

# Event-based Robot Vision

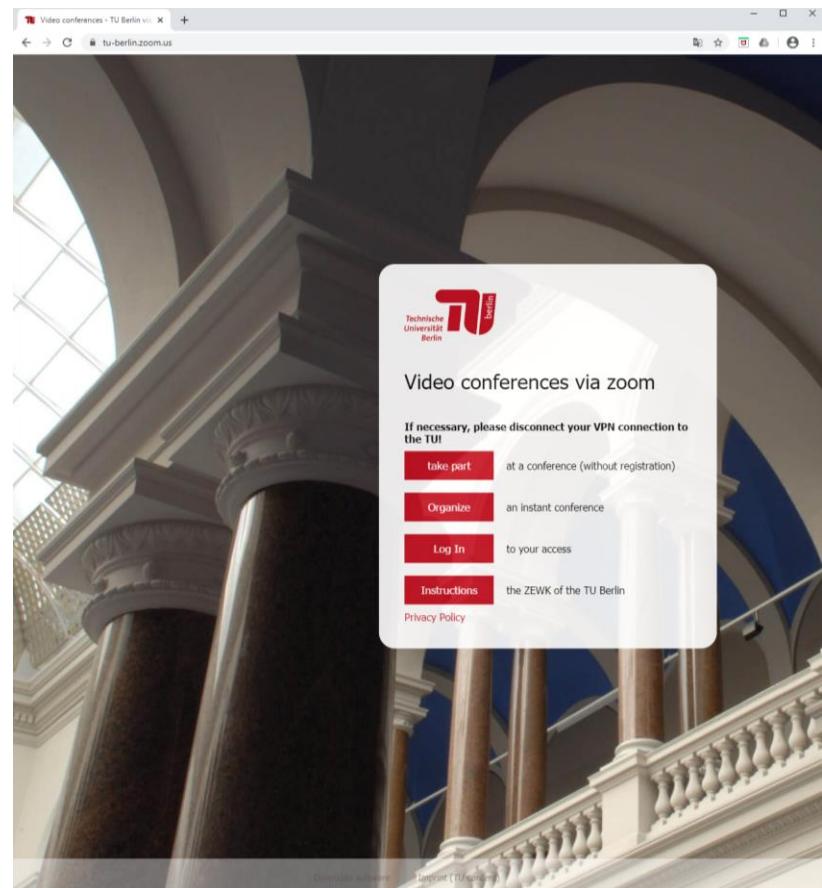
Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

# Use TU Berlin's Zoom

- Register / Login here:  
<https://tu-berlin.zoom.us/>
- Please download the [Zoom client](#) and familiarize yourself with this tool.
- **Join the lectures** either using URL or meeting ID and password.
  - **Tuesdays, 10 – 14 h** (Vorlesung):  
Join with [this link](#)  
Meeting ID: 956-1107-4544  
Password: 93310701
  - **Thursday, 14 - 16 h** (Übung)  
Join with [this link](#)  
Meeting ID: 988-0732-8139  
Password: 50820495

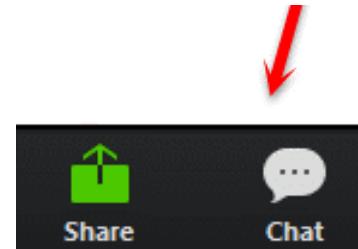


# Questions during the presentation?

- “Raise your hand” on Zoom or
  - Participants → Blue icon “Raise Hand”



- Write the question in the **Zoom chat**



# Robotics & Computer Vision have endless applications

AR/VR. Augmented life, work, entertainment.



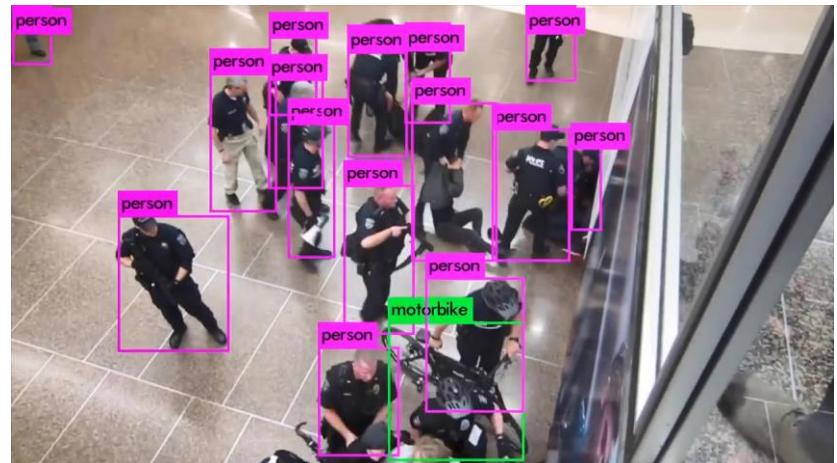
Drones. Entertainment. Autonomous vehicles



Agricultural Robotics



YOLOv3: Real-Time Object Detection. Surveillance,...



# Is everything solved?

- What are the limitations of these systems?
- What are the challenging scenarios?

Latency



Motion blur



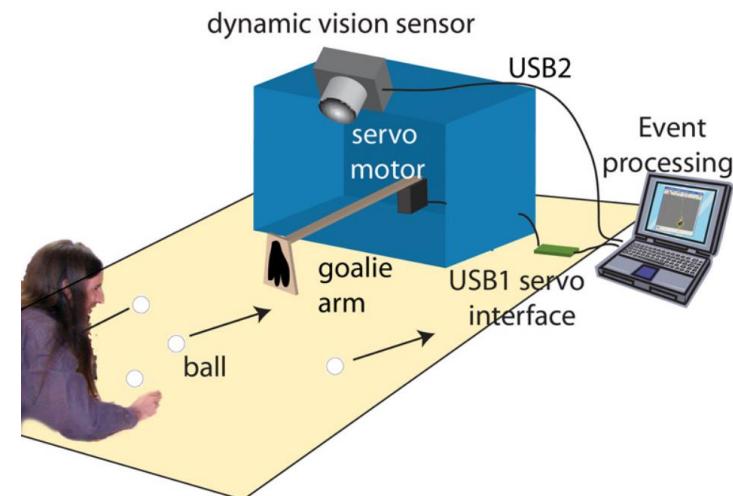
Dynamic Range



**Event-based cameras do not suffer from these problems!**

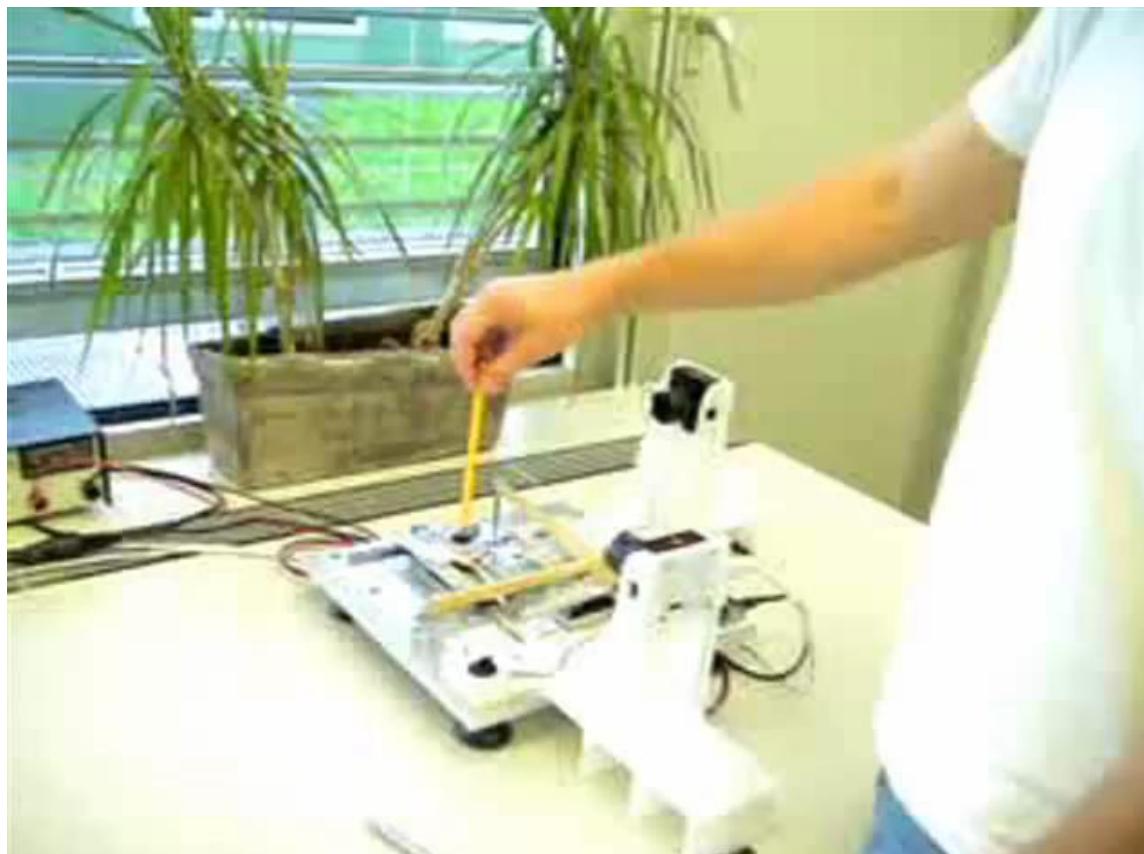
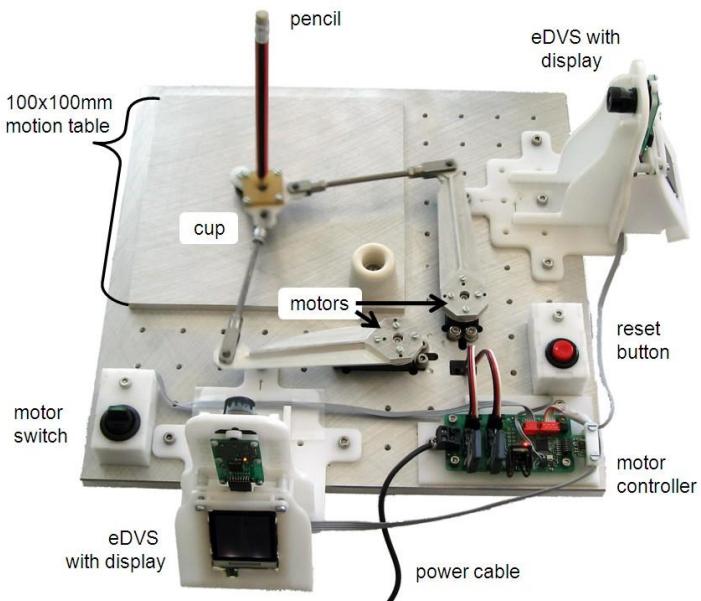
# “RoboGoalie”. Low Latency

- High-speed perception (with a DVS) and control



# Pencil Balancer. Low Latency

- High-speed perception (with two DVSs) and control



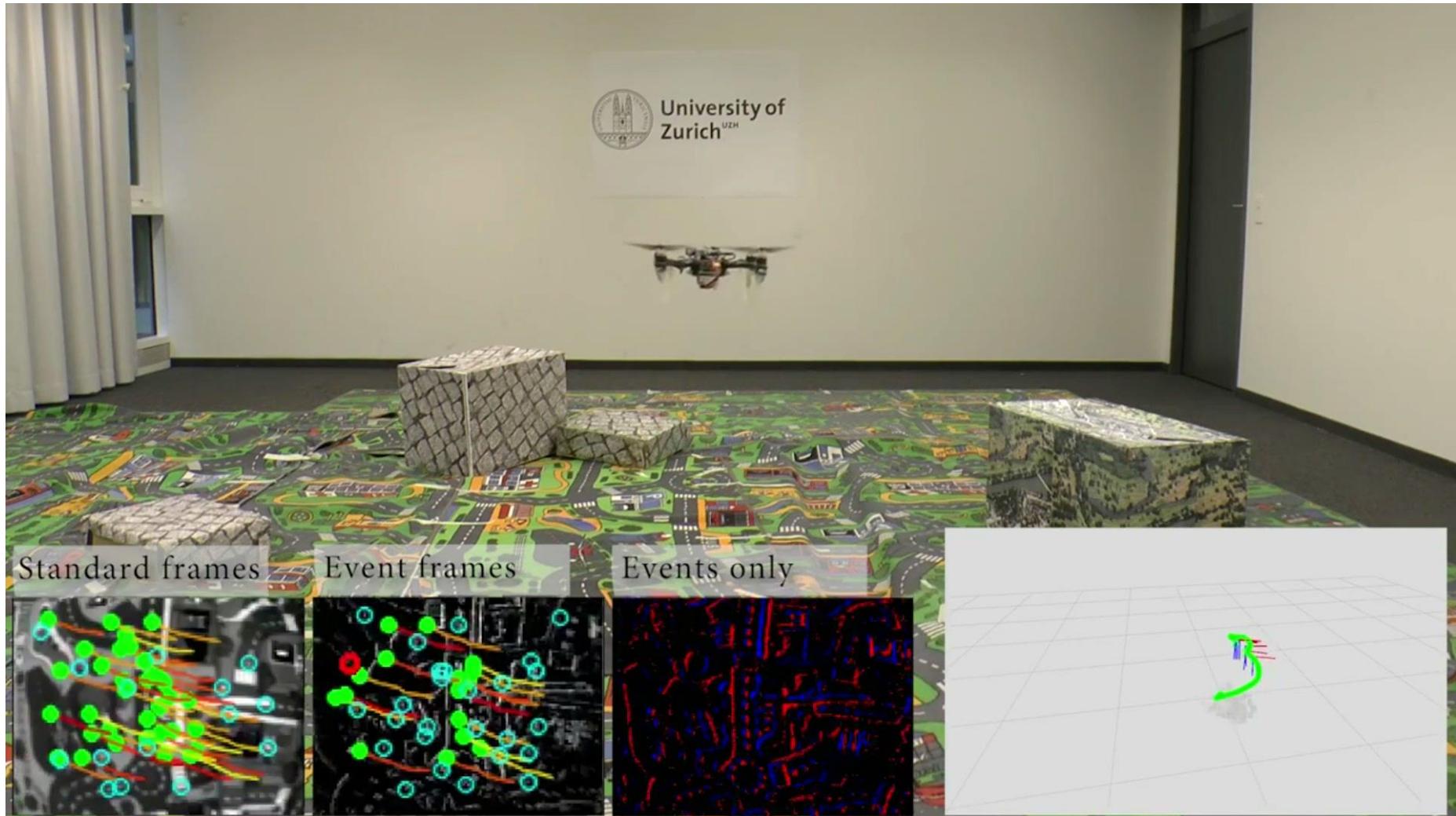
# High speed 3D scanner

- Using an event-based camera and a laser projector



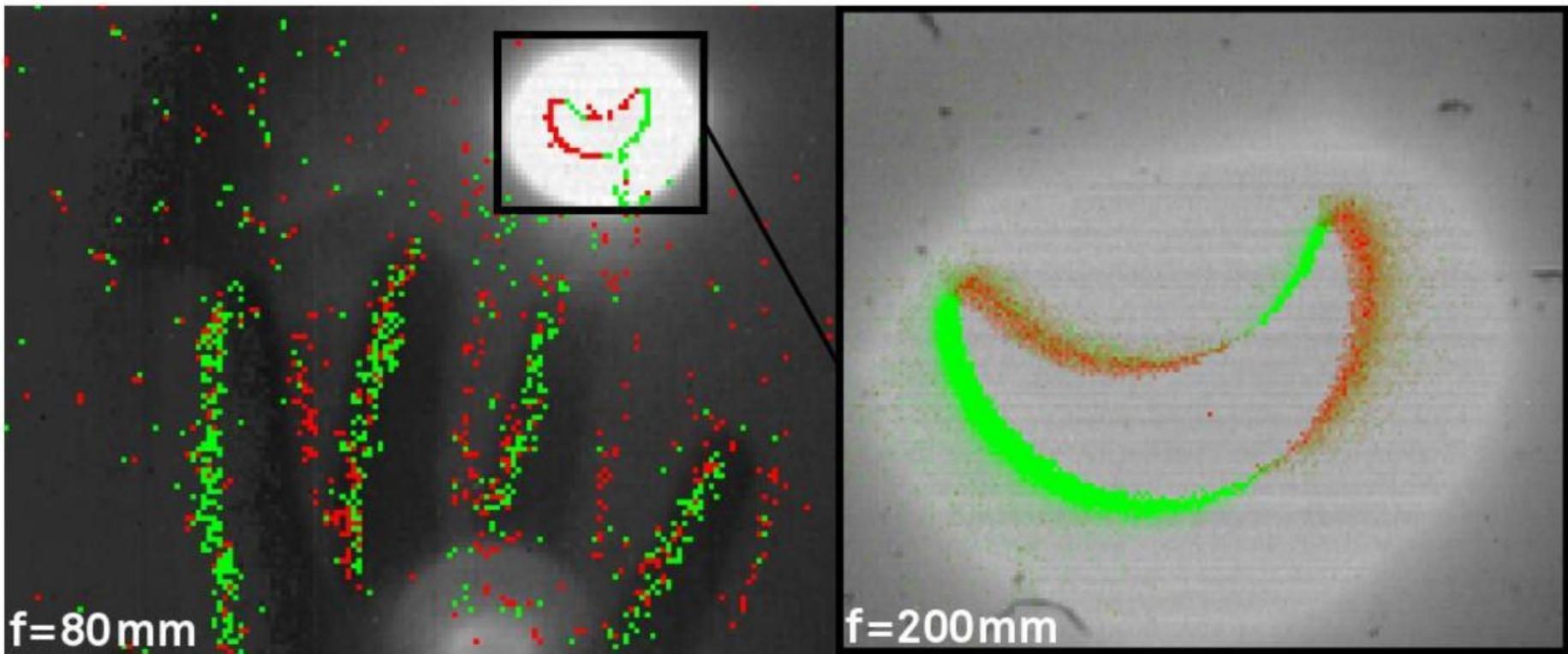
# Robot in Difficult Illumination Conditions

- Tightly coupled sensor fusion, fully onboard processing



# High Dynamic Range Scene

- Ability to see very bright and very dark scenes, simultaneously
- Image of the solar eclipse (March 2015) captured by a DAVIS

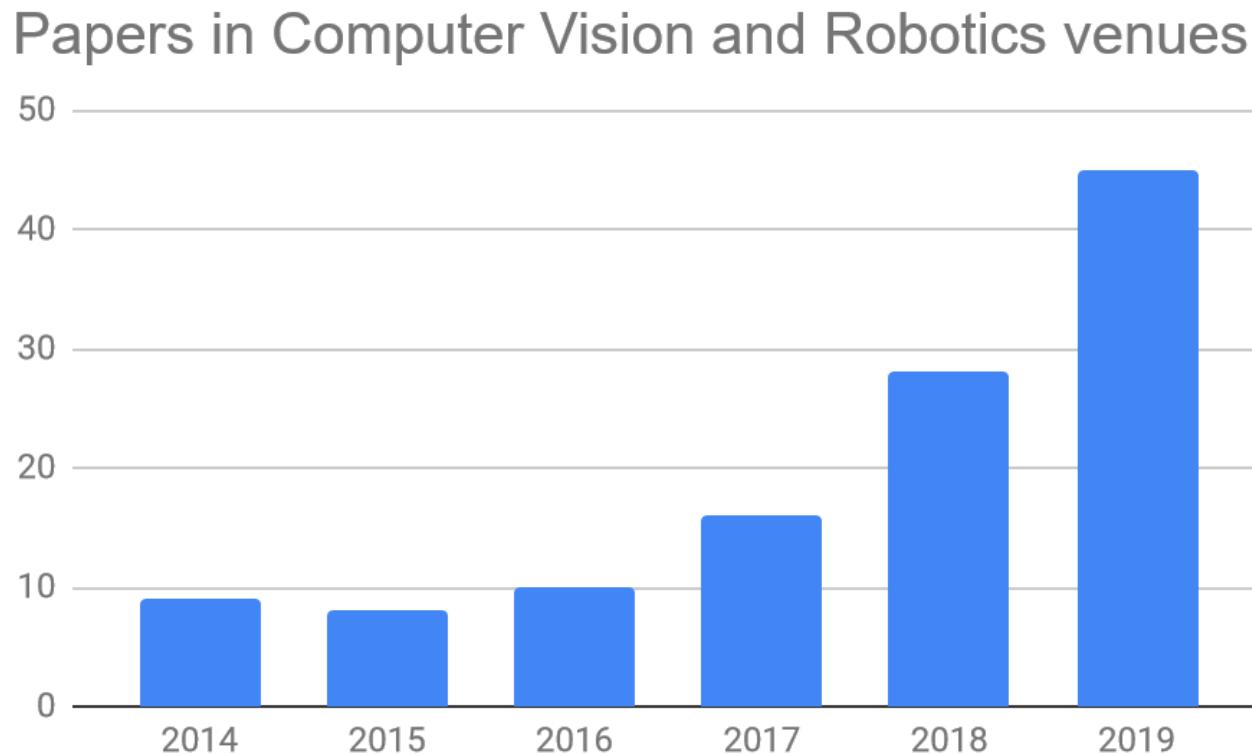


# Course Objectives

- To understand a new, efficient way of acquiring and processing visual information that is inspired in the human visual system.
- Know the state of the art in Event-based Vision
- Gain practical experience in processing event data

# Event-based vision is growing

- Papers published during the last six years at Computer Vision and Robotics venues\*



\* PAMI, IJCV, CVPR+W, ECCV+W, ICCV +W, BMVC, WACV, 3DV, ICIP, IROS +W, ICRA +W, RSS, ICCP, ICASSP, CoRL, IJRR, TRO, RAL

# Course Content in Questions

The topics covered include answering the following questions:

## 1. What is an event-based camera?

- How does it work?
- How does it capture a scene?
- Why does it work like that?
- What are the pros and cons with respect to standard cameras?
- How have event-based cameras evolved over the years?

## 2. What can it be used for?

- How can we extract information from the camera's output to solve our problem?
- What are the pros and cons of the resulting vision system?
- How can we use it to estimate motion?
- How can you use it for 3D reconstruction?
- Is it possible to recover absolute intensity from the events? How?
- How can we use it to recognize objects or gestures?
- How can these novel sensors be combined with others to produce more robust systems?

## 3. Where can I get one? (Practice)

- How can I play with the data that it produces?

# Course Content

The topics covered include the following:

## 1. Principle of operation of event-based cameras

- Bio-inspired motivation.
- Advantages, disadvantages and challenges / opportunities.

## 2. Algorithms / Applications:

- Feature detection and tracking
- Motion estimation: optical flow estimation, ego-motion estimation, simultaneous localization and mapping (SLAM)
- Depth estimation (3D reconstruction) monocular or stereo
- Image intensity reconstruction from events
- Event-based filtering and signal processing
- Event-based pattern recognition and machine learning
- Event-based sensor fusion
- ...

## 3. Physical devices: actual event-based cameras & companies



Mahowald & Mead  
Scientific American 1991

# Difference with Bio-inspired Computer Vision

- This course:
  - Lectures + per-session exercises
  - Cover a **broad range of topics in event-based vision**
  - Engineering viewpoint:
    - Being bio-inspired is nice, but it is not enforced.



- Bio-inspired Computer Vision:
  - **Brings together two subjects:**
    - Human visual perception – Biology, Psychology (Prof. Maertens)
    - Technological advances in handling visual information – Engineering
  - 4 introductory lectures, then students work in teams on **projects**

# Materials

- ISIS Webpage of the course:  
<https://isis.tu-berlin.de/course/view.php?id=19385>



- Overview paper:  
Gallego et al., [Event-based Vision: A Survey](#), arXiv 2019.



## Event-based Vision: A Survey

Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, Davide Scaramuzza

**Abstract**— Event cameras are bio-inspired sensors that work radically different from traditional cameras. Instead of capturing images at a fixed rate, they measure per-pixel brightness changes asynchronously. This results in a stream of events, which encode the time, location and sign of the brightness changes. Event cameras possess outstanding properties compared to traditional cameras: very high dynamic range (140 dB vs. 60 dB), high temporal resolution (in the order of  $\mu\text{s}$ ), low power consumption, and do not suffer from motion blur. Hence, event cameras have a large potential for robotics and computer vision in challenging scenarios for traditional cameras, such as high speed and high dynamic range. However, novel methods are required to process the unconventional output of these sensors in order to unlock their potential. This paper provides a comprehensive overview of the emerging field of event-based vision, with a focus on the applications and the algorithms developed to unlock the outstanding properties of event cameras. We present event cameras from their working principle, the actual sensors that are available and the tasks that they have been used for, from low-level vision (feature detection and tracking, optic flow, etc.) to high-level vision (reconstruction, segmentation, recognition). We also discuss the techniques developed to process events, including learning-based techniques, as well as specialized processors for these novel sensors, such as spiking neural networks. Additionally, we highlight the challenges that remain to be tackled and the opportunities that lie ahead in the search for a more efficient, bio-inspired way for machines to perceive and interact with the world.

**Index Terms**—Event Cameras, Bio-Inspired Vision, Asynchronous Sensor, Low Latency, High Dynamic Range, Low Power.

---

**1 INTRODUCTION AND APPLICATIONS**

“**T**HE brain is imagination, and that was exciting to me; I wanted to build a chip that could imagine something!”

as well as new computer vision and robotic tasks. Sight is, by far, the dominant sense in humans to perceive the world, and, together with the brain, learn new things. In recent years, this technology has attracted a lot of attention from

# Materials

- ISIS Webpage of the course:

<https://isis.tu-berlin.de/course/view.php?id=19385>

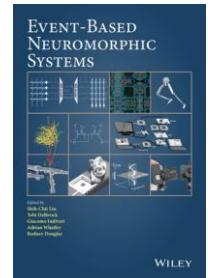


- Overview paper:

Gallego et al., [Event-based Vision: A Survey](#), arXiv 2019.

