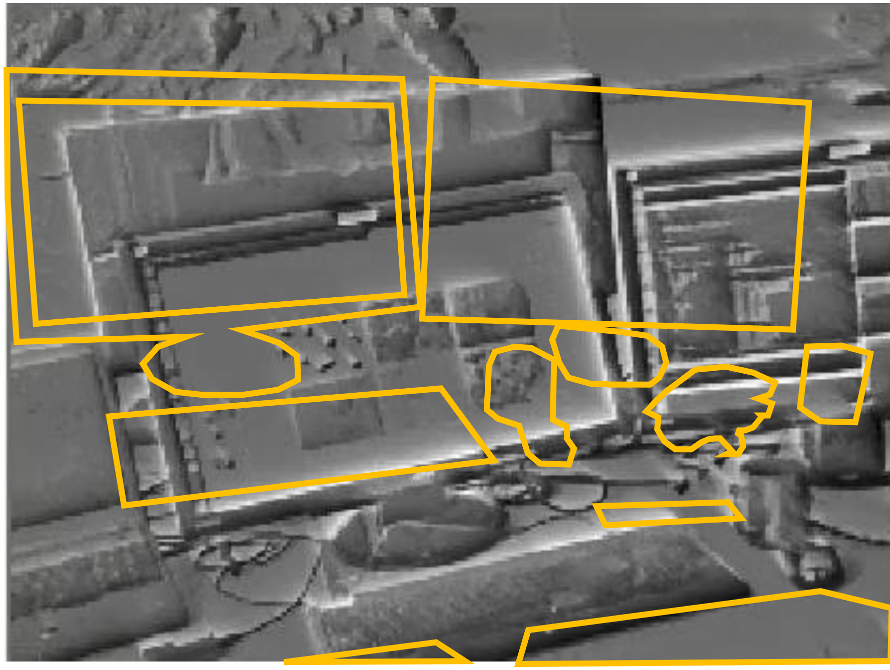
starting from *s*=0

$\alpha = 2$

Leaky integration. Short decay







Input: **events** (ON, OFF)
Grayscale frames are just used for visualization.

$$\tilde{L}(\boldsymbol{x}_k, t_k) = e^{-\alpha \Delta t_k} s(\boldsymbol{x}_k, t_{k-1}) + p_k$$

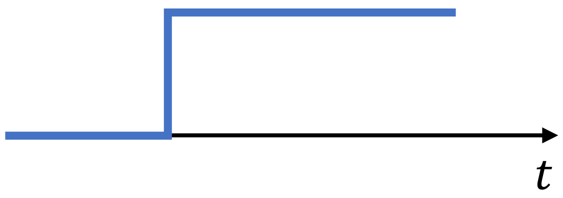
starting from s=0

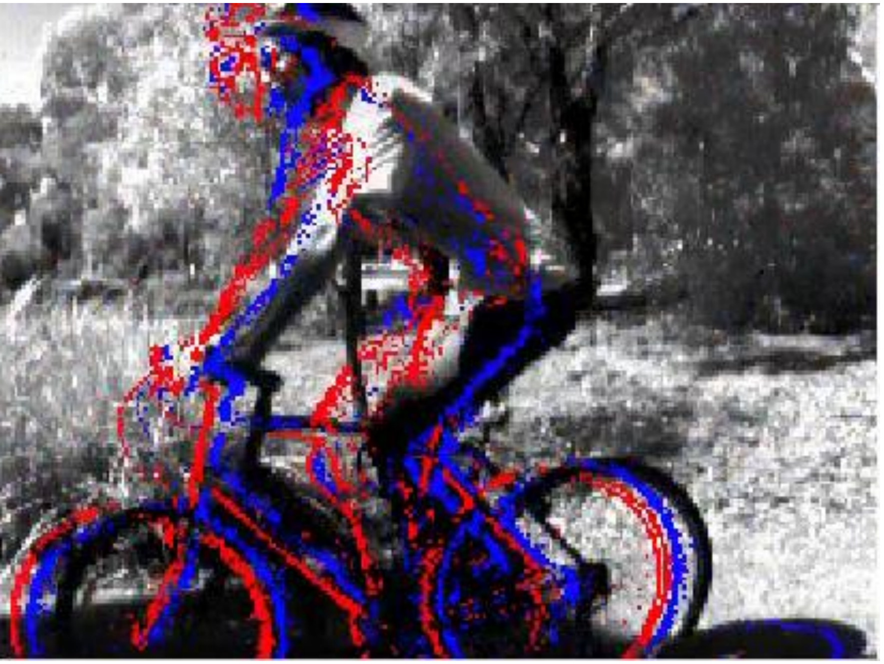
Real data (noisy)

Bicycle scene

$\alpha = 0$

Direct (ordinary) Integration





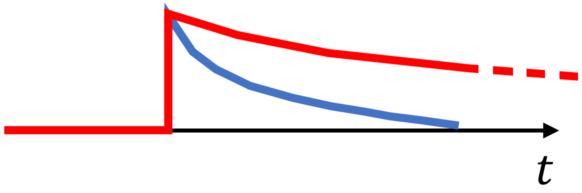
Input: **events** (ON, OFF)
Grayscale frames are just used for visualization.