- List of Event-based Vision Resources:

- Links to papers, books, videos, datasets, cameras, code...
- [Topical reviews](#)
- [Research talks at International Workshops and Tutorials](#)
- [PhD and Master theses](#) on event-based vision



Event-Based  
Neuromorphic Systems

- General computer vision & robotics textbooks:

- [R. Szeliski's book](#) is freely available
- P. Corke's book: [Robotics, Vision and Control \(2nd Ed\)](#)

# List of Event-based Vision resources

Screenshot of a GitHub repository page for "uzh-rpg / event-based\_vision\_resources".

The repository has the following statistics:

- Watched by 100 users (highlighted with a red box)
- Starred by 743 users
- Forked by 222 users
- Code: 584 commits
- Pull requests: 0
- Actions: 0
- Security: 0
- Insights: 0
- Branch: master
- New pull request
- Find file
- Clone or download

No description, website, or topics provided.

Recent commits:

File	Commit Message	Date
README.md	Update README.md	yesterday
Contributing.md	abbrev link ieee	16 months ago
README.md	Update README.md	yesterday
README.md	(empty)	(empty)

## Event-based Vision Resources

### Table of Contents:

- Survey paper
- Devices and Manufacturers
- Companies working on Event-based Vision
- Neuromorphic Systems



# Schedule

- Fully online, unless TU Berlin decides otherwise.
- 14 weeks.
  - **Theory** (Tu.): We will presumably follow the topics in the order covered in the Survey paper
  - **Exercises** (Th.): Start simple, data reading and processing. Review some image processing. Then, increase complexity.
  - Equipment: Computer + Linux + Python (+Matlab?) + ROS (Robot Operating System in C++ / Python)
- Interested in you **learning the fundamentals** rather than on covering a large amount of material.

# Grading

- No plan about examinations yet (originally, July 22<sup>nd</sup>).  
This is being coordinated by TU Berlin.

[https://www.pruefungen.tu-berlin.de/menue/sommersemester\\_2020\\_digital/studierende/](https://www.pruefungen.tu-berlin.de/menue/sommersemester_2020_digital/studierende/)

[https://www.pruefungen.tu-berlin.de/menue/sommersemester\\_2020\\_digital/lehrende/](https://www.pruefungen.tu-berlin.de/menue/sommersemester_2020_digital/lehrende/)

- Grading scale:

Note:	1.0	1.3	1.7	2.0	2.3	2.7	3.0	3.3	3.7	4.0
Punkte:	86.0	82.0	78.0	74.0	70.0	66.0	62.0	58.0	54.0	50.0

- Original plan:

- Two written test: 30% + 40%
- Practice (exercises): 30%

# Enrollment

- Currently, the class is full
- Maximum number of students: 30
- If people drop class, then use the waiting list.

# Next steps

- Skim through the material on ISIS
  - Specially, the overview paper.
- Prepare equipment (computer & software)
  - Get familiarized with Zoom
- Ask questions
  - During the online session or on ISIS (between sessions)

# Know your Audience

- What's your major?
- What's your experience in Computer Vision, ML and Robotics? (What courses have you taken?)
- What are your coding / programming skills?
  - What languages? What level?
- Why did you enroll in the course?
- What do you expect from the course?

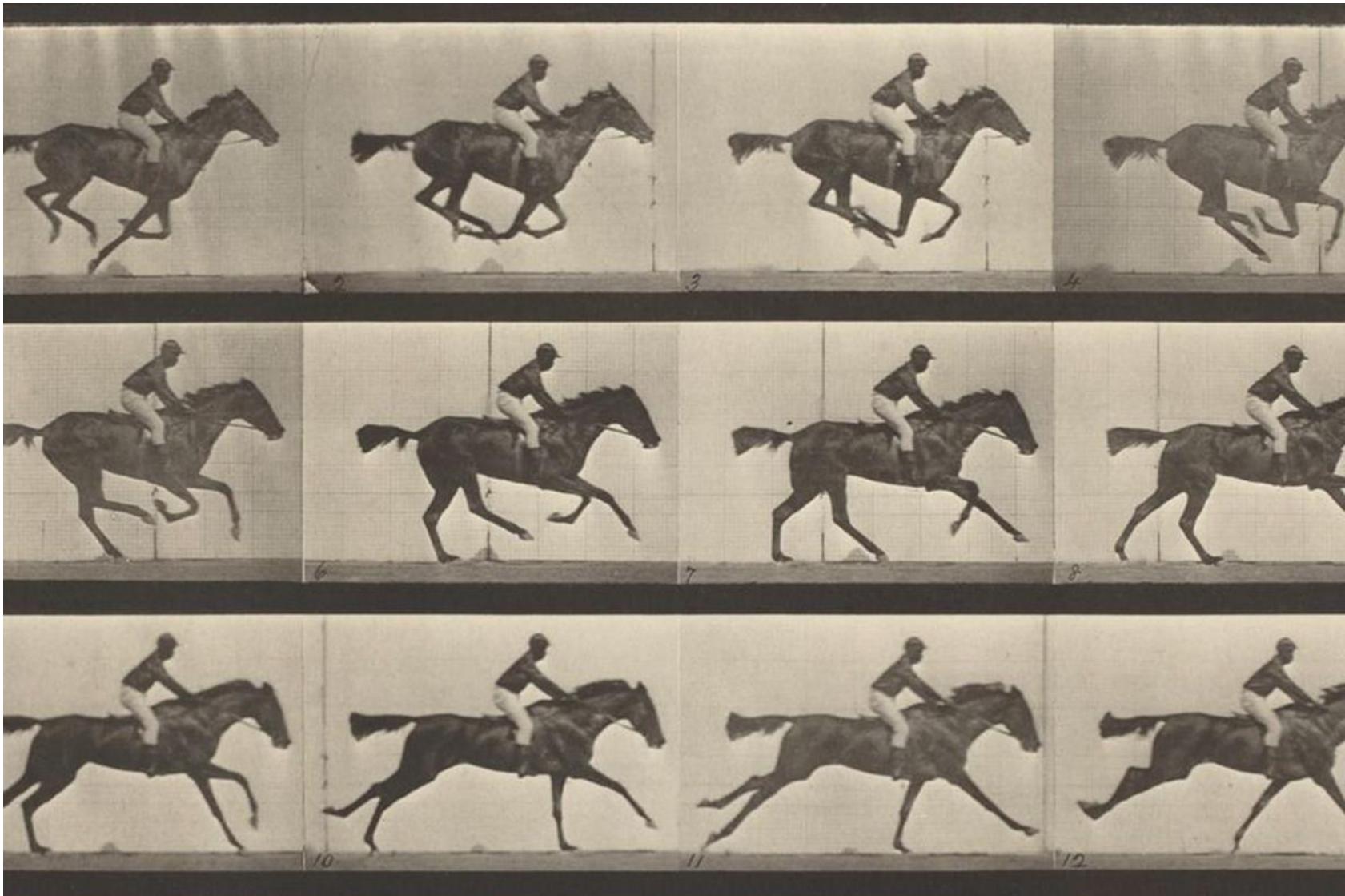
# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

# How do we capture visual information?



Master of Photography E. Muybridge ca. 1878. *"The Horse in Motion"*

# How do we capture visual information?

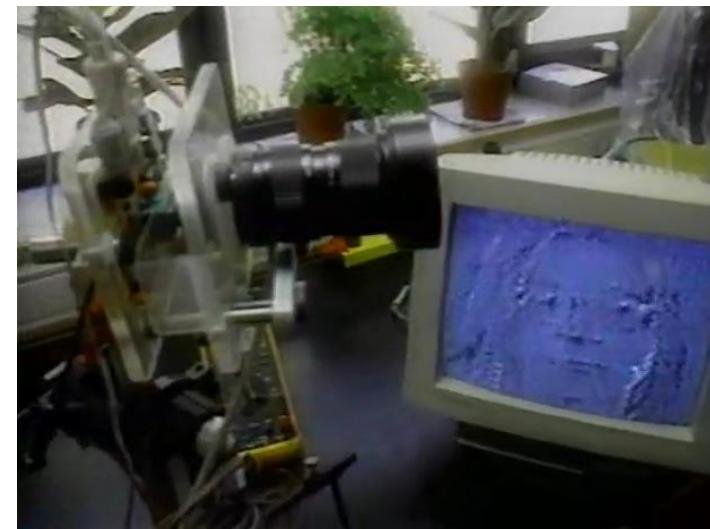
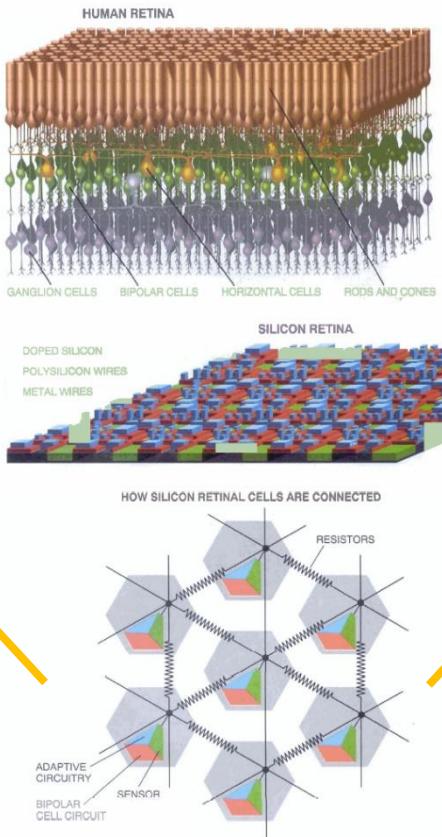


# M. Mahowald & C. Mead (~1990)

- The spark of “Neuromorphic Engineering”
- Bringing **biology** into chip design on silicon (**engineering**)



Modeling Neural Structures in Silicon



Retina-on-a-chip demonstration

# Two types of cameras important to us (~2014)

**Standard, frame-based camera**  
(outputs images  
Here, frame rate= 6 Hz)

vs

**Event-based camera**  
(outputs “events” or  
“spikes”, represented  
as **red** and **green** dots  
in the video)

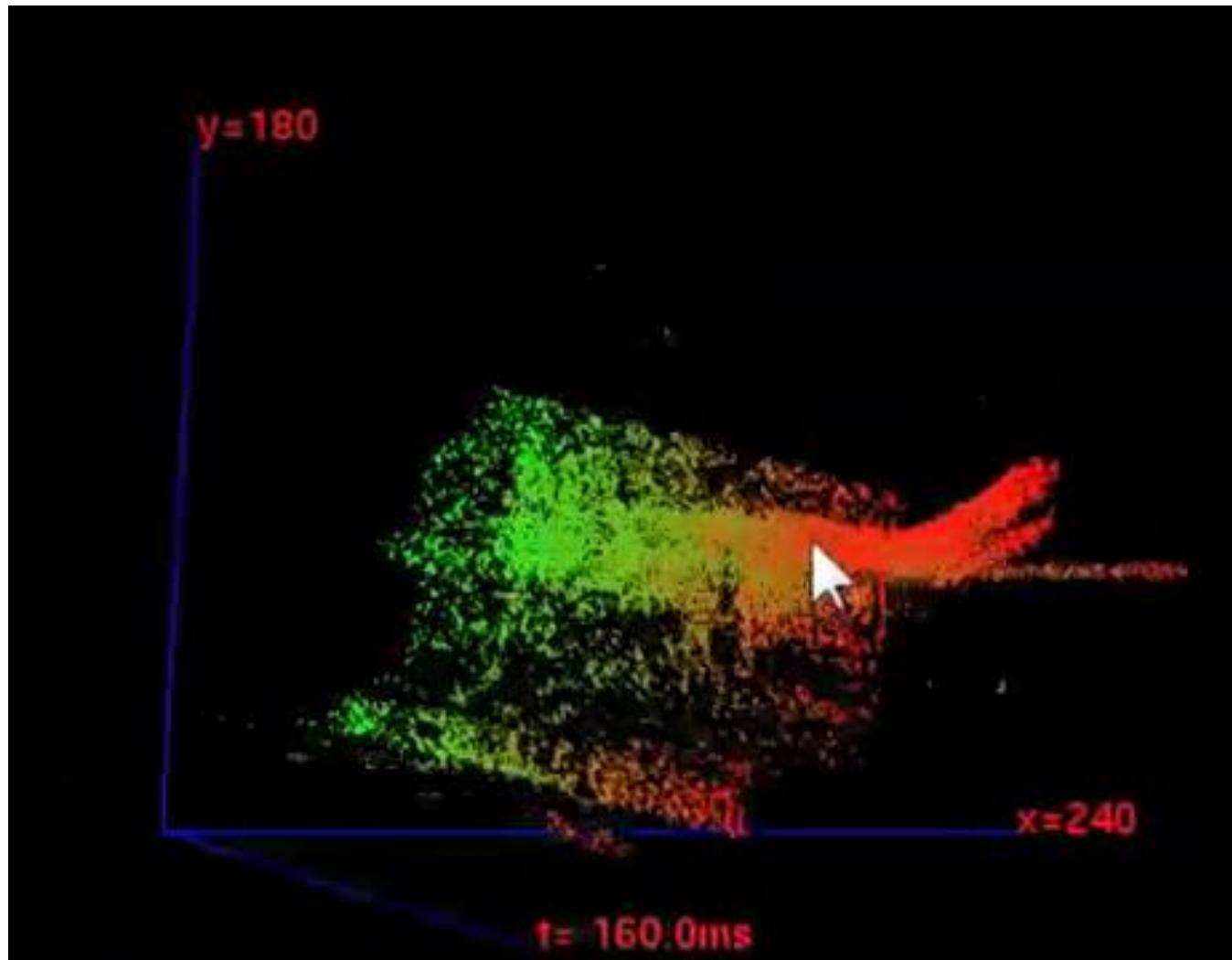


# Space-time visualization of events

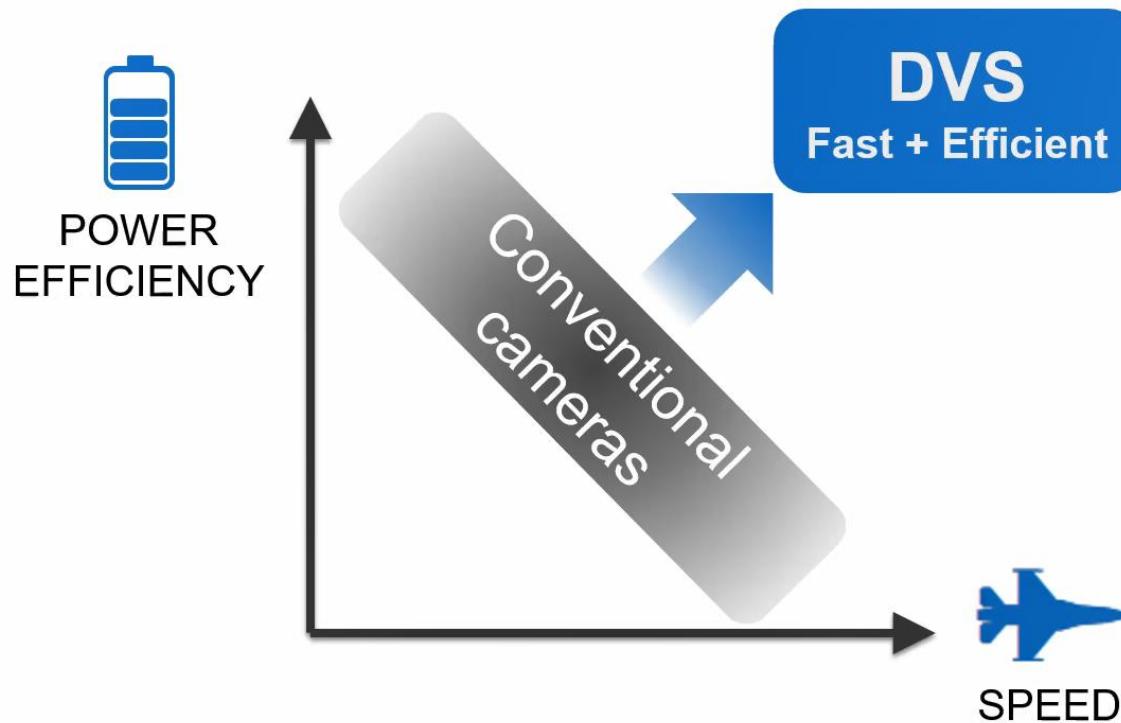
**Space:** sensor size is  
240 x 180 pixels

**Time:** 160 ms

In the video, time is  
colored: **red** means  
new, **green** means old  
(look at the trajectory  
of the tennis ball).



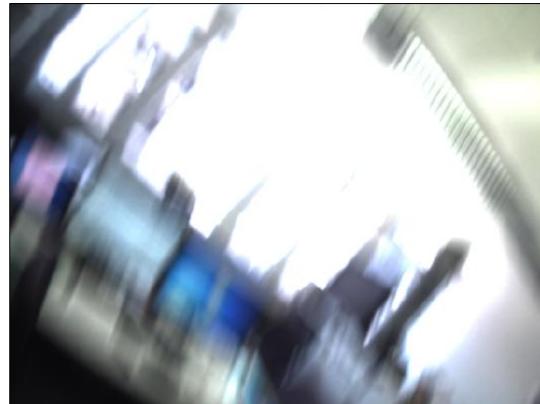
# Event camera. Efficiency - Latency trade off



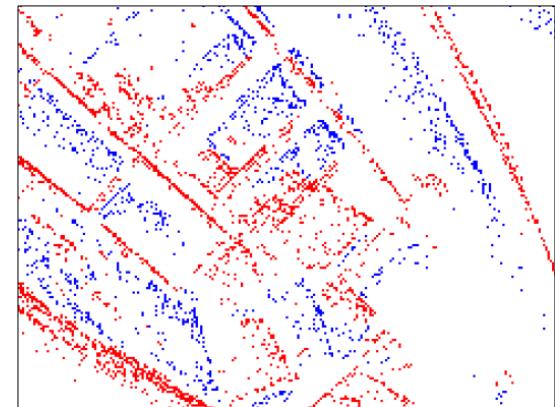
The DVS beats the power vs speed trade-off providing high speed at low power.

# Comparison of two camera types

> 60 years of research!



< 10 years research



## Standard Camera

	Standard Camera	Event Camera
Update rate	Low (synchronous)	High (asynchronous): 1 MHz
Dynamic Range	Low (60 dB)	High (140 dB)
Motion Blur	Yes	Almost none
Absolute intensity	Yes	No (but it can be reconstructed)
Mature?	Yes	Not yet

# References

## Reading:

- Mead & Mahowald. [\*The Silicon Retina\*](#), Scientific American 1991.
- If curious, read about early motion studies and devices:  
[https://en.wikipedia.org/wiki/Eadweard\\_Muybridge#1872-1879:\\_Stanford\\_and\\_horse\\_gaits](https://en.wikipedia.org/wiki/Eadweard_Muybridge#1872-1879:_Stanford_and_horse_gaits)  
[https://en.wikipedia.org/wiki/Eadweard\\_Muybridge#Later\\_motion\\_studies](https://en.wikipedia.org/wiki/Eadweard_Muybridge#Later_motion_studies)
- T. Delbruck, [\*The Slow but Steady Rise of the Event Camera\*](#), EE Times 2020.

## Videos:

- Silicon Vision - Misha Mahowald <https://www.dailymotion.com/video/x28ktma>
- E. Muybridge BBC documentary <https://youtu.be/5Awo-P3t4Ho>

# Event-based Robot Vision

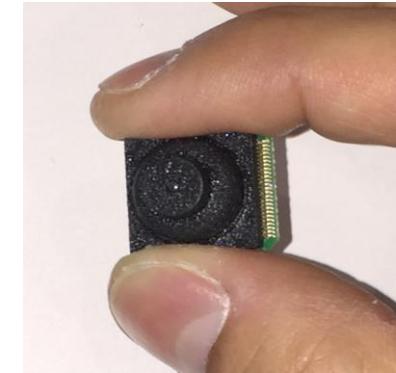
Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

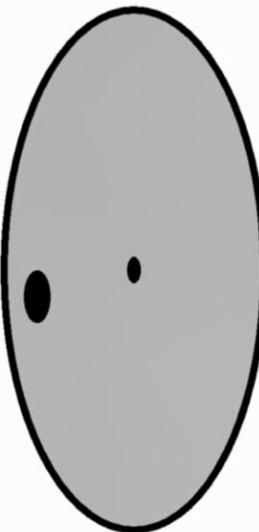
# What is an Event Camera?

- A novel sensor that only outputs **intensity changes, asynchronously!**
- Output is a stream of **events** at **microsecond resolution**
- **Low-latency** ( $\sim 1 \mu\text{s}$ )
- **Almost no motion blur**
- **High dynamic range** (140 dB instead of 60 dB)
- **Low power** (mW)
- **Traditional vision algorithms** cannot be directly used



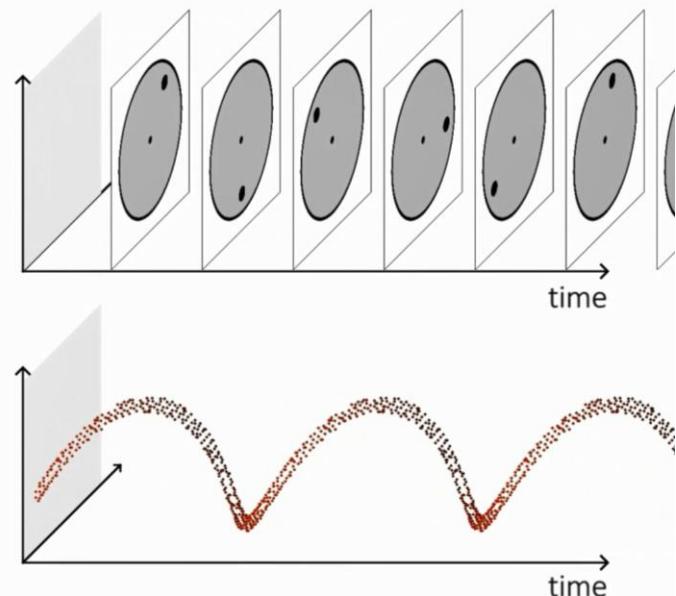
Mini DVS sensor from iniVation.com

<https://youtu.be/LauQ6LWTkxM?t=24>

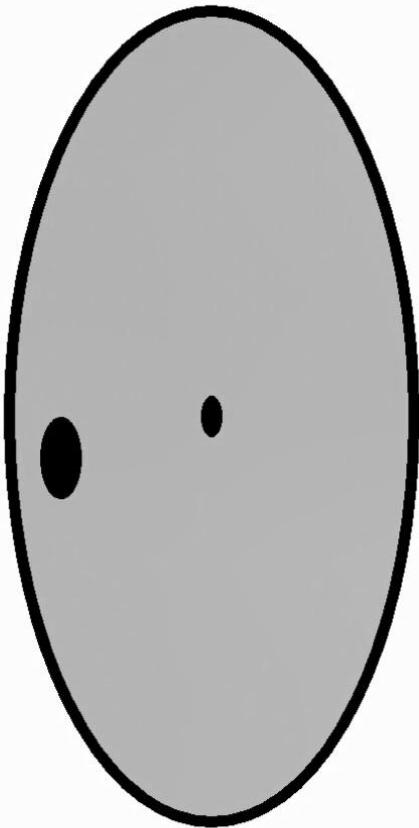


**standard  
camera  
output:**

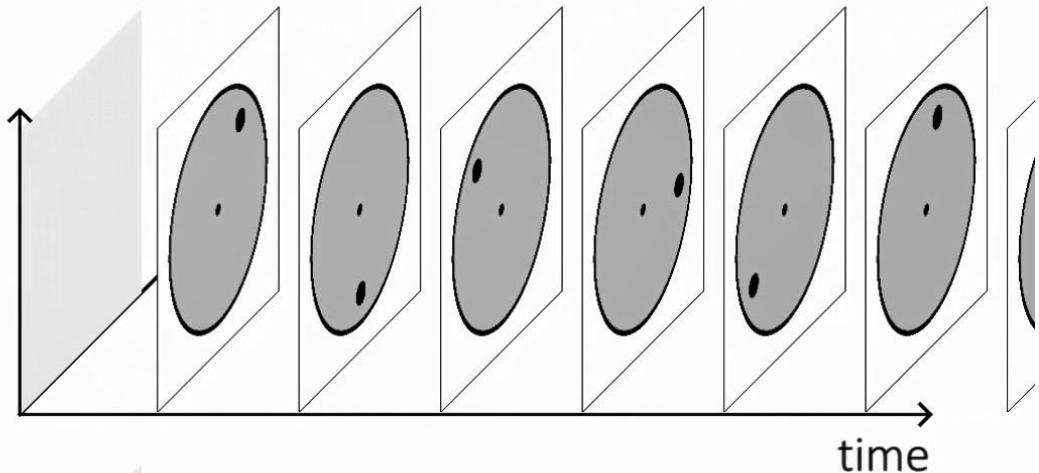
**event  
camera  
output:**



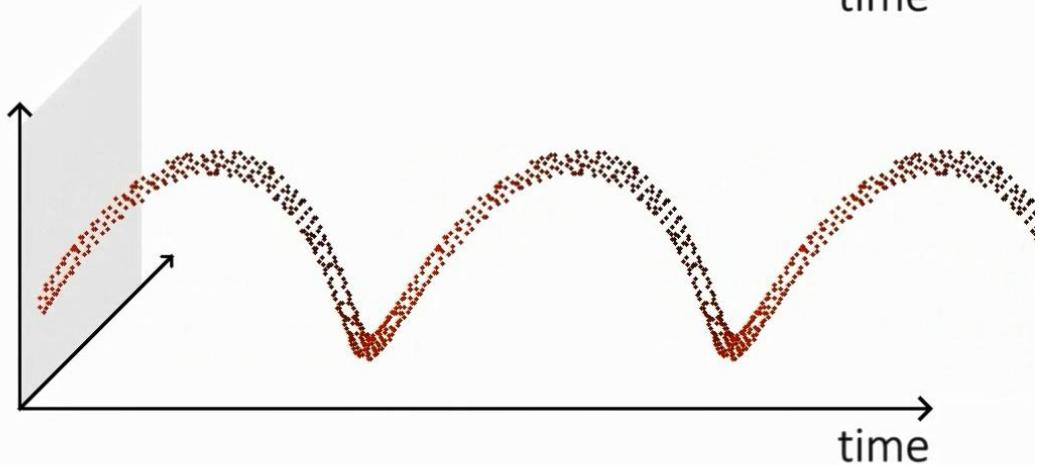
# What is an Event Camera?



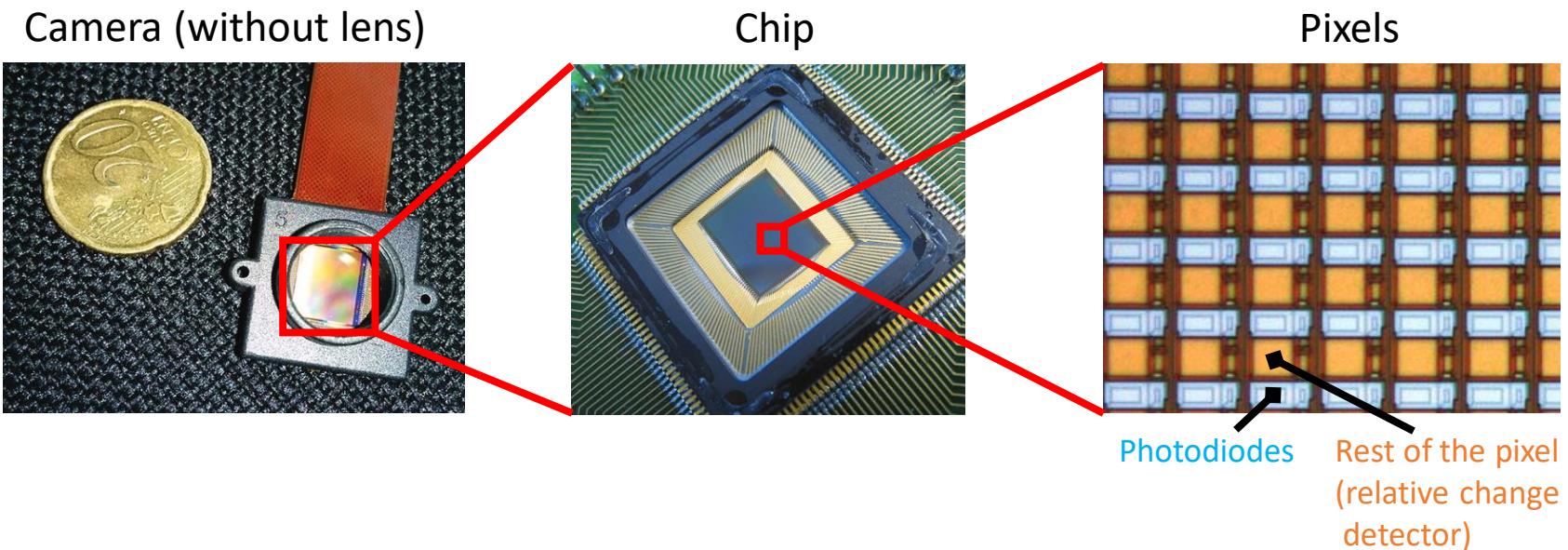
**standard  
camera  
output:**



**event  
camera  
output:**

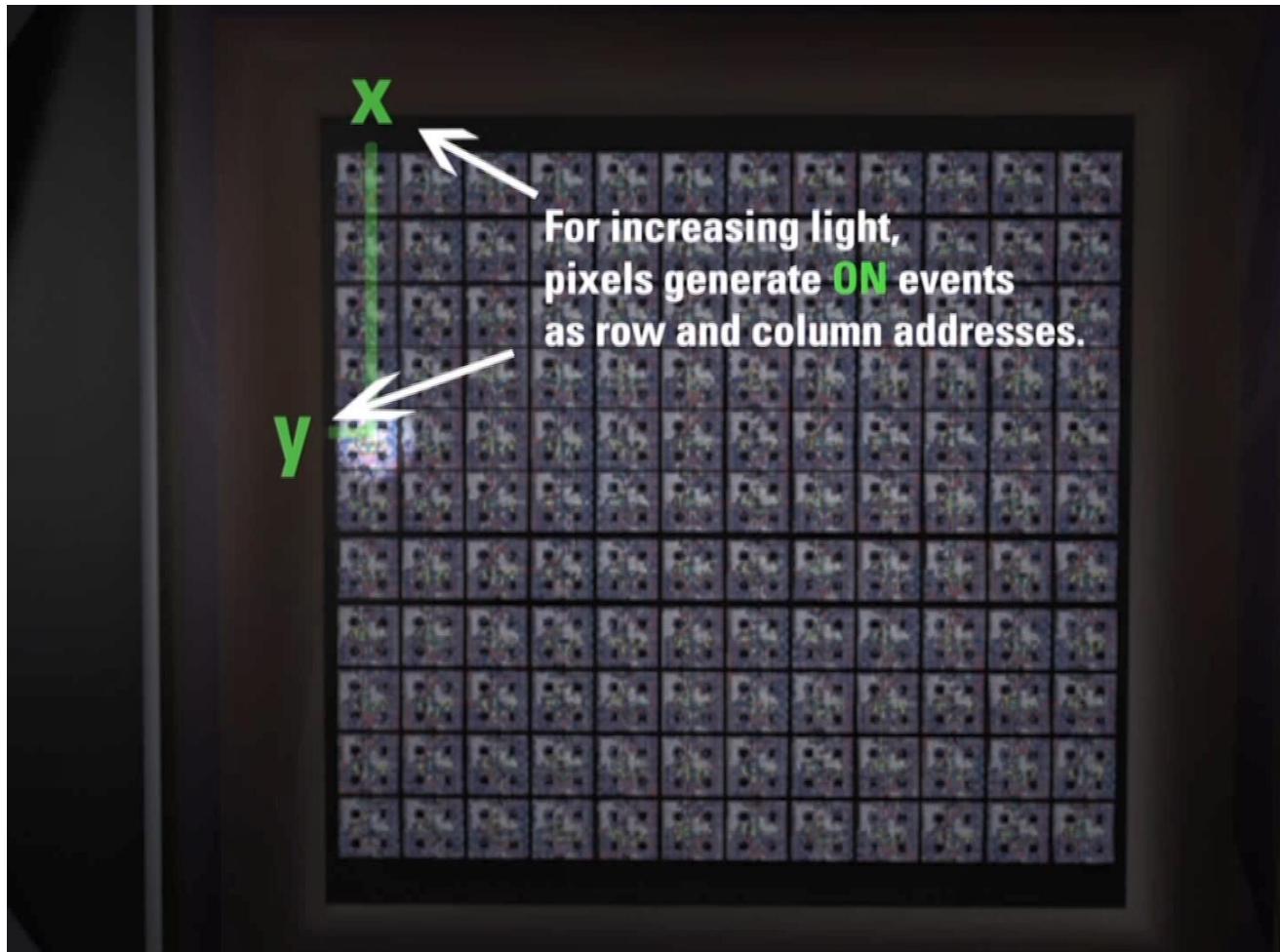


# From the camera to the pixels



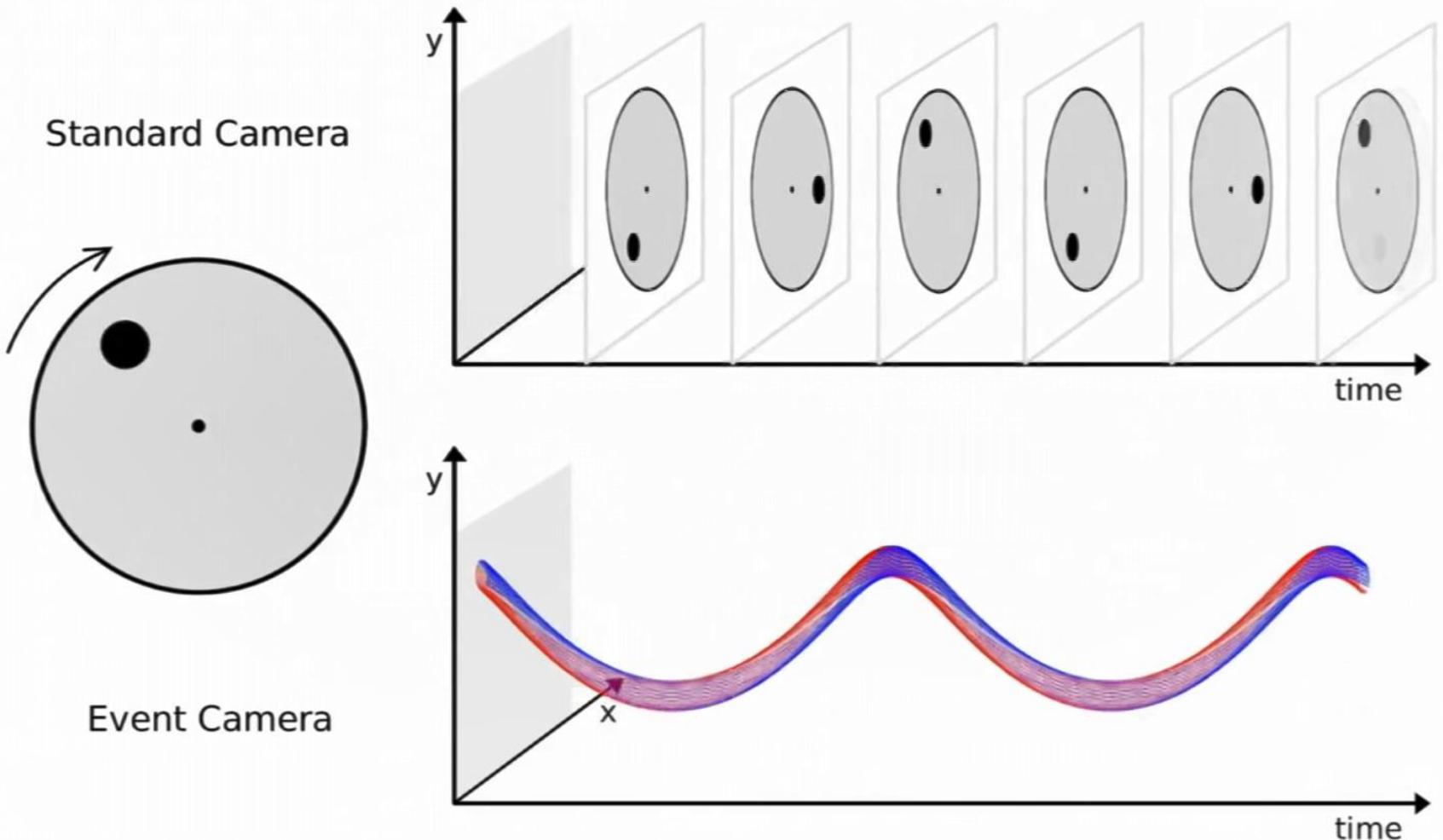
# Illustration with a light beam

- Event camera pixels respond **independently** and **asynchronously** to light intensity **changes**.

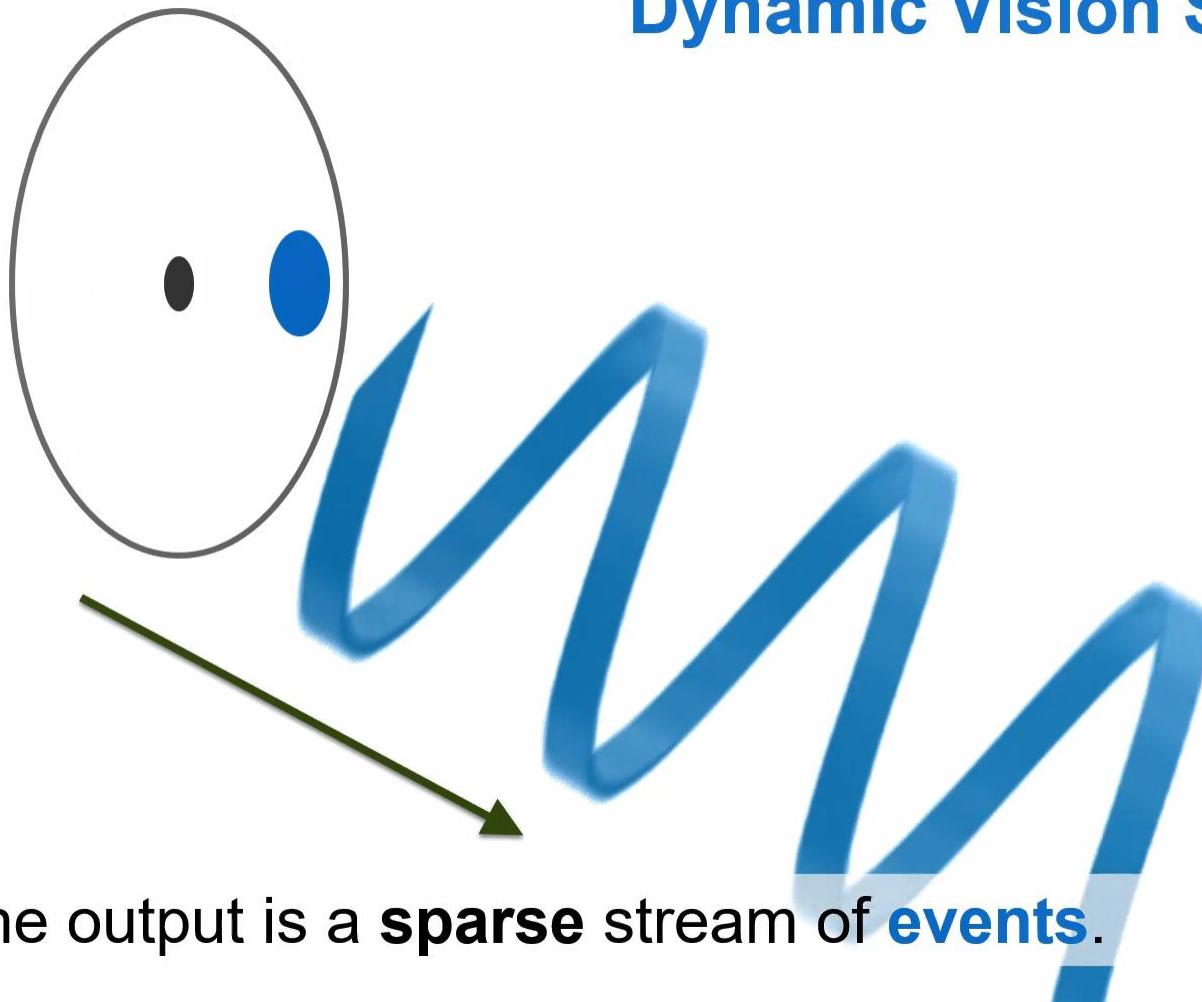


# Same explanation (including event polarity)

only informative pixels are provided

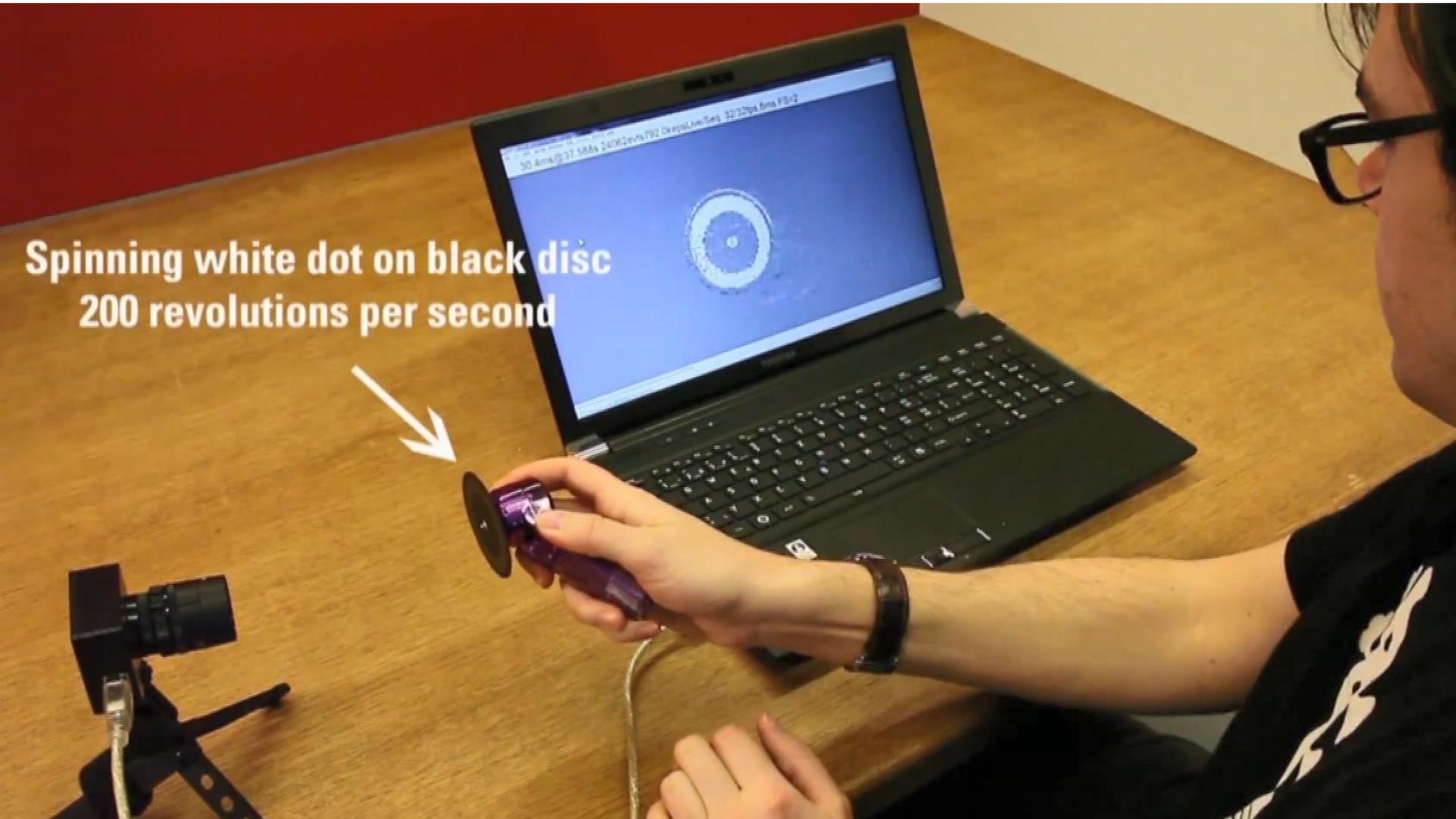


# Frame-based camera vs Event-based camera

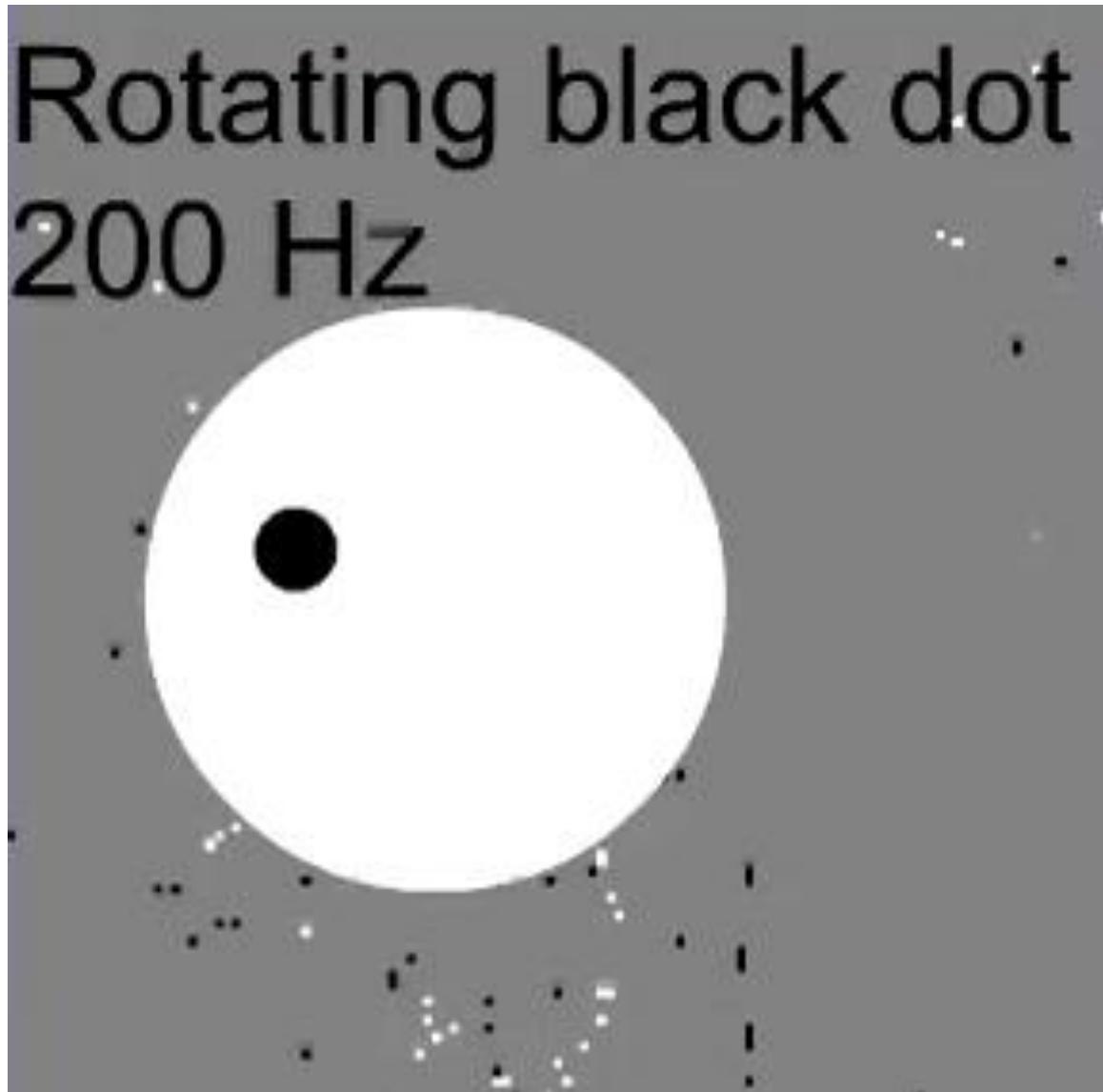


The output is a **sparse** stream of **events**.

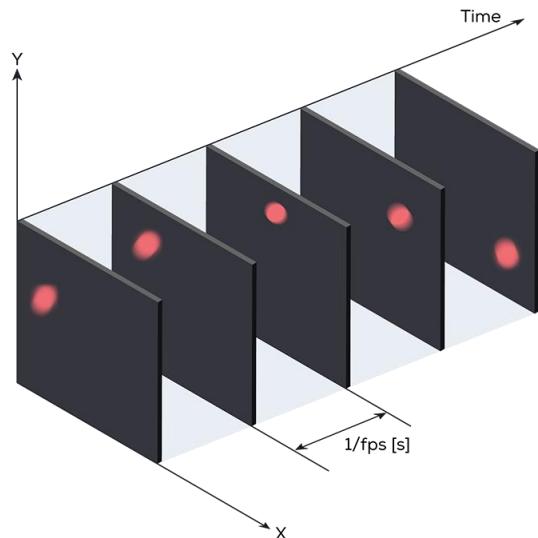
# Real data. High temporal resolution



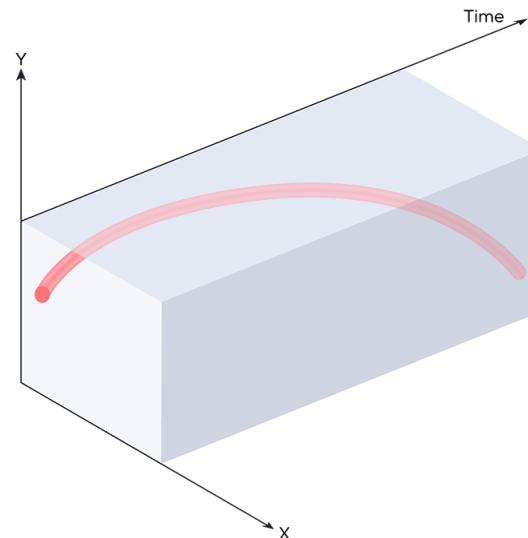
# Real event data & Space-time display



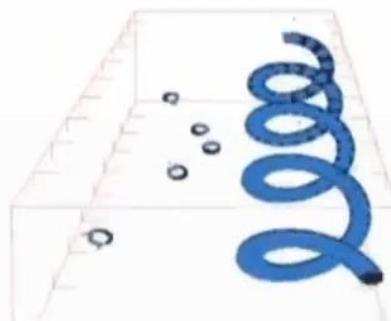
# Frame-based camera vs Event-based camera



Frame-based

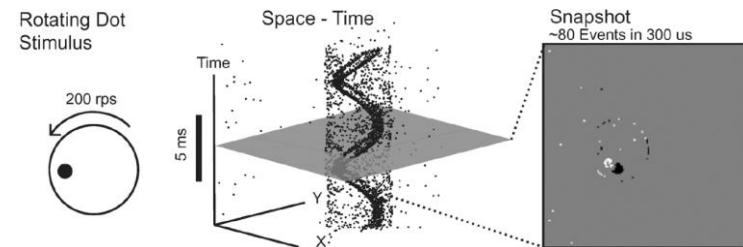


Event-based



# Why so many spinning-dot animations?

- It is on the seminal paper (2008)
- Every lab or company “has its own animation”.
- Event cameras work fundamentally different from standard cameras.
- Need to convey it in a simply way to your audience
  - Avoid getting your work rejected because there is a gap in understanding the context, the *novel sensing paradigm*
  - It also shows the high-speed capabilities of the sensor.