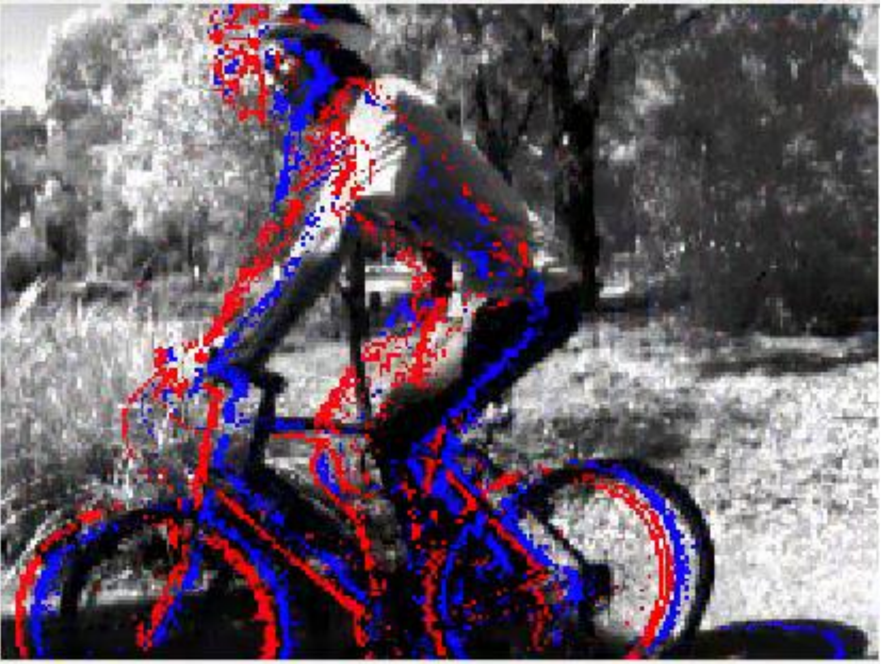
$$\tilde{L}(\boldsymbol{x}_k, t_k) = s(\boldsymbol{x}_k, t_{k-1}) + p_k$$

starting from *s*=0

$\alpha = 0.1$

Leaky Integration. Long decay





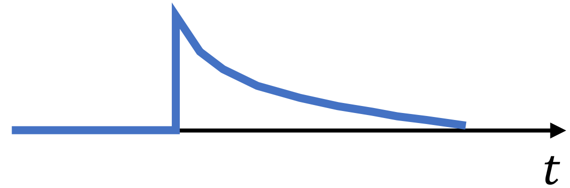
Input: **events** (ON, OFF)
Grayscale frames are just used for visualization.

$$\tilde{L}(\boldsymbol{x}_k, t_k) = e^{-\alpha \Delta t_k} s(\boldsymbol{x}_k, t_{k-1}) + p_k$$

starting from s=0

$\alpha = 0.3$

Leaky Integration



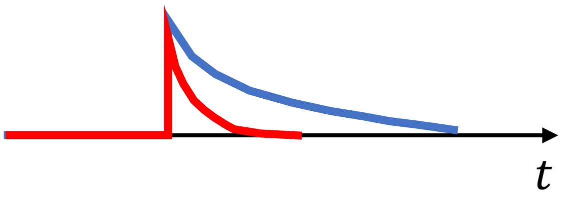
Input: **events** (ON, OFF)
Grayscale frames are just used for visualization.