# Event-based Robot Vision

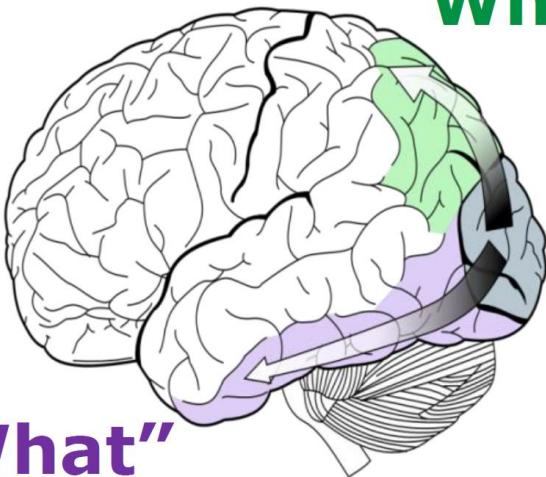
Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

# Human Visual Perception

- Dorsal (Transient) visual pathway
- Ventral (Sustained) visual pathway



**"Where"**

Transient pathway  
Magno cells  
Motion  
All over the retina  
Fast

**"What"**

Sustained pathway  
Parvo cells  
Details  
Fovea  
Slow

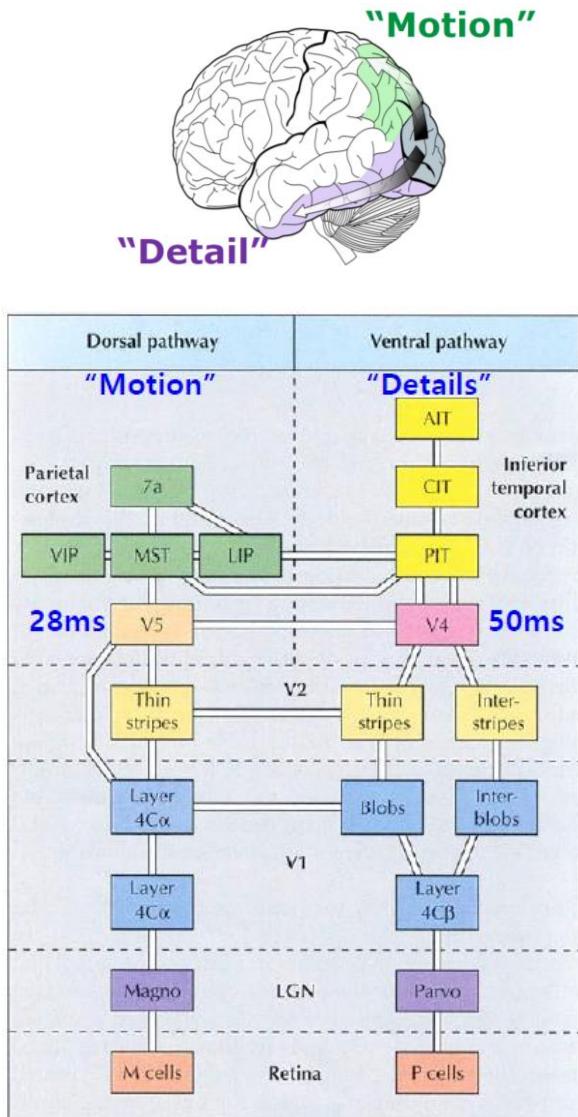


Color image



Events

# Pathways of Human Visual Processing



- “**Motion**” and “**Details**” are processed differently each other
  - “Motion” detection uses “edges”
  - “Details” recognition uses “shapes” and “colors”
- **Ventral pathway (“Details”) are deeper than Dorsal pathway (“Motion”).**
  - “Motion” detection: <200ms
  - “Details recognition: 400~500ms

*M. J. Tovée, Current Biology, Dec. 1994*

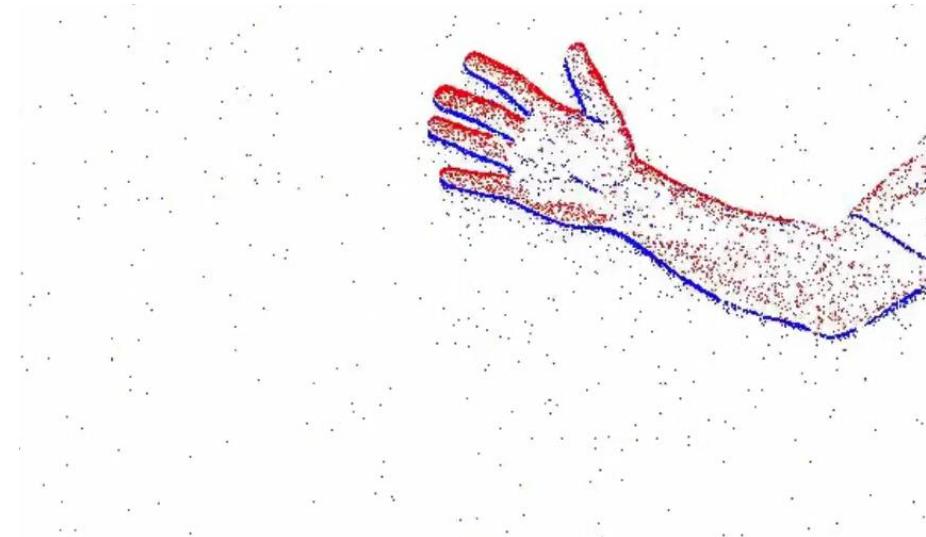
# Event camera. Static camera

- Events are caused by moving edges on the retina

Standard Camera



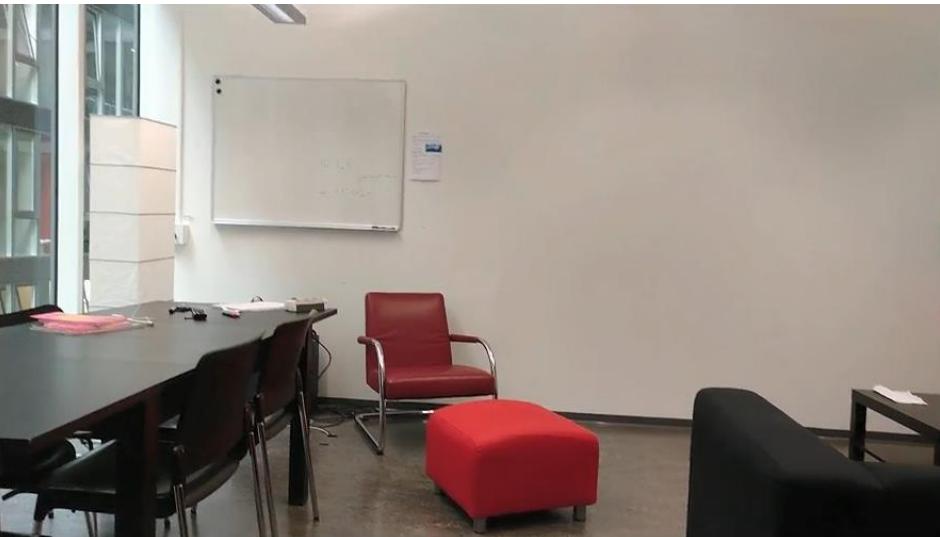
Event Camera (**ON**, **OFF** events)



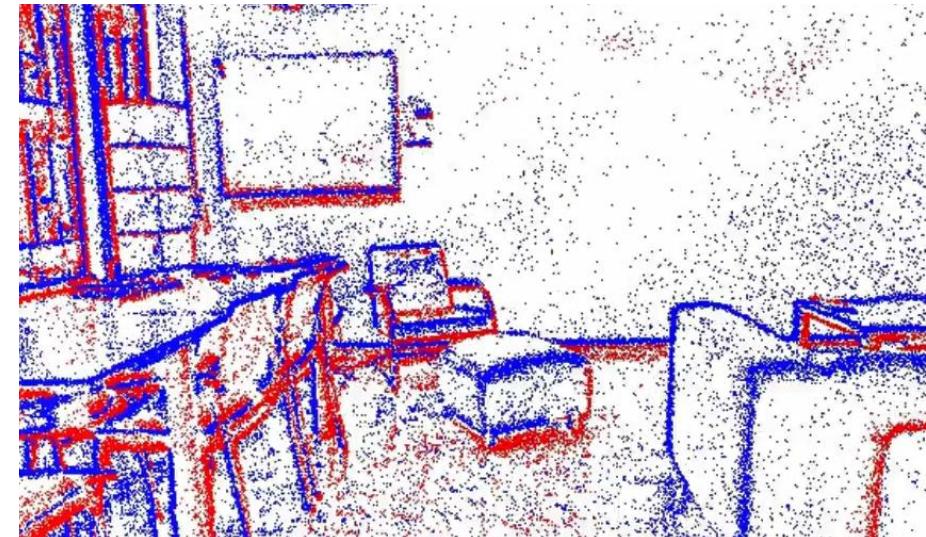
# Event camera. Moving camera

- Events are caused by moving edges on the retina
- When the camera moves, events are triggered everywhere

Standard Camera



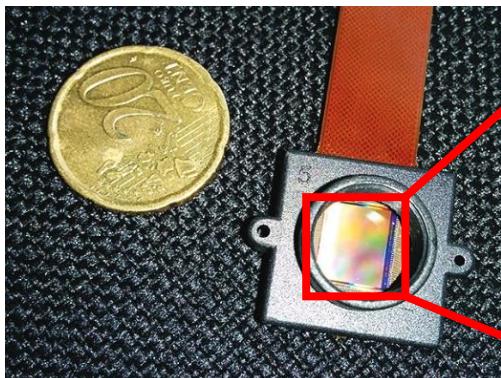
Event Camera (ON, OFF events)



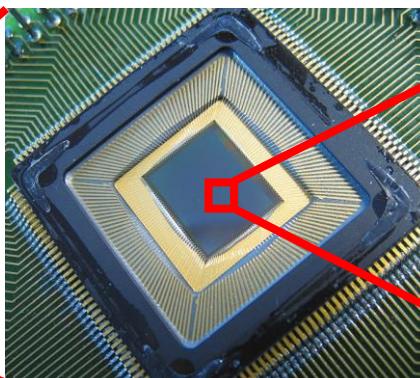
# Giving Machines Humanlike eyes

Posch et al. IEEE Spectrum is a good introductory reading to the topic

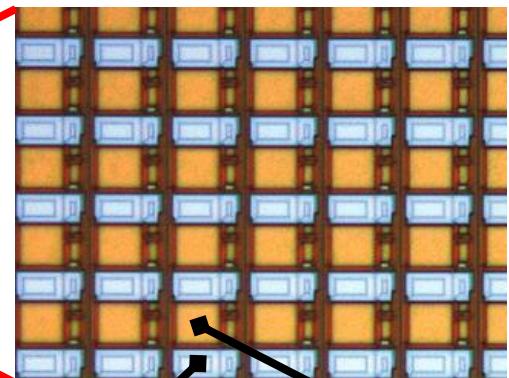
Camera (without lens)



Chip



Pixels

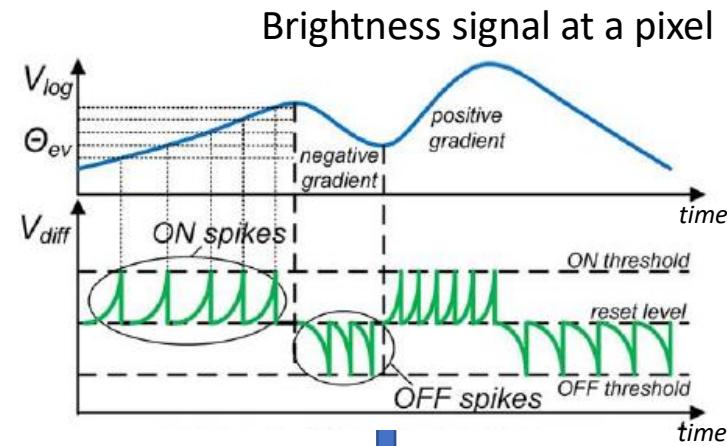
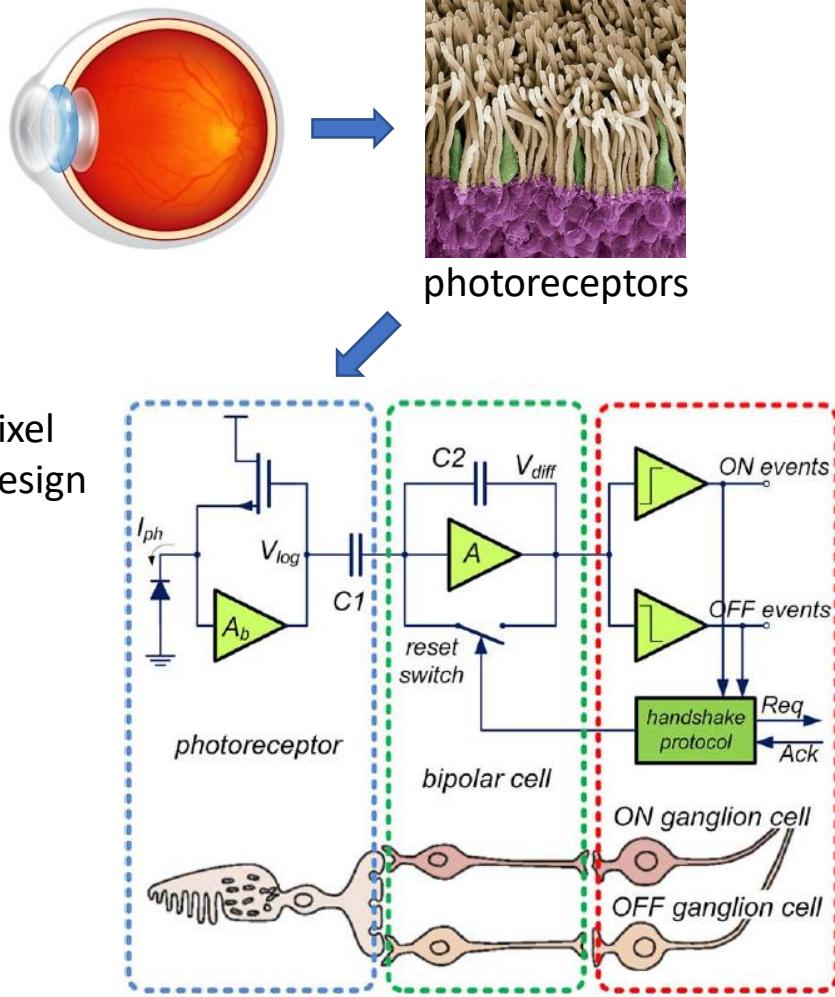


Photodiodes

Rest of the pixel  
(relative change  
detector)

# Bio-inspiration in Human Visual System

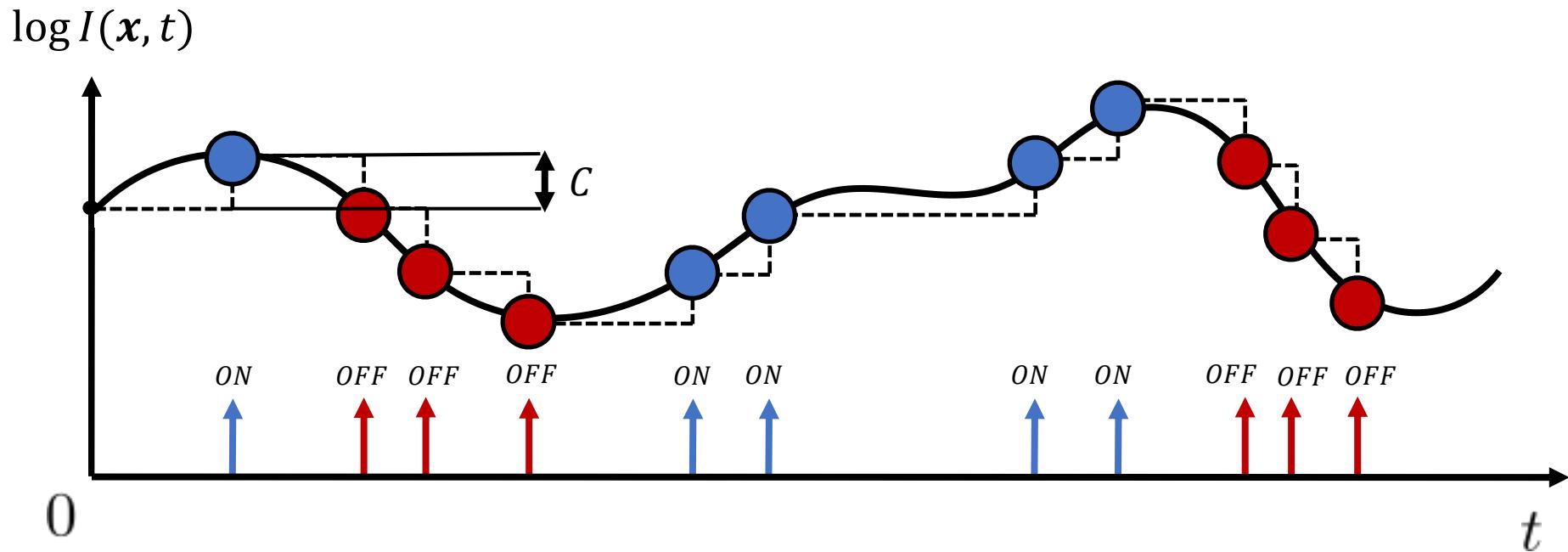
- Modeling the dorsal pathway



# Event generation model

- Consider the intensity at a **single pixel**  $x$ ...

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



Incident light at the pixel is converted (“transduced”) into a train of **asynchronous** events

# What is an event?

Each event

$$e = (x, y, t, p)$$

conveys four quantities of the brightness change:

- **Pixel coordinates**  $x = (x, y)$
- **Timestamp**  $t$  (with  $\mu s$  resolution)
- **Sign** of the brightness change:
  - Brightness increase  $\leftrightarrow$  Positive (“ON”) event.
  - Brightness decrease  $\leftrightarrow$  Negative (“OFF”) event.

Also called “**polarity**”

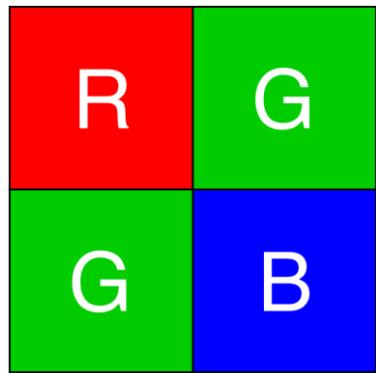
$$p = \text{sgn} \left( \frac{\partial I(x, y, t)}{\partial t} \right) \in \{+1, -1\} \text{ (binary)}$$

# Color Event Camera

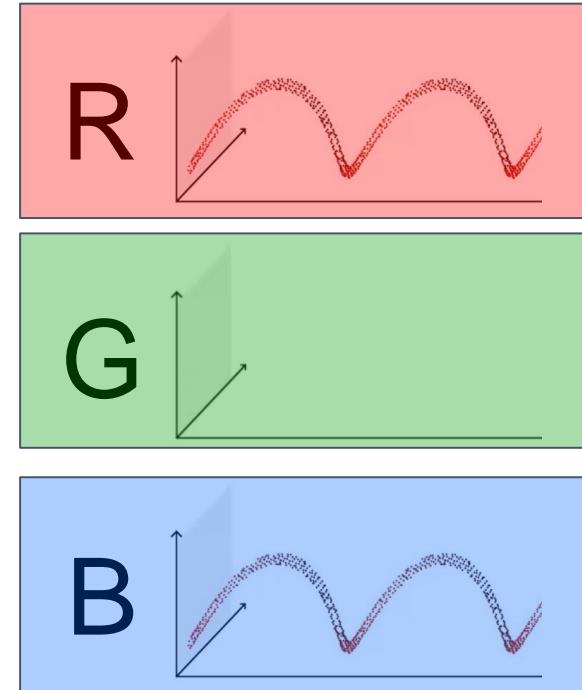
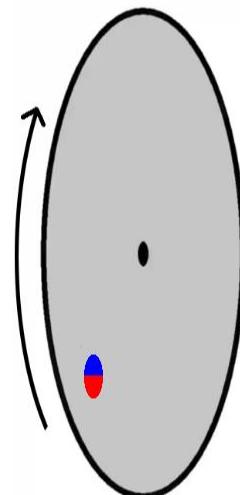
- Each pixel is sensitive to red (R), green (G) or blue (B) light
- It transmits brightness changes in each color channel



DAVIS346 Red Color



Bayer filter mosaic



# References

## Reading:

- The List of Event-based Vision Resources has a [section on sensor designs & bio-inspiration](#), with papers:
  - Posch et al., [Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output](#), Proc. IEEE, 2014. [PDF](#)
  - Posch et al. [Giving Machines Humanlike Eyes](#), IEEE Spectrum, 52(12):44-49, 2015. [PDF](#)
- Goldstein, *Sensation and Perception*, 2017. Chapter 2.
- Wikipedia: Two-streams hypothesis.  
[https://en.wikipedia.org/wiki/Two-streams\\_hypothesis](https://en.wikipedia.org/wiki/Two-streams_hypothesis)

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

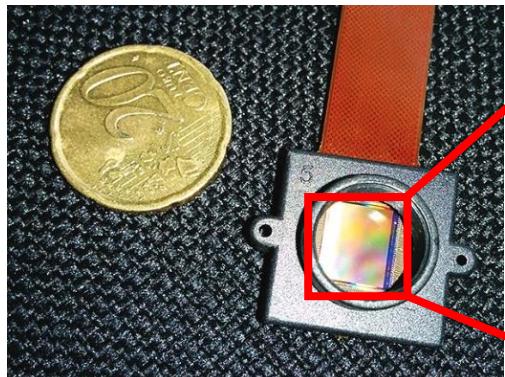
[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

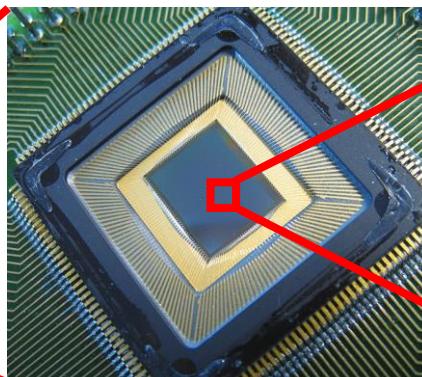
# Giving Machines Humanlike eyes

Posch et al. IEEE Spectrum is a good introductory reading to the topic

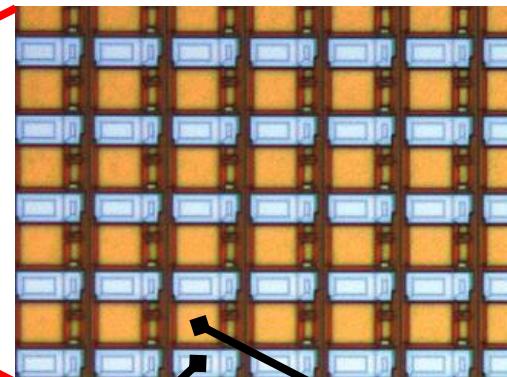
Camera (without lens)



Chip



Pixels



Input  
scene



Output events (collected into an image)  
gray = no event happened  
white = brightness increase ↑  
black = brightness decrease ↓

# Sampling based on the scene dynamics

Sky, grass and trees  
change slowly.  
(Oversampled,  
redundant in video)

The golfer, club and  
ball move fast.  
(Undersampled,  
missing information  
in video)

**Frame-based cameras**  
sample the scene based  
on an external clock  
(e.g., 25 Hz)

**Event-based cameras**  
sample the scene based  
on how intensity evolves  
at each pixel

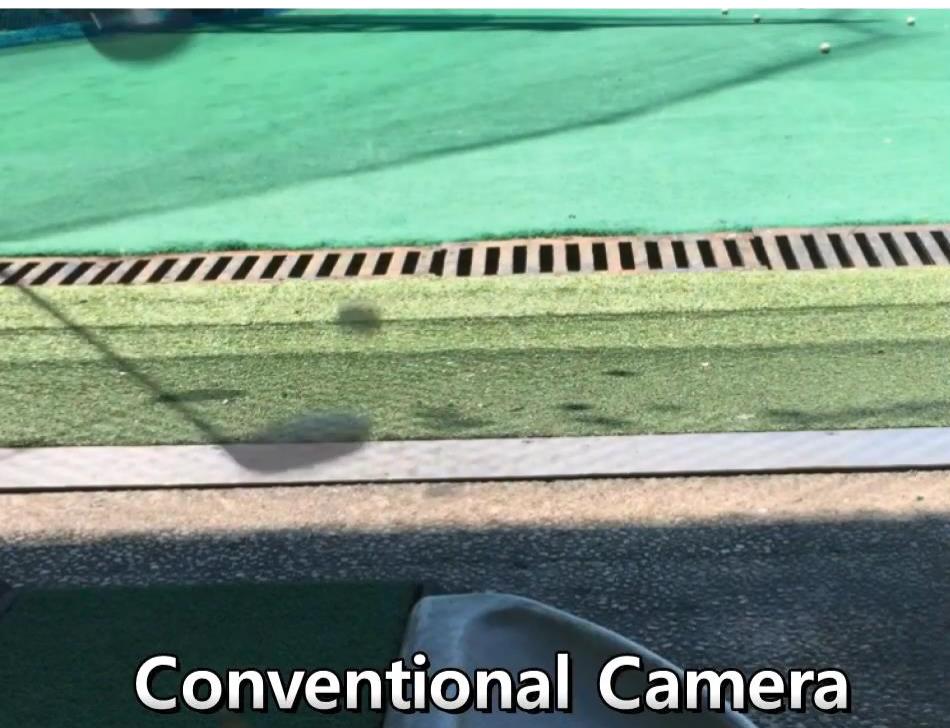


Frame-based sampling  
with individual frames  
superimposed.



# Real data

Slow motion: 0.1x (play back 10 times slower)



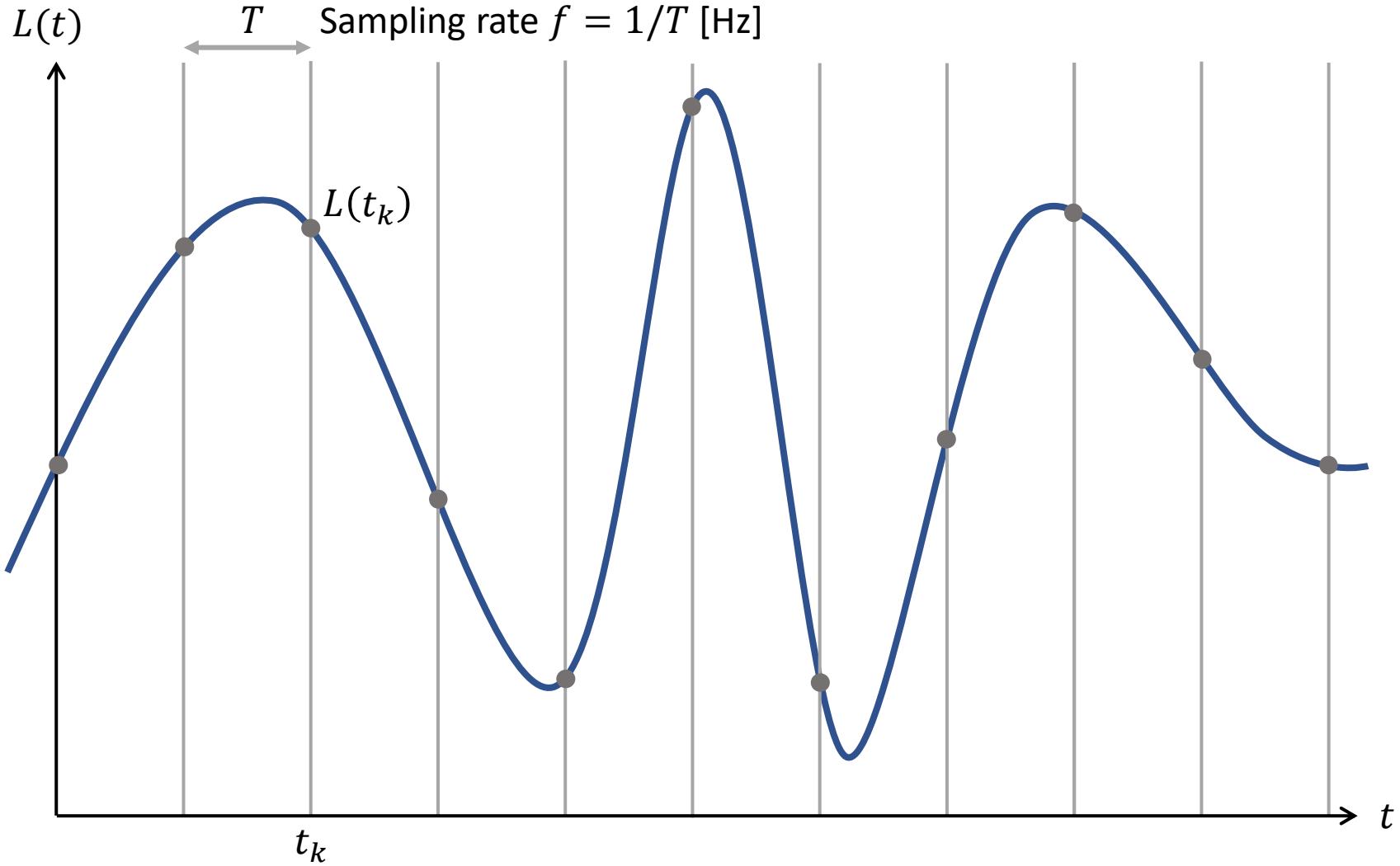
Conventional Camera



CeleX-Digital mode

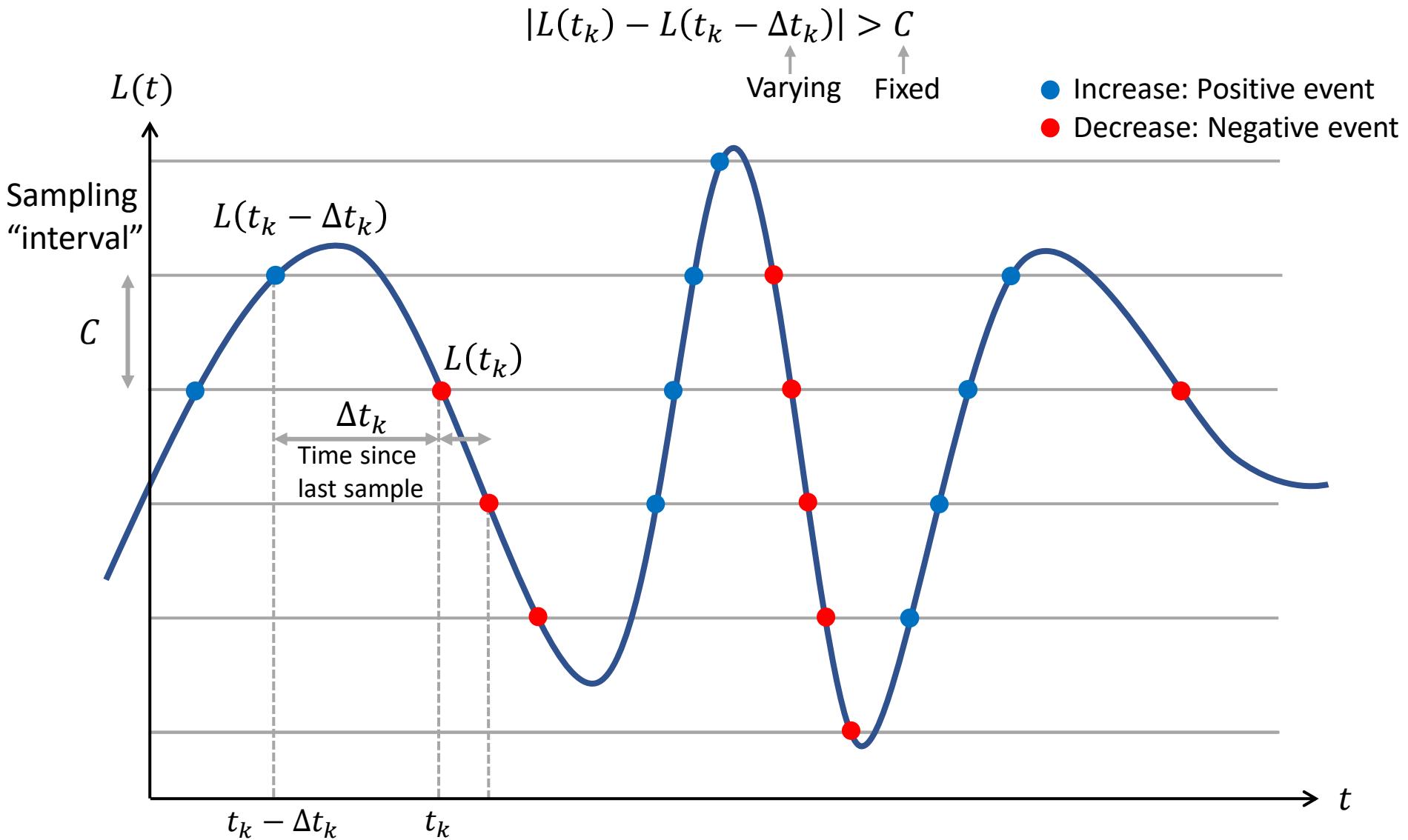
# Traditional Sampling: equispaced in time

A sample is obtained at  $t_k$  multiple of a sampling interval  $T$ :  $L(t_k)$ , with  $t_k = kT$



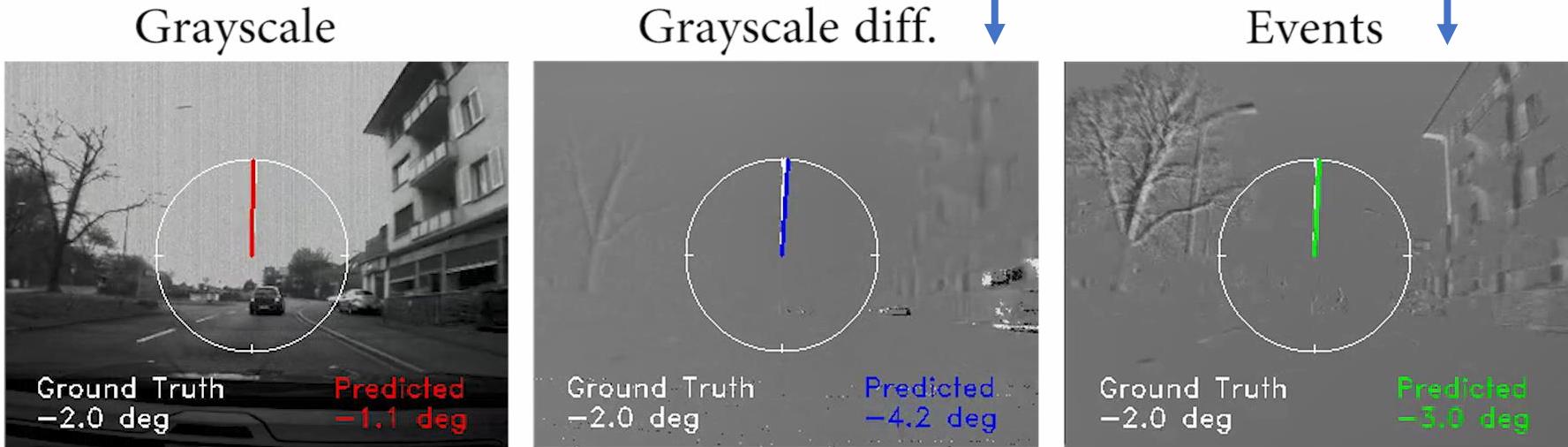
# Cross-Level Sampling: equispaced in range

A sample (event) is obtained when the signal *change* exceeds a predefined value  $C$



# Asynchronous sampling

- Event frames obtained by accumulating event polarities may look like differences of video frames, but they are not the same:
  - Different dynamic range
  - Different speed (i.e., motion blur)



# References

## Reading:

- Posch et al. [Giving Machines Humanlike Eyes](#), IEEE Spectrum, 52(12):44-49, 2015. [PDF](#)
- Mueggler et al., [The Event-Camera Dataset and Simulator](#), Int. J. Robotics Research, 2017. [PDF](#)
  - Pages 1-3 and Fig. 2
- H. Rebecq et al., [ESIM: an Open Event Camera Simulator](#), CoRL 2018.

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

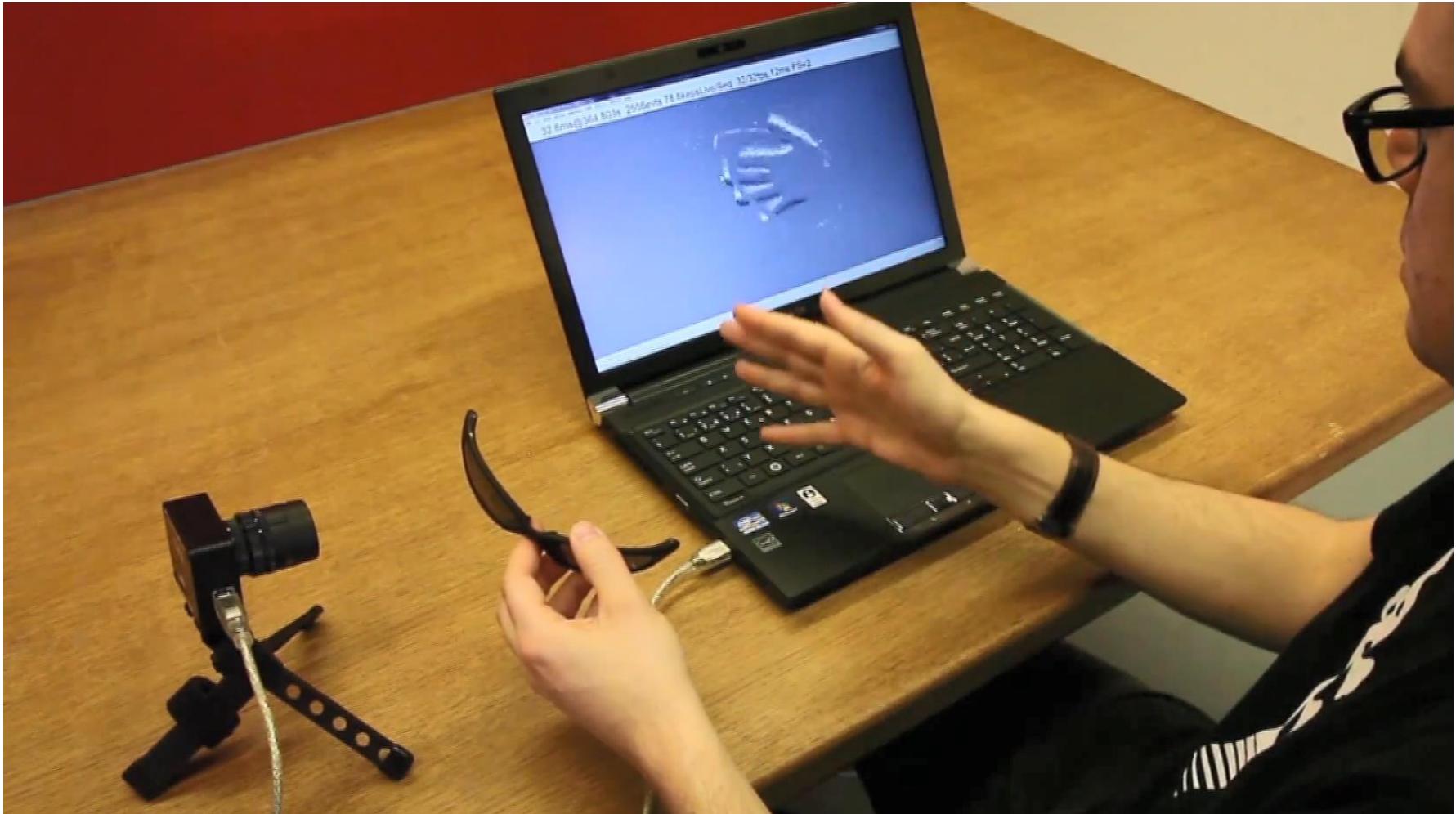
<http://www.guillermogallego.es>

# DVS with Sunglasses. High Dynamic Range



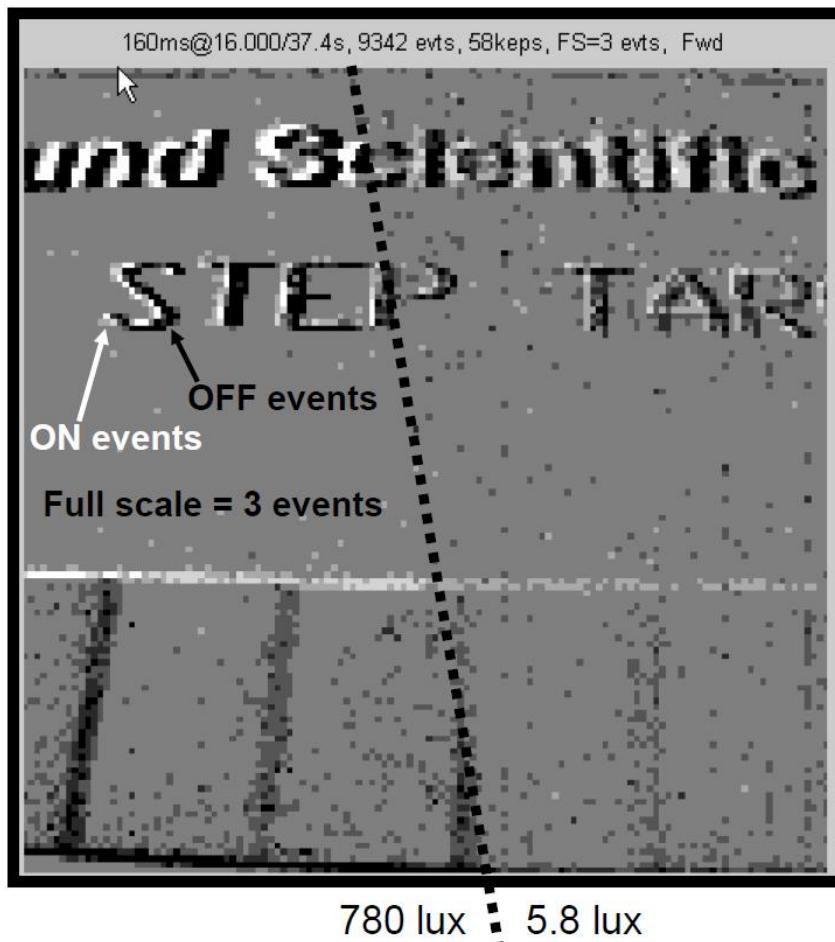
# Event camera: High Dynamic Range (HDR)

- Each pixel adapts **independently** to the light it receives and chooses its own operating point.



# Using a light meter and a density step target

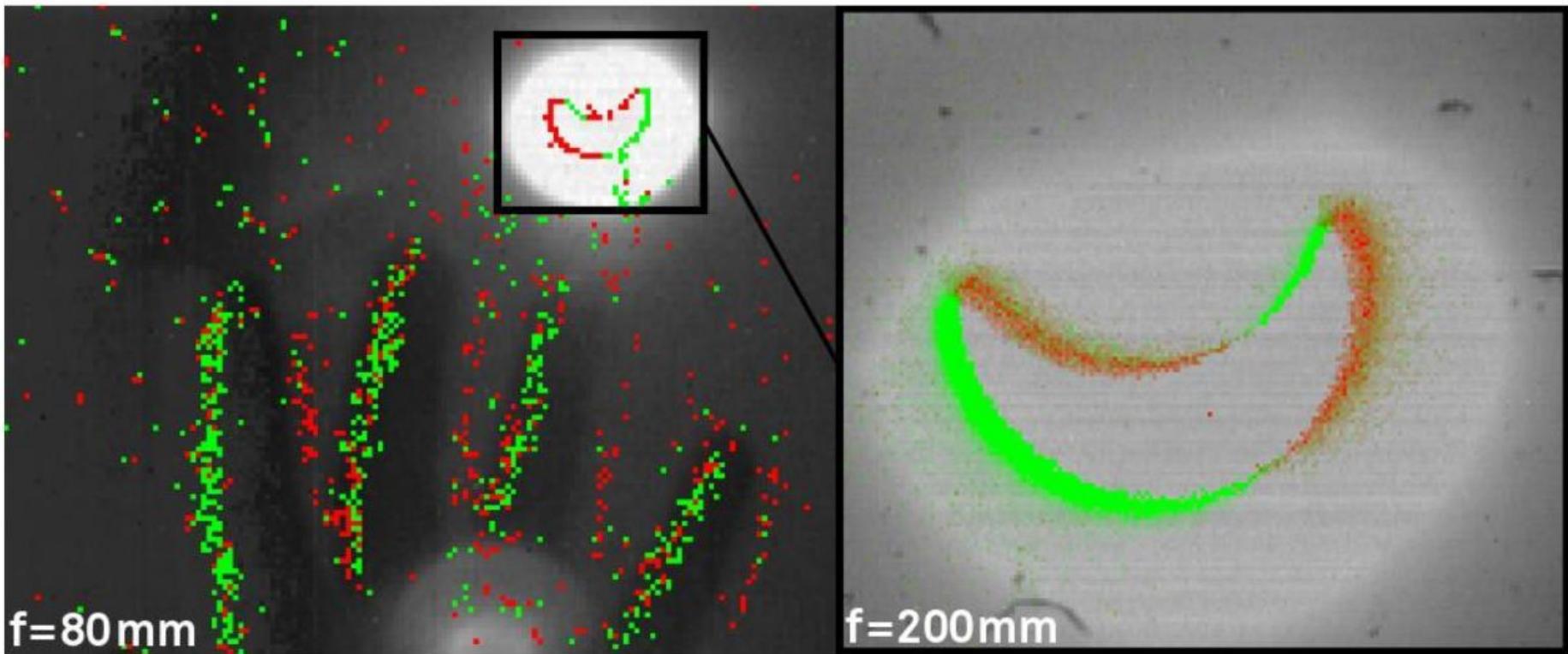
- A more quantitative approach



Edmund 0.1 density chart  
Illumination ratio=135:1

# High Dynamic Range Scene

- Ability to see very bright and very dark scenes, simultaneously
- Image of the solar eclipse (March 2015) captured by a DAVIS

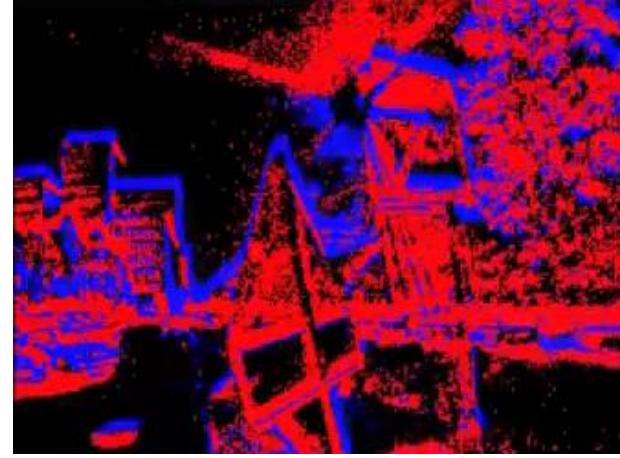


# Visual Odometry in High Dynamic Range Scenes

Standard camera



Event camera



SLAM output:  
Camera pose and 3D map



# Automotive Applications

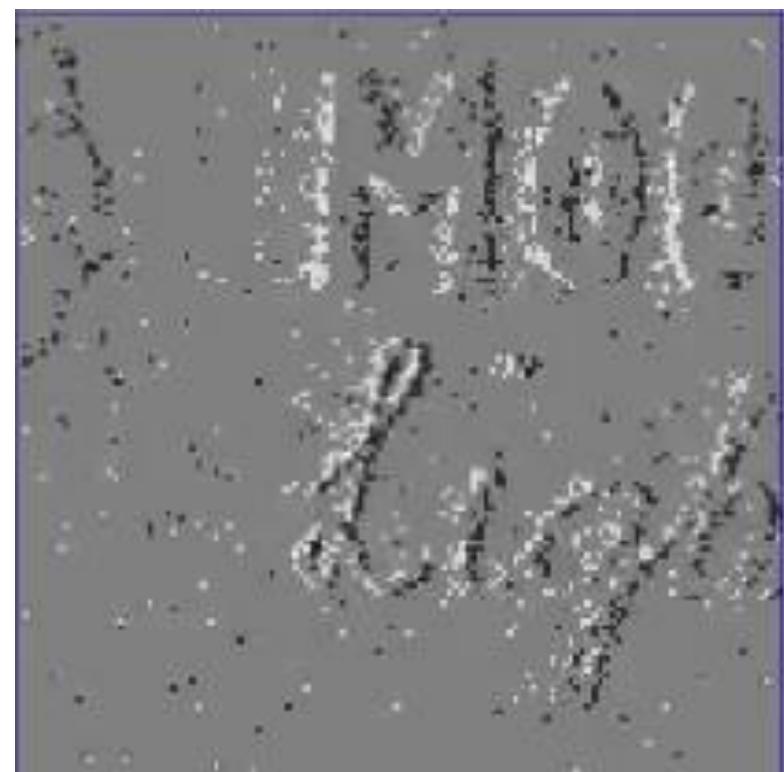
- High Dynamic Range (HDR)

Driving out of a tunnel



# Response in Low Light Conditions

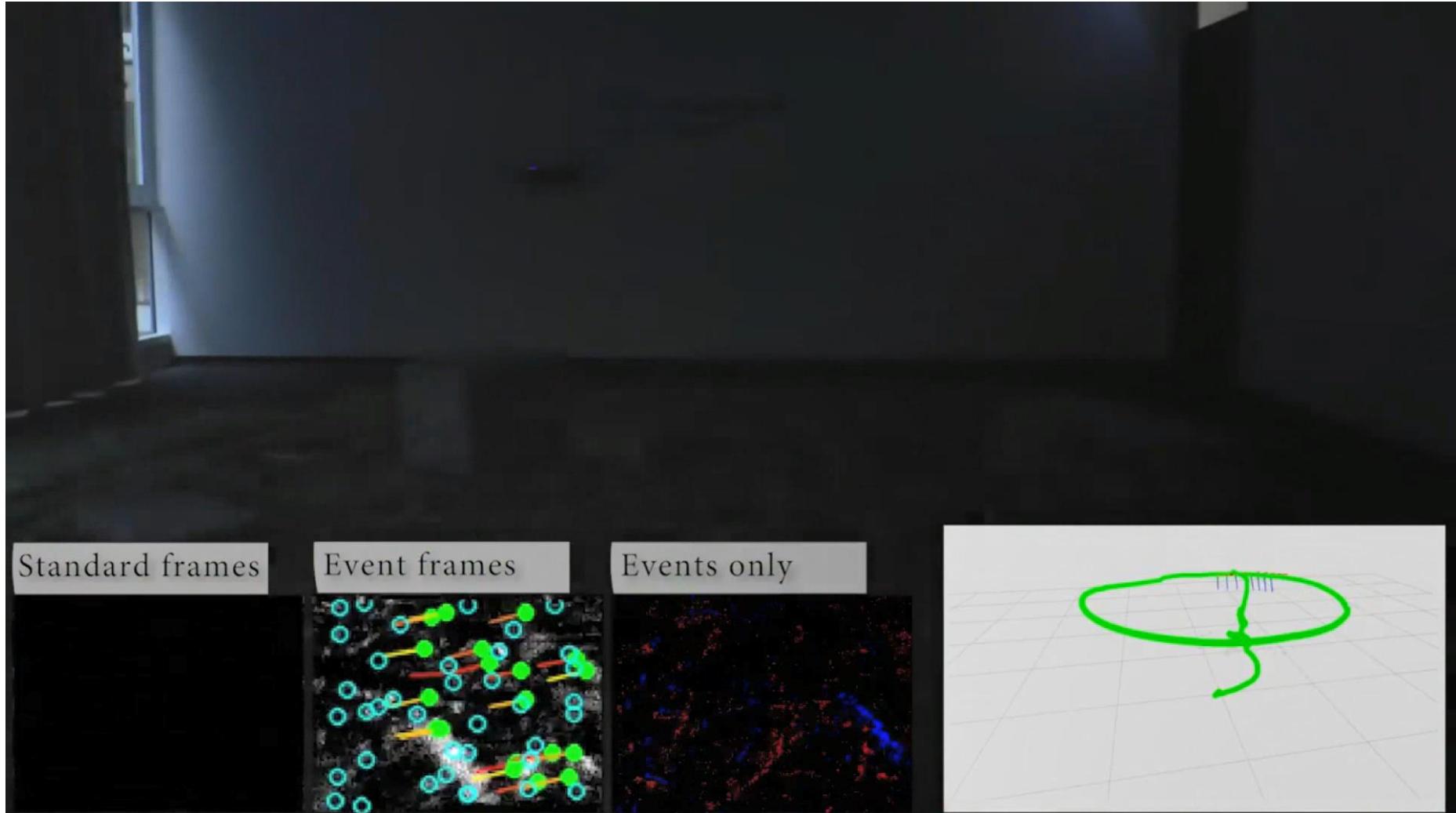
- Moving black text on white background under 3/4 moon (**< 0.1 lux**) illumination (180 ms, 8200 events)
- Minimal motion blur!