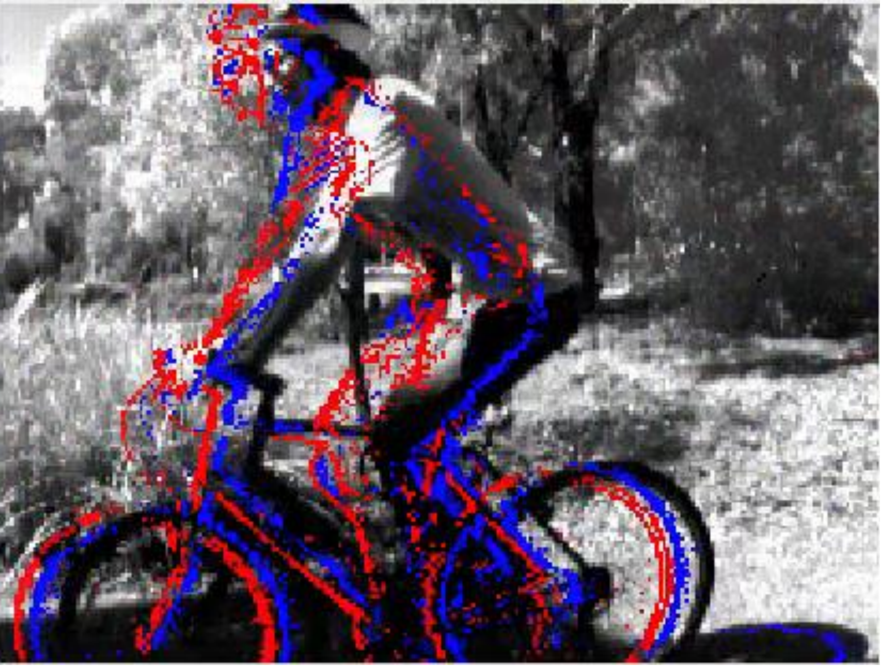
$$\tilde{L}(\boldsymbol{x}_k, t_k) = e^{-\alpha \Delta t_k} s(\boldsymbol{x}_k, t_{k-1}) + p_k$$

starting from s=0

$\alpha = 2$

Leaky Integration. Short decay





Input: **events** (ON, OFF)
Grayscale frames are just used for visualization.

$$\tilde{L}(\boldsymbol{x}_k, t_k) = e^{-\alpha \Delta t_k} s(\boldsymbol{x}_k, t_{k-1}) + p_k$$

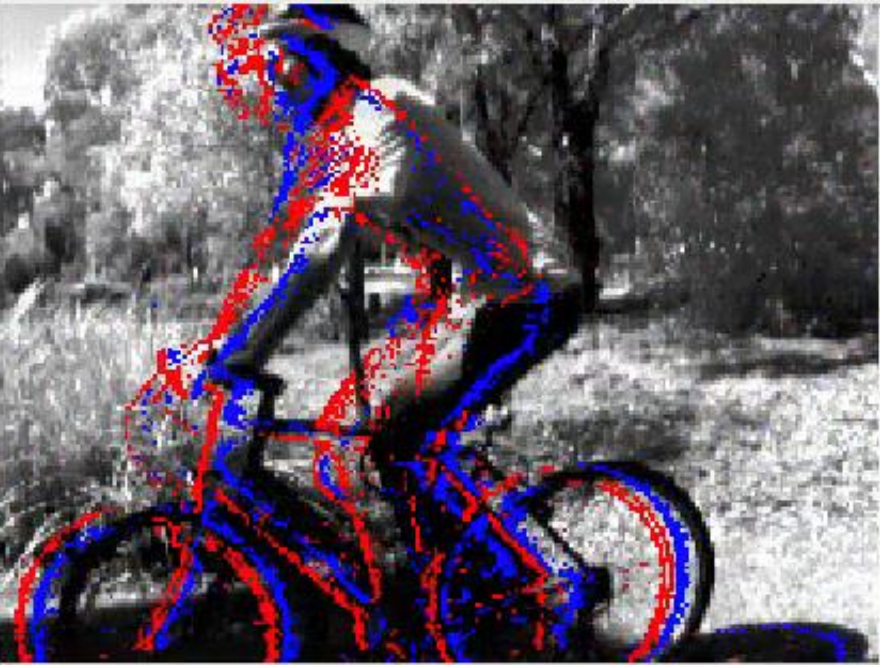
starting from *s*=0

$\alpha = 2$

Complementary Filter

- Combining **frames** and **events** in per-pixel filters.
- **Frames** provide slowly varying information (**low** temporal freq.)
- **Events** provide **high** temporal frequency information

Slowed down 5x



Input: **events** (ON, OFF) and grayscale **frames**

Output of the filter

Scheerlinck et al., *Continuous-time Intensity Estimation Using Event Cameras*, ACCV 2018.

References

Reading:

- Section 3 of Gallego et al., Event-based Vision: A Survey, TPAMI 2020
- E. Mueggler et al., *The Event-Camera Dataset and Simulator,* IJRR 2017, page 3.
- Scheerlinck et al., *Continuous-time Intensity Estimation UsingEvent Cameras*, ACCV 2018.
- Scheerlinck et al. *Asynchronous Spatial Image Convolutions for Event Cameras*, RA-L 2019

- Jupyter notebook demo