Video: <https://youtu.be/-OrnWXDJ0WI>

# Robot in Difficult Illumination Conditions

- Tightly coupled sensor fusion, fully onboard processing



# Discussion

- Understand the reasons that enable HDR:
  - Every pixel has its own operating point (There is no global shutter)
  - Pixel intensity is measured in logarithmic scale
- Investigate:
  - How is dynamic range measured?
  - Trade-off between dynamic range and contrast sensitivity (survey paper IEEE TPAMI 2020, Section 2.4).

# References

## Reading:

- Look at **experiments** that show the HDR capabilities of the camera (and the resulting vision system). Often, they are qualitative, comparing with a standard frame-based camera.
  - Lichtsteiner et al., [\*A 128x128 120dB 15μs latency asynchronous temporal contrast vision sensor\*](#), IEEE J. Solid-State Circuits, 2008. (quantitative)
  - Kim et al., [\*Simultaneous Mosaicing and Tracking with an Event Camera\*](#), BMVC 2014.
  - Bardow et al., [\*Simultaneous Optical Flow and Intensity Estimation from an Event Camera\*](#), CVPR 2016. [YouTube](#)
  - Rebecq et al., [\*EVO: A Geometric Approach to Event-based 6-DOF Parallel Tracking and Mapping in Real-time\*](#), RA-L 2016. [Youtube](#)
  - Rebecq et al., [\*High Speed and High Dynamic Range Video with an Event Camera\*](#), PAMI 2020. [YouTube](#)

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

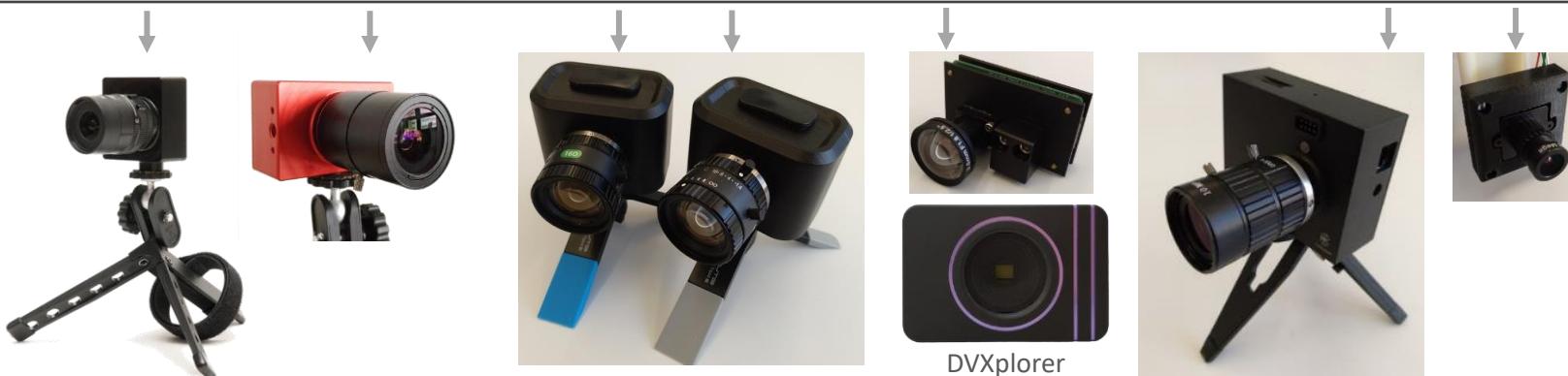
# Hardware companies developing cameras



# Actual Event-based Cameras

Comparison of commercial or prototype event cameras. Values are approximate since there is no standard measurement testbed.

Supplier	iniVation			Prophesee				Samsung			CelePixel		Insightness
Camera model	DVS128	DAVIS240	DAVIS346	ATIS	Gen3 CD	Gen3 ATIS	Gen 4 CD	DVS-Gen2	DVS-Gen3	DVS-Gen4	CeleX-IV	CeleX-V	Rino 3
Year, Reference	2008 [2]	2014 [4]	2017	2011 [3]	2017 [66]	2017 [66]	2020 [67]	2017 [5]	2018 [68]	2020 [39]	2017 [69]	2019 [70]	2018 [71]
Resolution (pixels)	128 × 128	240 × 180	346 × 260	304 × 240	640 × 480	480 × 360	1280 × 720	640 × 480	640 × 480	1280 × 960	768 × 640	1280 × 800	320 × 262
Latency (μs)	12μs @ 1klux	12μs @ 1klux	20	3	40 - 200	40 - 200	20 - 150	65 - 410	50	150	10	8	125μs @ 10lux
Dynamic range (dB)	120	120	120	143	> 120	> 120	> 124	90	90	100	90	120	> 100
Min. contrast sensitivity (%)	17	11	14.3 - 22.5	13	12	12	11	9	15	20	30	10	15
Power consumption (mW)	23	5 - 14	10 - 170	50 - 175	36 - 95	25 - 87	32 - 84	27 - 50	40	130	-	400	20-70
Chip size (mm <sup>2</sup> )	6.3 × 6	5 × 5	8 × 6	9.9 × 8.2	9.6 × 7.2	9.6 × 7.2	6.22 × 3.5	8 × 5.8	8 × 5.8	8.4 × 7.6	15.5 × 15.8	14.3 × 11.6	5.3 × 5.3
Pixel size (μm <sup>2</sup> )	40 × 40	18.5 × 18.5	18.5 × 18.5	30 × 30	15 × 15	20 × 20	4.86 × 4.86	9 × 9	9 × 9	4.95 × 4.95	18 × 18	9.8 × 9.8	13 × 13
Fill factor (%)	8.1	22	22	20	25	20	> 77	11	12	22	8.5	8	22
Supply voltage (V)	3.3	1.8 & 3.3	1.8 & 3.3	1.8 & 3.3	1.8	1.8	1.1 & 2.5	1.2 & 2.8	1.2 & 2.8	1.2 & 2.8	1.8 & 3.3	1.2 & 2.5	1.8 & 3.3
Stationary noise (ev/pix/s) at 25C	0.05	0.1	0.1	-	0.1	0.1	0.1	0.03	0.03	0.03	0.15	0.2	0.1
CMOS technology (nm)	350	180	180	180	180	180	90	90	90	65/28	180	65	180
	2P4M	1P6M MIM	1P6M MIM	1P6M	1P6M CIS	1P6M CIS	BI CIS	1P5M BSI			1P6M CIS	CIS	1P6M CIS
Grayscale output	no	yes	yes	yes	no	yes	no	no	no	no	yes	yes	yes
Grayscale dynamic range (dB)	NA	55	56.7	130	NA	> 100	NA	NA	NA	NA	90	120	50
Max. frame rate (fps)	NA	35	40	NA	NA	NA	NA	NA	NA	NA	50	100	30
Max. Bandwidth (Meps)	1	12	12	-	66	66	1066	300	600	1200	200	140	20
Interface	USB 2	USB 2	USB 3	no	USB 3	USB 3	USB 3	USB 2	USB 3	USB 3	no	no	USB 2
IMU output	no	1 kHz	1 kHz		1 kHz	1 kHz	no	no	1 kHz	no	no	no	1 kHz



Event camera hardware is getting mature

# References

- Gallego et al., [Event-based Vision: A Survey](#), TPAMI 2020, Section 2.5
- Delbruck, [The Slow but Steady Rise of the Event Camera](#), EETimes 2020
- The List of Event-based Vision Resources has sections on [devices & manufacturers](#) and [companies working on these cameras](#):
  - **inivation:** <https://inivation.com> and [SynSense](#)
  - **iniLabs:** <https://inilabs.com>
  - **Samsung Electronics** ([SmartThings Vision product](#))
  - **Prophesee** [www.prophesee.ai](http://www.prophesee.ai) (with [Sony as 2020 partner](#))
  - **Insightness** <https://www.insightness.com>  
(now **Sony Advanced Visual Sensing**)
  - **CelePixel** <https://www.celepixel.com>  
(acquired by **Will Semiconductors** in 2020, **Omnivision**)

There are many prototype event cameras that have not been commercialized. There may be more companies “under the radar” working on event-based vision technologies. Check out the partners of the above companies.

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

# Working principle and its Consequences

- Fact: “*DVS pixels respond independently and asynchronously to relative light intensity changes*”.
- Consequences (**Advantages**):

Intensity changes	Redundancy suppression Low power (efficiency) Sparse output data (low computational and storage cost)
-------------------	---

Asynchronously	High speed (low latency, high temporal resolution, very small motion blur)
----------------	---

Independently	High dynamic range (HDR)
---------------	--------------------------

# Working principle and its Consequences

- Fact: “*DVS pixels respond independently and asynchronously to relative light intensity changes*”.
- Consequences (**Challenges**):

Unfamiliar output      Cannot simply apply image-based computer vision methods

Intensity changes      “Absolute” intensity is not directly available.  
                            Novel vision algorithms need to reconsider photometric aspects (edge information, log scale, polarity)

Asynchronous & Sparse      Novel vision algorithms need to reconsider space-time processing and variable data rate

# Advantages

- Redundancy removal
  - “Sparsity” / Sparse output
  - Low bandwidth and storage costs
- Low power
  - Efficiency. Power savings due to sparsity
- High speed
  - Low latency (Asynchronous)
  - High temporal resolution
  - Almost no motion blur
- High Dynamic Range (HDR)

# Disadvantages

- Unfamiliar output
  - No “absolute” intensity information; only log-intensity changes.
  - Cannot simply apply computer vision methods for standard cameras. Require new algorithms: “Rethink Computer Vision”.
- Low spatial resolution
  - The original DVS had 128 x128 pixels
  - Latest event cameras have 1 Mpixel
  - But: the more pixels the more events need to be processed
- Event noise. Besides, it is not fully characterized.
- Software ecosystem is not as developed as for standard cameras (e.g., there is no “OpenCV” for event cameras)
- Price. Price will drop with mass production.

# Challenges

- To **unlock the potential advantages** of event cameras by designing **novel computer vision algorithms** (avoid introducing bottlenecks).
  - This means taking into account the particular characteristics of event cameras: output is asynchronous and spatially sparse, it conveys intensity changes and it is noisy.
- To **demonstrate the impact** of the resulting computer vision system to tackle...
  - ...problems that are currently difficult with other sensors
  - ...new problems.

# Ideal vs Real event data

- Real data has quite non-uniform **noise**
- Limited speed (pixel bandwidth is finite)
- Refractory period (time after event generation when no other event is produced at the pixel)
- Sensor noise has not been fully characterized
  - It is a big challenge because it depends on multiple variables
- Noise depends on amount of incident light at the pixel
- Strong edges have trailing events (“switch bouncing”)

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

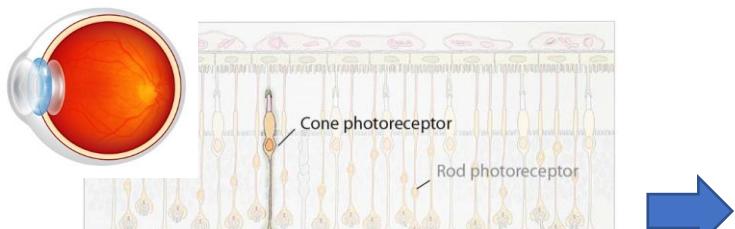
[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

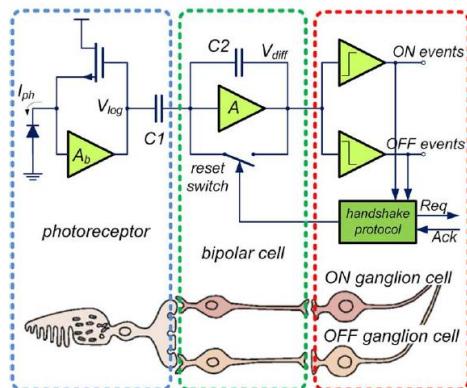
# Types of Event Cameras

# Dynamic Vision Sensor – DVS

- **Output:** events  $e = (x, y, t, p)$  representing brightness changes (i.e., temporal contrast)
- Models the **transient** visual pathway (“where” system)
- Spatial resolution: from  $128 \times 128$  pixels (DVS128 in 2008, **1<sup>st</sup> commercially available**) to recent 1 Mpixel versions (2020)
- Manufactured by iniVation, Samsung



Highlighted cells:  
simplified biological functions



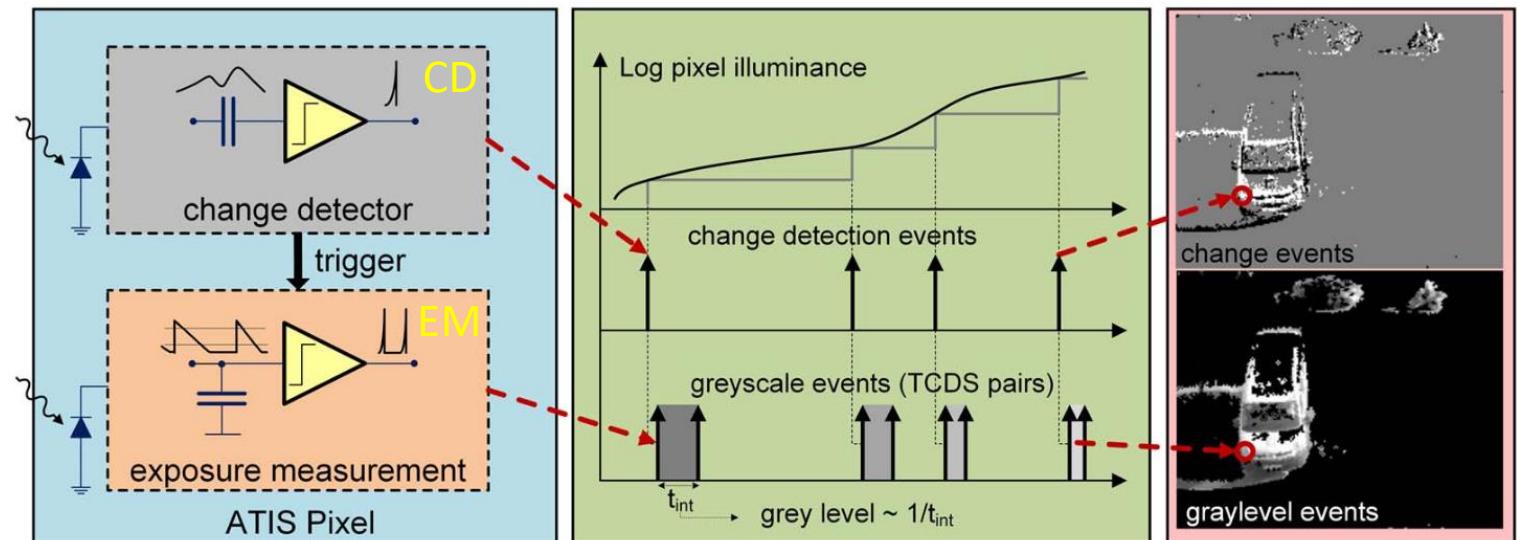
Pixel electrical design



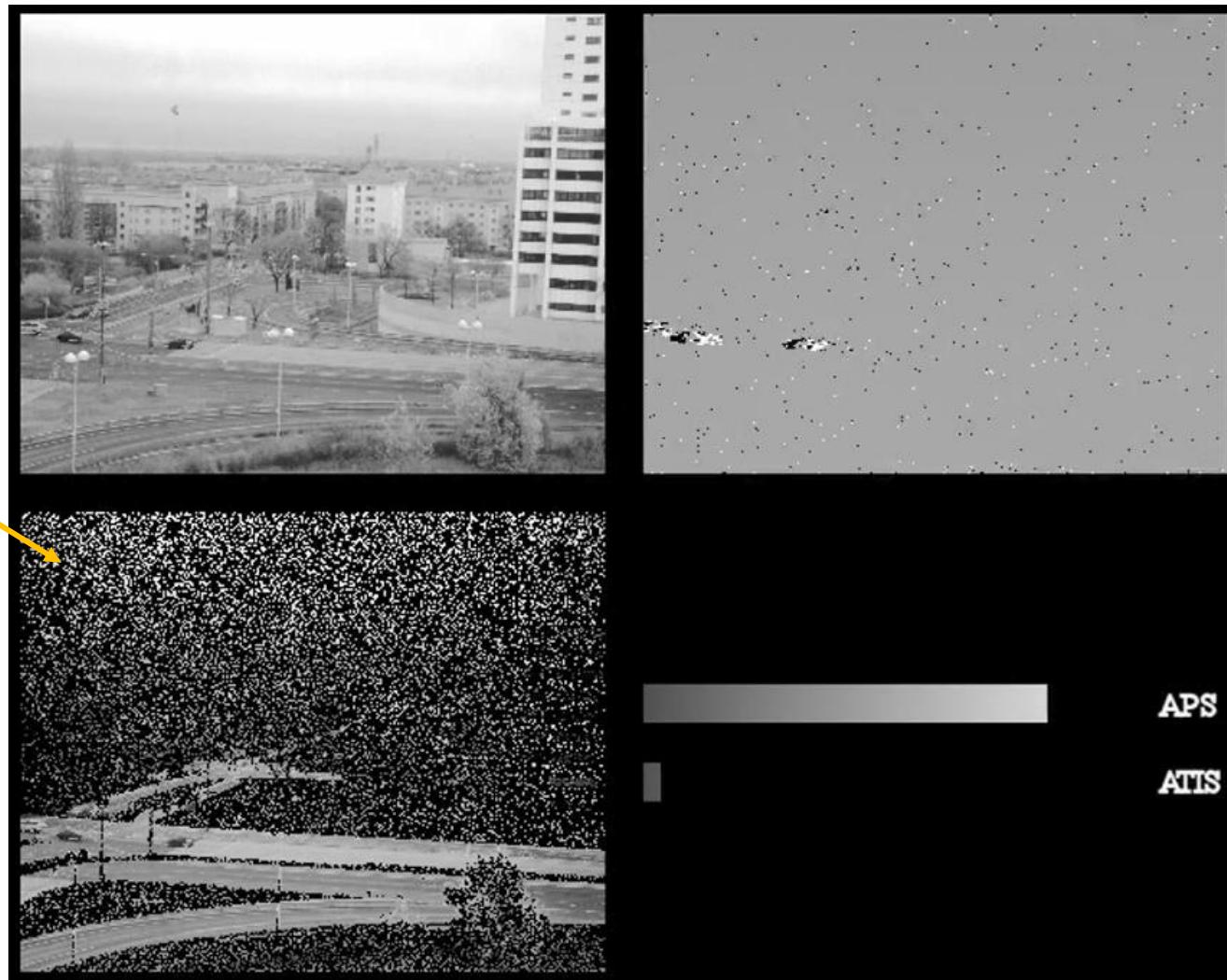
VLSI → Vision sensor

# Asynchronous Time-based Image Sensor - ATIS

- **Output:**
  - Change detection (**CD**) events (i.e., like DVS) that model the transient visual pathway (“where” system)
  - Exposure measurements (**EM**) events (grayscale events) that model the sustained visual pathway (“what” system). Intensity-encoding is time-based
- No frames: both CD and EM events are asynchronous
- Manufactured by Prophesee (Paris)



# Asynchronous Time-based Image Sensor - ATIS



# Asynchronous Time-based Image Sensor - ATIS

Pixel-level video compression by transmitting only EM grayscale events



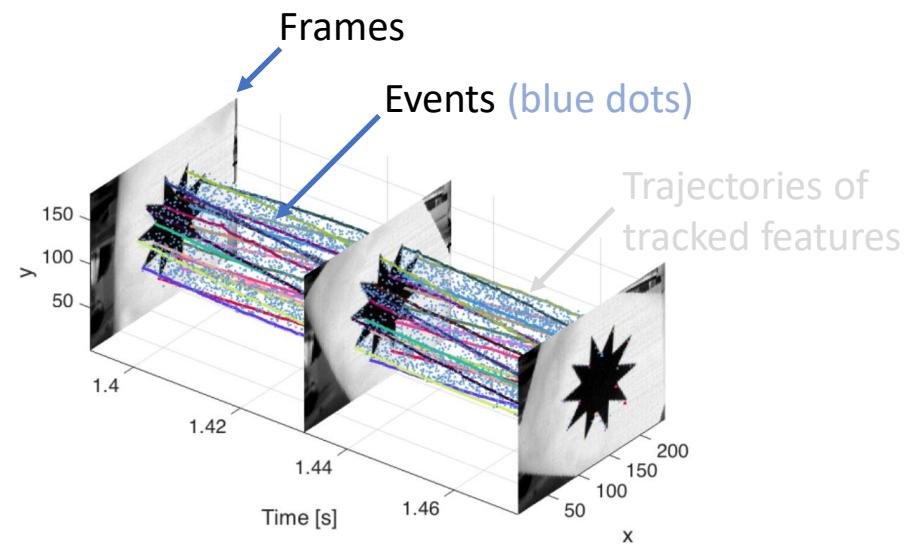
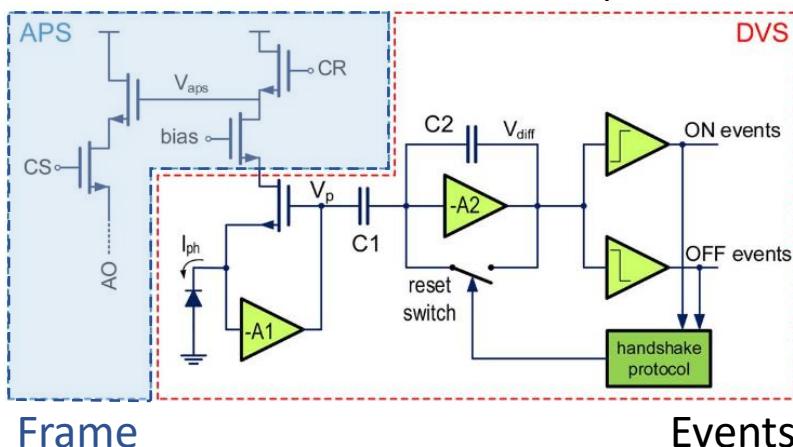
Video by X. Lagorce <https://youtu.be/3Wiw8LA8hLs>

Posch et al., [A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression ...](#), IEEE JSSC 2011.

# Dynamic Pixel and Active Vision Sensor - DAVIS

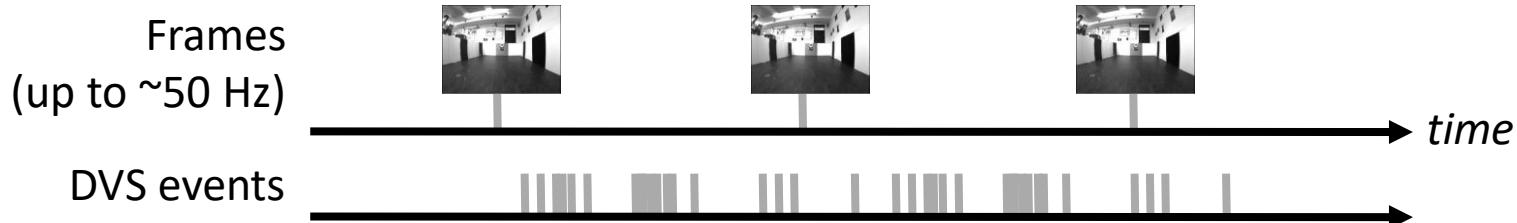
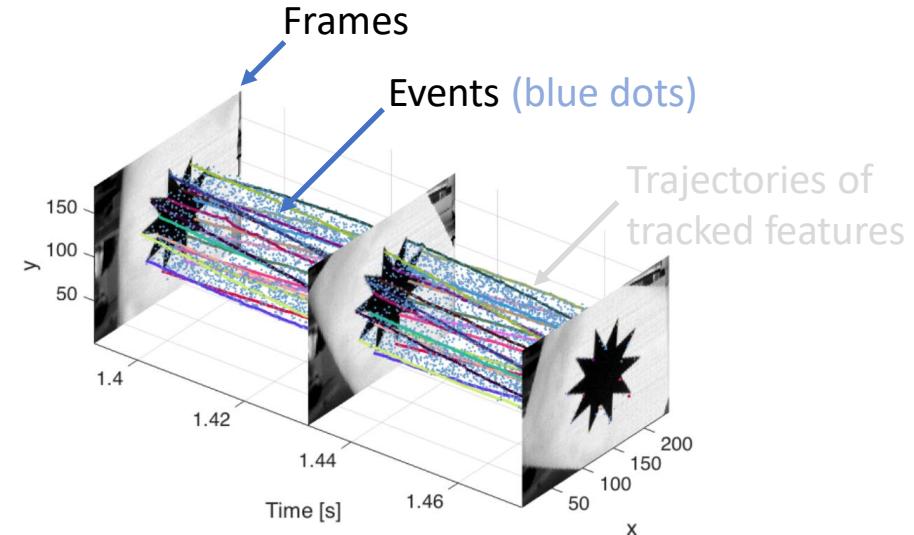
- **Output:** DVS events, standard frames and IMU data
  - Combines a DVS and a standard camera **in the same pixel array**
  - Frames (rolling or global shutter, grayscale or color) are not HDR ( $\sim 55$  dB)
  - Frames resemble the information in the “what” visual pathway
- Spatial resolution: from  $240 \times 180$  pixels (DAVIS240C) to  $640 \times 480$  pixels (VGA)
- Manufactured by iniVation and Insightness

Add a few transistors to each DVS pixel...



# Dynamic Pixel and Active Vision Sensor - DAVIS

- **Output:** DVS events, standard frames and IMU data
  - Combines a DVS and a standard camera **in the same pixel array**
  - Frames (rolling or global shutter, grayscale or color) are not HDR ( $\sim 55$  dB)
  - Frames resemble the information in the “what” visual pathway



# Very Short Summary

	DVS (2008)	DAVIS (2014)	ATIS (2011)
<b>Events</b> (change detection)	Yes	Yes	Yes
<b>Grayscale</b>	No	Frames (~ 55dB)	EM events (HDR) Time-based encoding
HDR, Latency, Power	All similar		
Manufacturer	inViation, Samsung	inViation, Insightness	Prophesee

# Types of Events

# Types of Events

- DVS or “change detection (CD)” event  $e = (x, y, t, p)$
- ATIS grayscale events (Exposure Measurement)  $e = (x, y, t, L)$
- Event from a **stereo** camera
  - Include camera index (left/right):  $e = (x, y, t, p, l/r)$
- **Color** events
  - Color DVS (include channel index)  $e = (x, y, t, p, channel)$
  - Color ATIS (exposure measurement)  $e = (x, y, t, \textcolor{red}{r}, \textcolor{green}{g}, \textcolor{blue}{b})$
- **Augmented** events  $e = (x, y, t, p, extra)$ 
  - Additional information given by the output of some algorithm
  - Lifetime estimation: normal optical flow, lifetime
  - Sensor fusion (event camera + depth): 3D depth of each event

# The meaning of “Event”

- Events can signal **any kind of information with associated place and time** ( $x, y, t$ ): intensity, local spatial contrast, etc.
- Ideally, events represent **meaningful information** to reduce data rate and therefore decrease demands on bandwidth, memory, and computer power for transmission, storage and processing.



- The term has evolved from address-event (AER) representation. Today “event” mostly refers to “brightness changes” output by DVSs.

# Where could this be going?

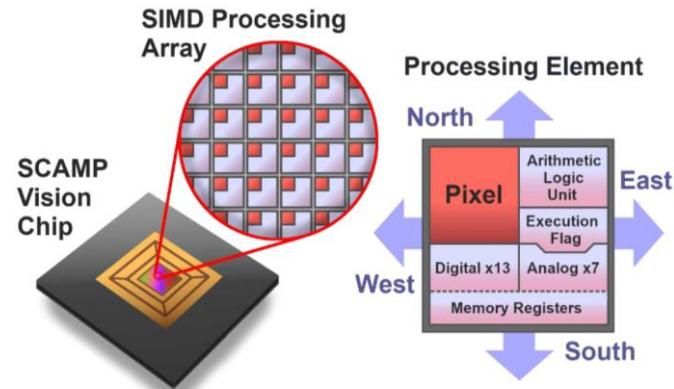
- **Perform early visual processing on chip, near the image plane**  
(in analog and transmit only “meaningful information” off the chip)

- Increasing **pixel complexity**:

- Standard Pixel → DVS pixel → DAVIS pixel  
→ Pixel Processor Arrays (PPAs)

- PPA

- One mini-processor per pixel (SCAMP-5)
  - Dedicates more area to processing, less for transducing light (photodiode). Light conversion efficiency could be improved with a stacked approach.
  - It is a prototype sensor (academic), not mass-produced.
  - We are entering the realm of computational imaging/photography, co-designing the visual pipeline end-to-end.



# References

## Reading:

- Journal papers on main sensor designs:
  - **DVS**: Lichtsteiner et al., [\*A 128x128 120dB 15μs latency asynchronous temporal contrast vision sensor\*](#), IEEE J. Solid-State Circuits, 2008.
  - **ATIS**: Posch et al., [\*A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS\*](#), IEEE J. Solid-State Circuits, 2011.
  - **DAVIS**: Brandli et al., [\*A 240x180 130 dB 3 μs Latency Global Shutter Spatiotemporal Vision Sensor\*](#), IEEE J. Solid-State Circuits, 2014.
- Papers comparing the sensors:
  - Posch et al., [\*Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output\*](#), Proc. IEEE, 2014. [PDF](#)
  - T. Delbrück, [\*The Slow but Steady Rise of the Event Camera\*](#), EE Times 2020.
  - **Section 2.1** of Gallego et al., [\*Event-based Vision: A Survey\*](#), IEEE TPAMI 2020.

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

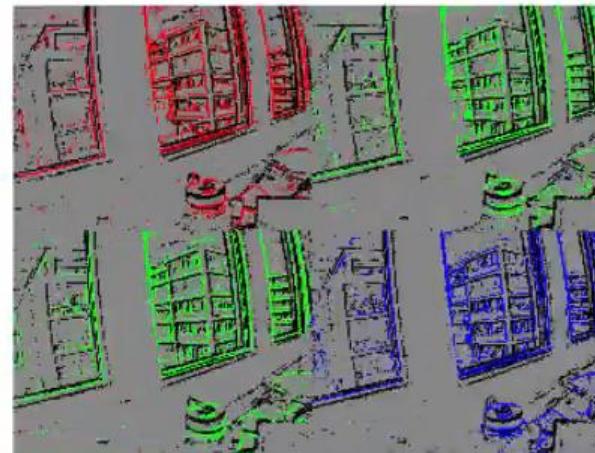
<http://www.guillermogallego.es>

# Color Event Cameras

# Example: DVS with RGB Color Filter Array



DAVIS frames  
(narrow dynamic range)



Events



High Dynamic Range?

No

Yes

A blue double-headed vertical arrow labeled "No" at the top and "Yes" at the bottom, positioned between the DAVIS frames and the Events visualization.



Reconstructed intensity (with E2VID)

# List of (prototype) cameras

- **Color-DAVIS (using Color Filter Arrays - CFAs)**
  - **C-DAVIS:** RGBW-VGA color frames and QVGA (320 x 240 pixels) monochrome events. ISCAS 2015
  - **SDAVIS192:** RGBW color filter array for frames and events (192 x 188 pixels, 1% contrast sensitivity). TBCAS 2018
  - **DAVIS346 color:** RGB color filter array for frames and events (346 x 260 pixels, 15% contrast sensitivity). TCAS-II 2018
- **Color ATIS (using beam splitters)**
  - RGB events (exposure measurement - EM)

# RGBW color SDAVIS192

- Each pixel is sensitive to red (R), green (G), blue (B) or white (W = unfiltered) light
- RGBW color filter array for frames and events (192 x 188 pixels, 1% contrast sensitivity)

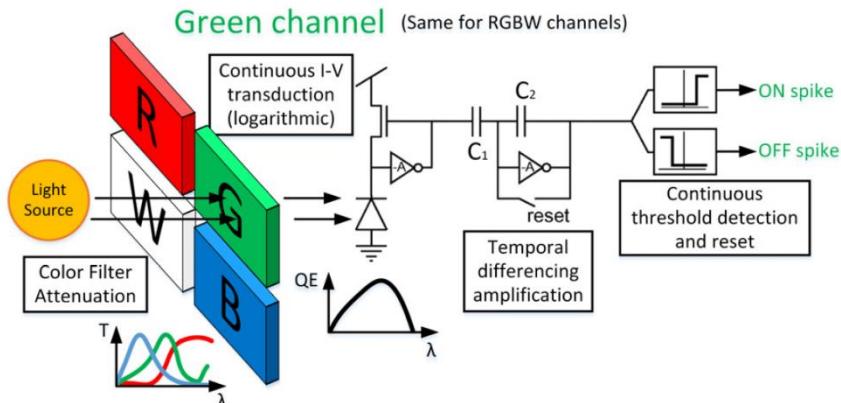
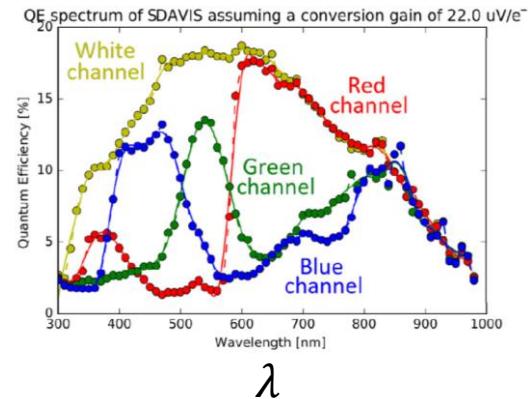


Fig. 1 Simplified block diagram of DVS pixels with color filters. Graphs show CFA transmission coefficients T vs wavelength  $\lambda$  and photodiode QE vs  $\lambda$ .



Scene

RGBW Events

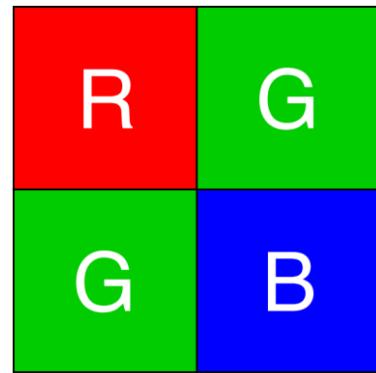
Reconstructed  
intensity

# DAVIS346 color

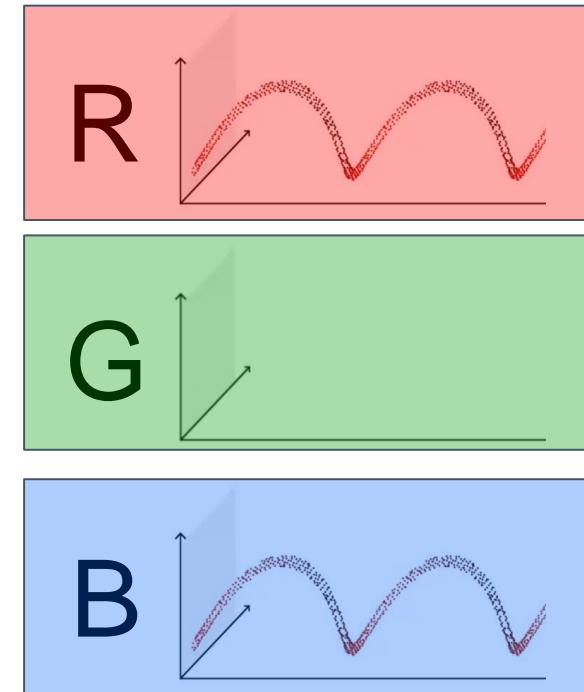
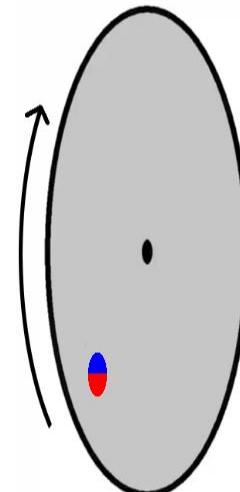
- Each pixel is sensitive to red (R), green (G) or blue (B) light
- It transmits brightness changes in each color channel



DAVIS346 Red Color

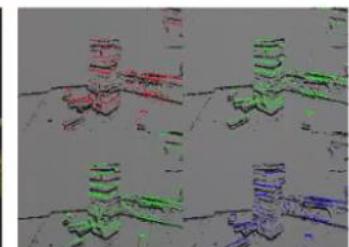
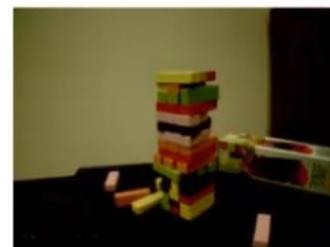
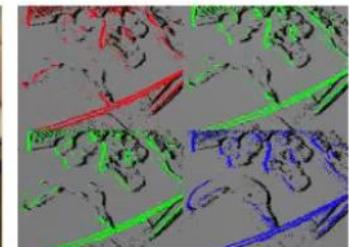


Bayer filter mosaic



# Color Event Camera Dataset

- 50 minutes footage of DAVIS frames and color events.
- Wide variety of scenes
- Wide variety of lighting: from low light (0.8 lux) to direct sunlight (10,000 lux)
- HDR scenes
- High speed, 6-DOF motions
- Recorded with a DAVIS346 color

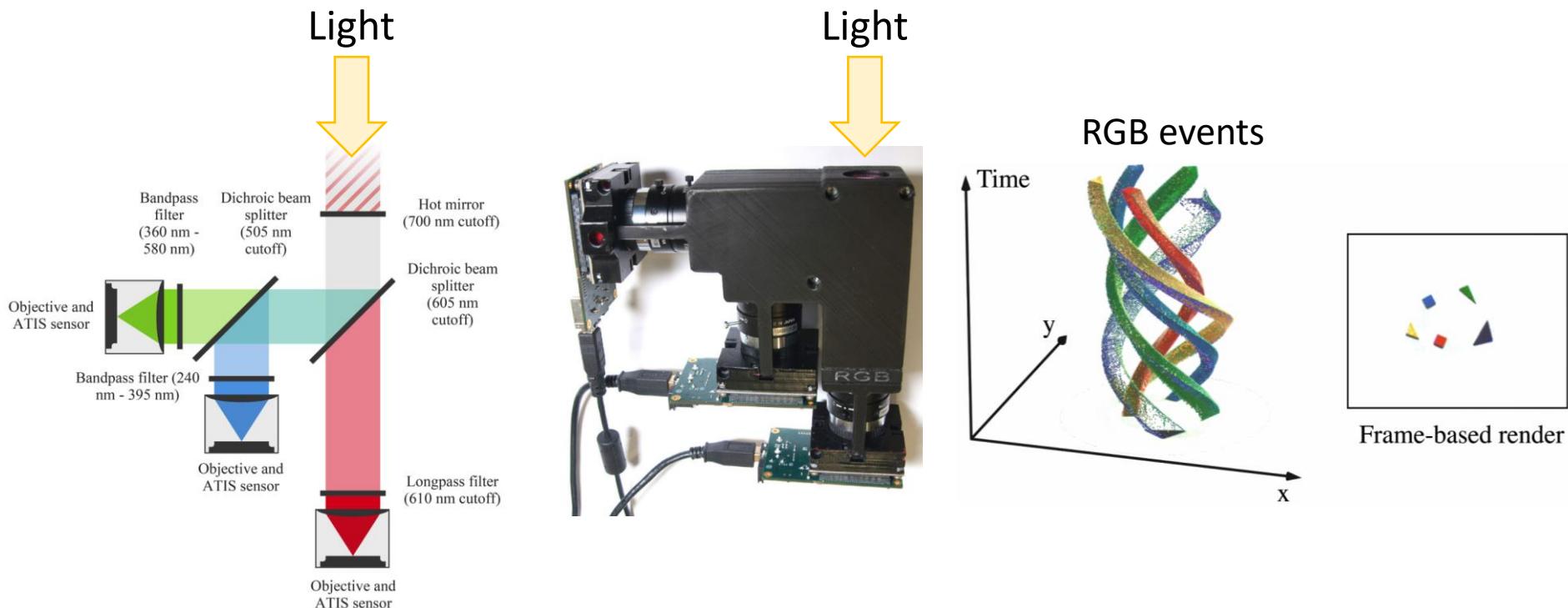


DAVIS frame

Events

# ATIS Color Camera

- 3 ATIS cameras, 3 color filters and 2 beam splitters
- The cameras share the same field of view (FOV)



Using ATISs, grayscale events (exposure measurement) become **RGB events**:  $e = (x, y, t, r, g, b)$

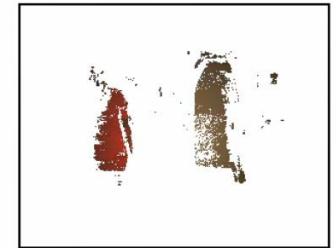
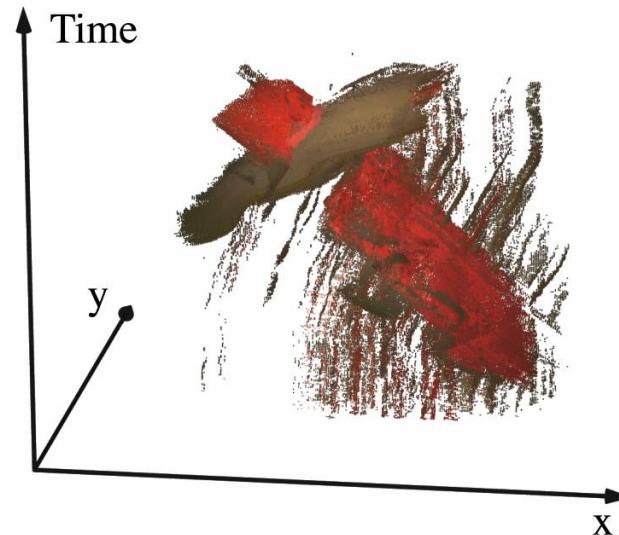
# Applications

- Intensity reconstruction (HDR, high-speed...)
- Object recognition improves with color
- Microscopy imaging (neurons sensitive to calcium)

Object detection / recognition



Segmentation using color cues



Frame-based render

# References

## Reading:

- SDAVIS192:
  - Moeys et al., [\*A Sensitive Dynamic and Active Pixel Vision Sensor for Color or Neural Imaging Applications\*](#), TBCAS 2018
  - Moeys et al., [\*Color Temporal Contrast Sensitivity in Dynamic Vision Sensors\*](#), ISCAS 2017.  
Check out the supplementary material (3 videos in the paper references)
- DAVIS346 color:
  - Taverni et al., [\*Front and back illuminated Dynamic and Active Pixel Vision Sensors comparison\*](#), TCAS-II 2018
- Color ATIS:
  - Marcireau et al., [\*Event-Based Color Segmentation With a High Dynamic Range Sensor\*](#). Front. Neurosci., 2018. Check out the supplementary material (3 videos)

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

# Event Representations

Welcome to the Zoo

# A Zoo of Representations

- Individual events (filters, SNNs...)
- Point sets on image plane
- 3D point set
- Event frame (2D grid)
  - Histogram
  - Time surface
  - Motion-compensated event frames (given motion hypothesis)
- Voxel grid (3D histograms)
- Reconstructed intensity / brightness images
- ...

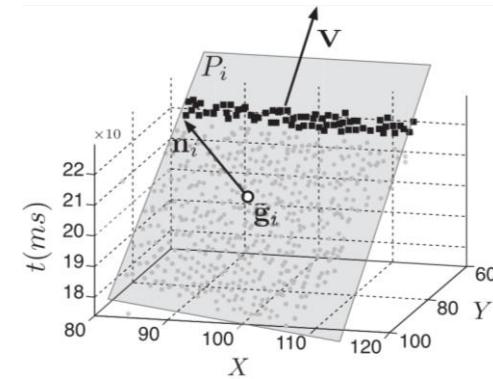
# Point sets

- **Individual Events**

- An event  $e_k = (x_k, t_k, p_k)$  is represented by its (four) numbers.
- This representation is used in event-by-event processing methods, such as filters and Spiking Neural Networks (SNNs).

- **3D Point set**

- Treat events as points in space-time  $R^3$ .  
Think geometrically in terms of point “clouds”
- **Pros:** Preserves space-time information
- **Cons:** Introduces some latency and might discard polarity

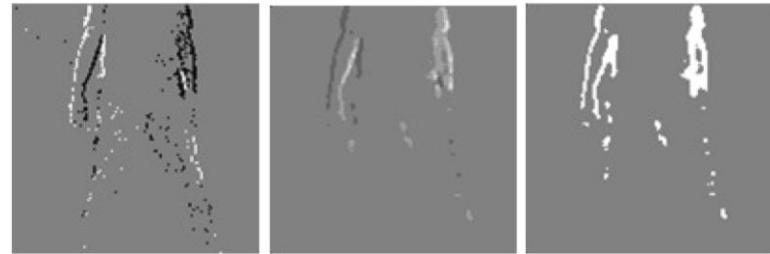


- **Evolving point set on the image plane**

- Treat events as samples of a time-evolving “shape” on the image plane. Intuitive since events are caused by moving edges.

# Event Frames

- Events  $\mathcal{E} = \{e_k\}_{k=1}^{N_e}$  in a space-time neighborhood are **converted** into (2D) images
- **Types:**
  - Histograms of events (pixelwise)
  - Balance of event polarities
  - Saturated histogram (e.g., ternary image)
  - Time surfaces (they are 2D images after all); discuss separately
- Event **selection** (sliding window):
  - Constant time: “all events in a time interval of given size  $T$ ”
  - Constant number of events  $N_e$
  - Adaptive Area-Event-number
  - Other strategy



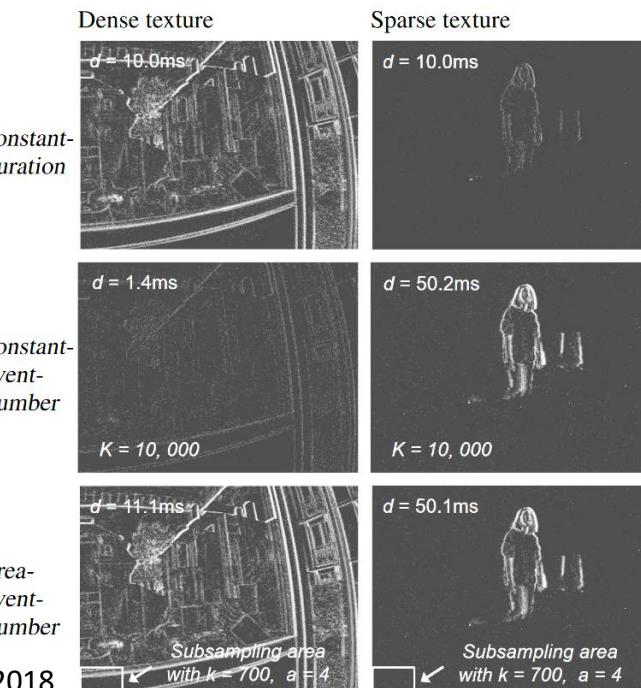
Kogler et al. ICVS 2009  
“Address-event to frame converter”

# Event Frames

- **Advantages:** Why do they have such a high impact?
  - **Compatible** with conventional computer vision: convert the unfamiliar event output to the familiar one (images) and **re-utilize** methods from standard cameras.
  - Events are caused by moving edges, so frames have an **intuitive** and **informative** interpretation: “edge maps”. Edges convey a lot of the information of a scene.
  - They are good as a **baseline** of what is achievable.
  - Frames are HDR and may be asynchronous

- **Disadvantages:**

- Not the event-based paradigm
  - **Quantizes** the event timestamps
  - **Power** is spent in creating frames
  - Some **latency** is also introduced
- How many events to use?  
Key parameter; may be difficult to tune



# Brightness Increment Images

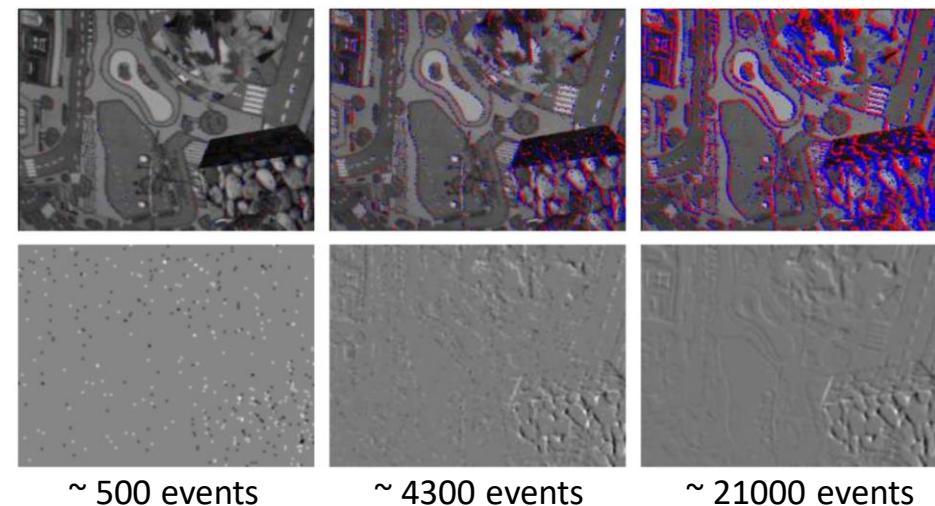
Obtained by accumulating event polarities, pixelwise.

- **Advantages:**

- Very intuitive interpretation  $\Delta L(x)$  images, like the type of images obtained by subtracting two video frames, related to “brightness constancy” equation.
- Polarity is used

- Usage cases:

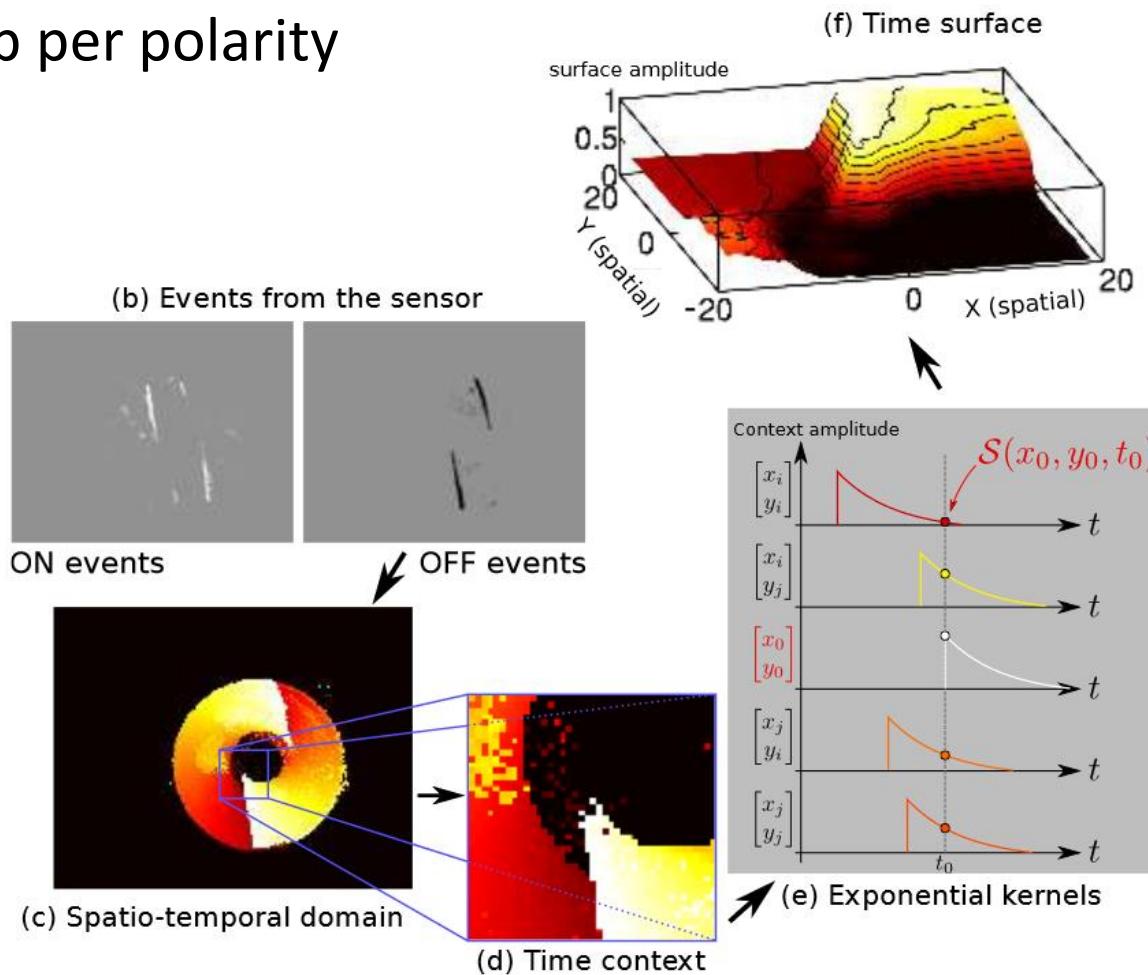
- Stereo depth estimation
- Camera pose estimation
- Optical Flow estimation
- Grayscale frame prediction
- ...



Bryner et al. ICRA 2019

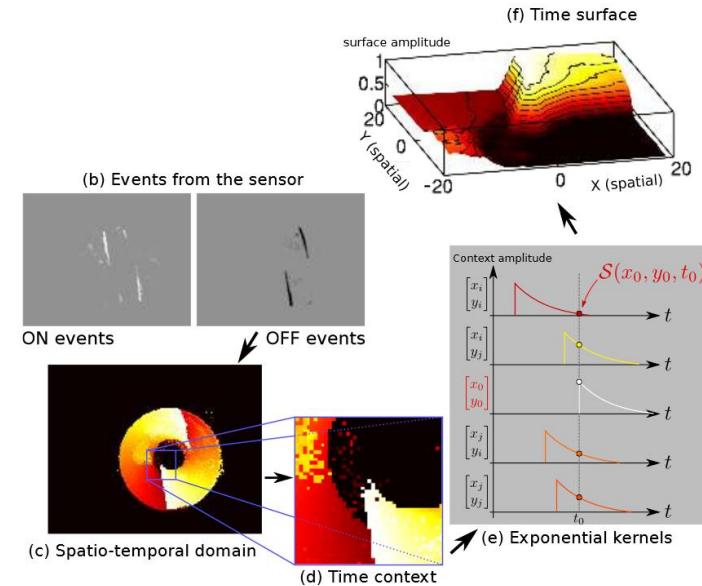
# Time Surfaces

- A time map / image
  - Pixel value =  $f$  (time of last event)
  - One map per polarity



# Time Surfaces

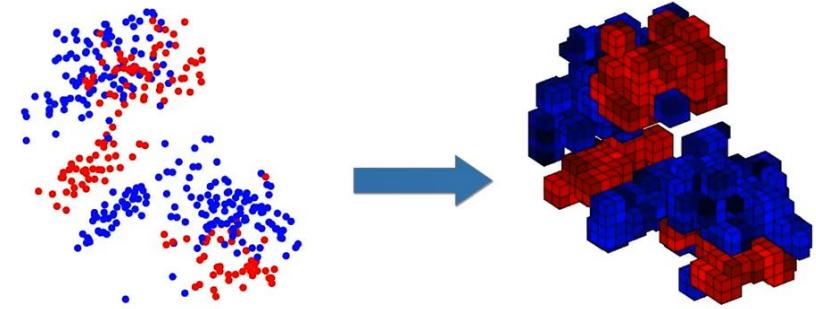
- A time map / image
  - Pixel value =  $f$  (time of last event)
  - Separate by polarity
  - Kernels may emphasize recent events
- Advantages:
  - Expose rich temporal information of events
  - Intuitive: intensity is a function of motion history
  - Can be robust to noise by local filtering (Sironi et al. CVPR 2018)
  - Asynchronous update with every event
  - Compatible with conventional computer vision
- Disadvantages:
  - Only one value per pixel even if multiple events on same pixel
  - Not good for textured scenes (pixel overwrite frequently)



Lagorce et al. PAMI 2015

# Voxel Grids

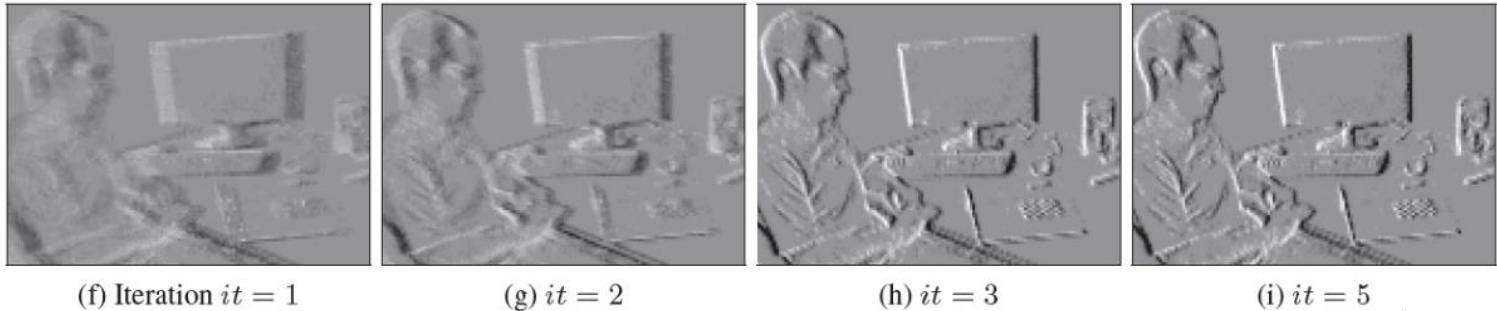
- **3D histograms of events.**  
voxel = discretized space-time
- **Voting (“insertion”) schemes:**
  - Nearest neighbor: each event votes for one cell only
  - Linear: each event splits its vote according to distance to neighboring voxels. Produces a smoother histogram.
- **Advantages:**
  - Preserves better the space-time structure of event data than 2D grid representations (such as event frames, time surfaces)
  - Compatible with conventional computer vision (CNNs, etc.)
- **Disadvantages:**
  - Memory (3D grid). Sparsity is lost: many grid values are zero.
  - Time is still quantized



Zhu et al., CVPR 2019

# Motion-Compensated Event Frames

- It is a function of **events** and a candidate **motion field**



Motion Compensation

- **Advantages:**

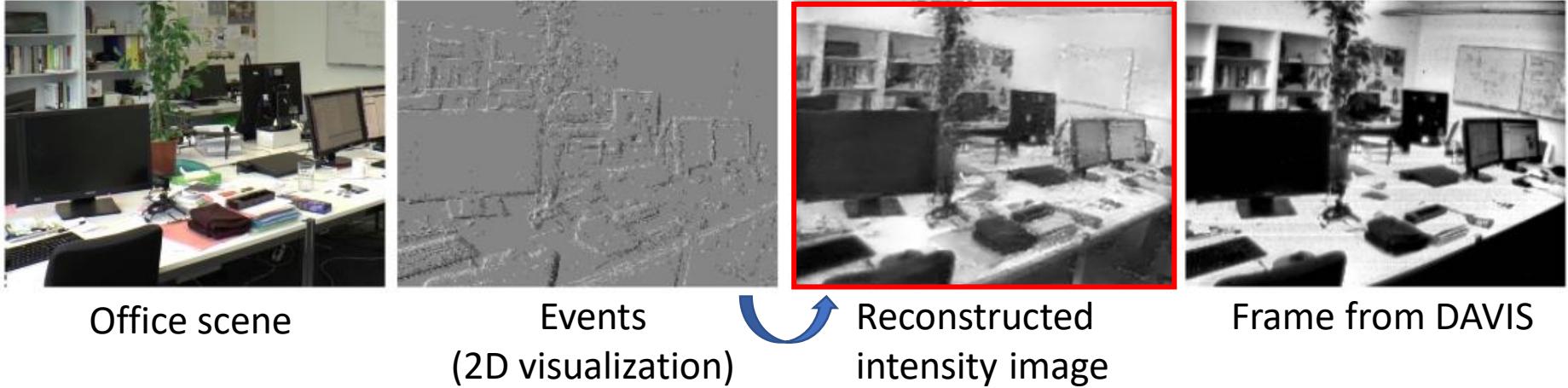
- Intuitive meaning: a sharp map of the edges causing the events
- Can be used to estimate the motion field that best fit the events
- Sharp images can be useful for later processing stages

- **Disadvantages:**

- If motion is not given, an estimation algorithm must provide it
- It is not fully motion invariant

# Reconstructed Intensity Images

- Brightness images **reconstructed from events** can be interpreted as a more **motion-invariant** representation of the visual information contained in the events



- **Pros:**
  - **Compatible** with conventional computer vision
  - HDR, High-speed video recovered
- **Cons:**
  - Expensive to compute, latency, and may contain artefacts

# Why so many different representations?

- Event data is **unconventional**. Cannot directly use the methods we have for standard cameras
- This is research. People **try new things** and see if they work (for their particular tasks and problem constraints)
- Representation may be **suggested by** constraints on method or platform utilization

# Visual code

- Events **encode** visual information: they are a “code”
- There are other codes (e.g., provided by other sensors)
- Most representations shown are **data pre-processing**. Representations for higher levels of abstraction in visual processing can be built from events, for example, using hierarchical NN. The output of such networks is another “visual code”.
- The boundary between “representation” and **feature extraction** is fuzzy.

# References

Reading:

- Section 3 of Gallego et al., [Event-based Vision: A Survey](#), TPAMI 2020
- Gehrig et al., [End-to-End Learning of Representations for Asynchronous Event-Based Data](#), ICCV 2019
- Rebecq et al., [High Speed and High Dynamic Range Video with an Event Camera](#), TPAMI 2020

# Event-based Robot Vision

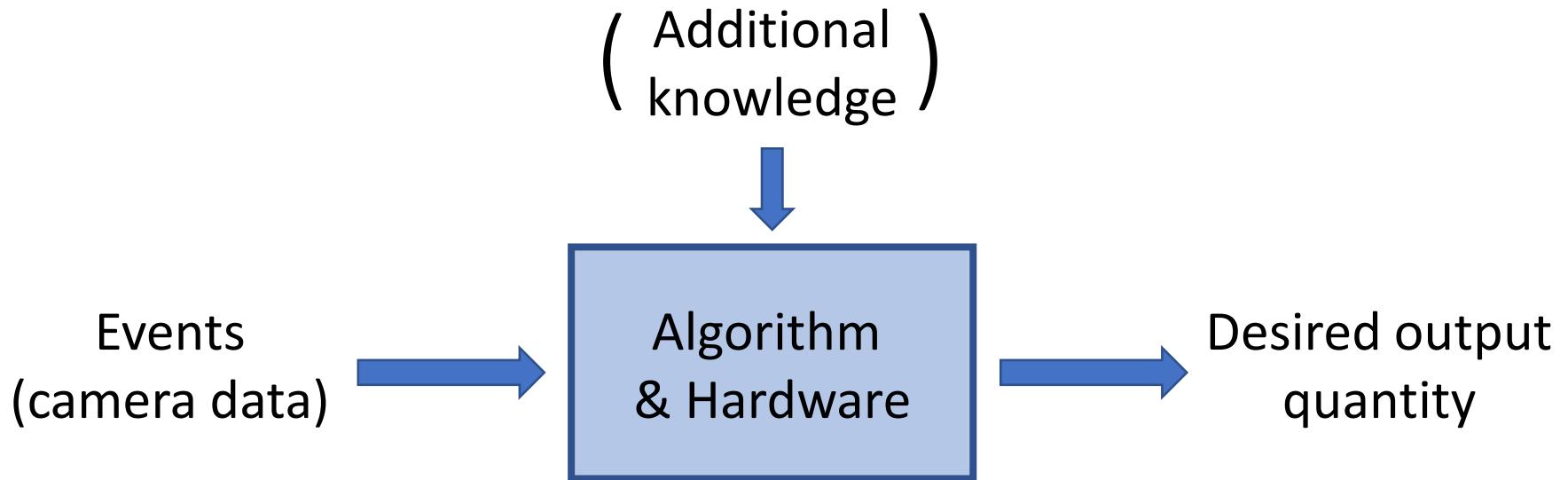
Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

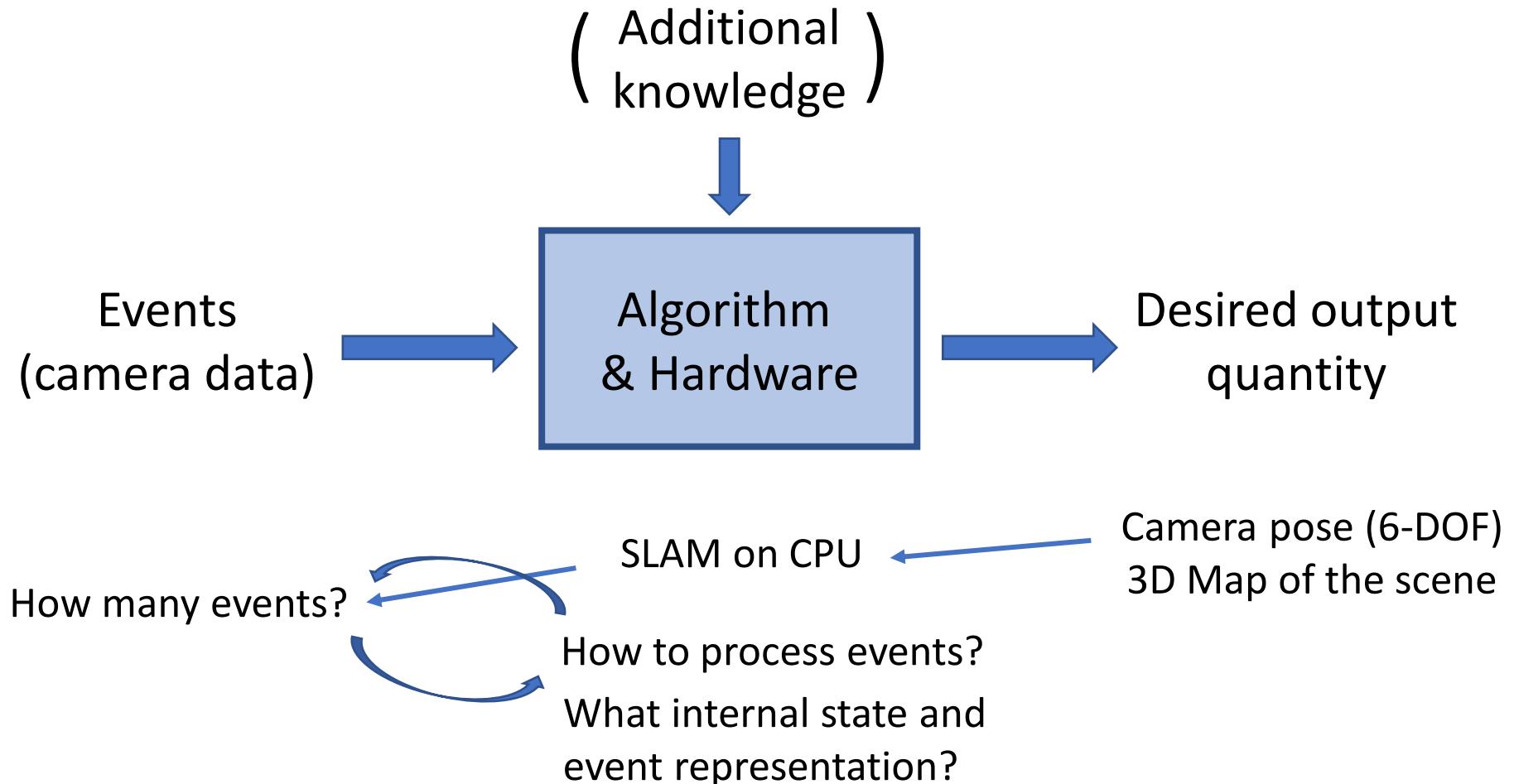
<http://www.guillermogallego.es>

# How to Process the Events?

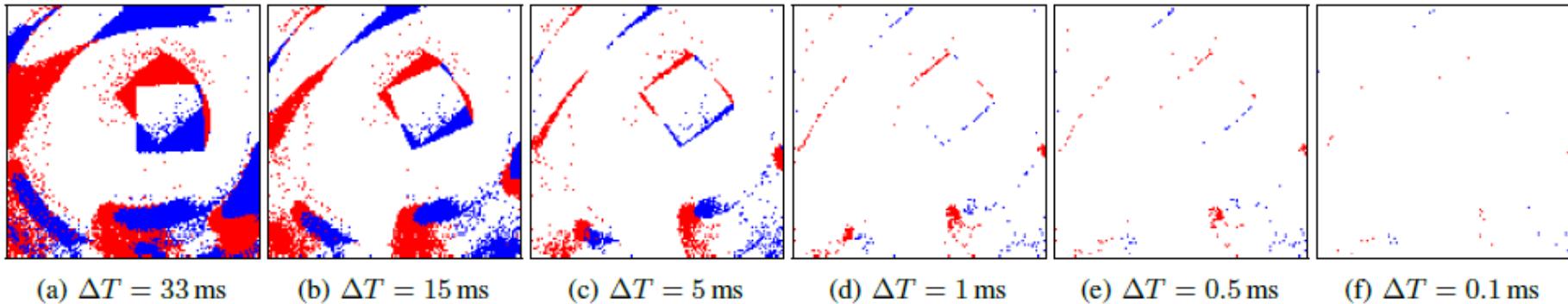
# Overview



# Overview



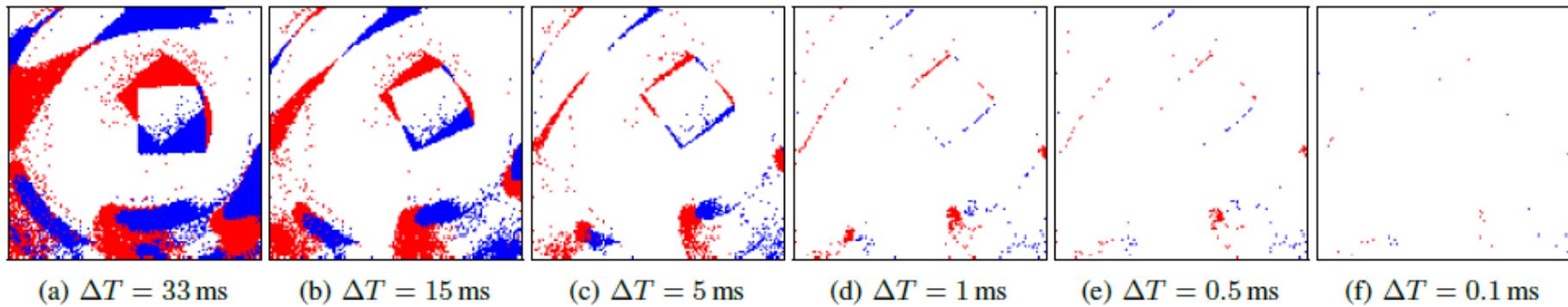
# How many events should be used?



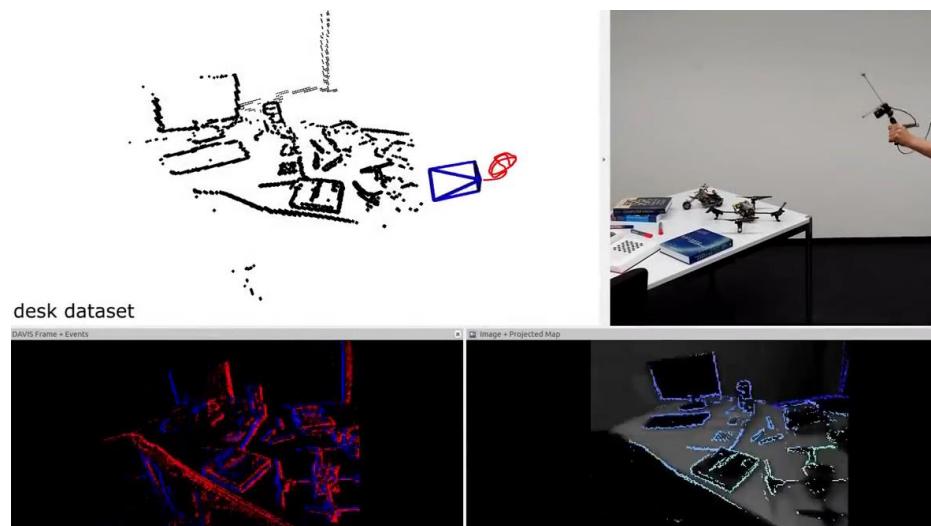
## How can we estimate unknown parameters?

- **Event-by-event processing** (i.e., process **one event** a time)
  - **Pros:** low latency (in principle down to microseconds)
  - **Cons:** with high-speed motion  $\rightarrow$  millions of events per seconds  $\rightarrow$  GPU
- **Event-packet processing** (i.e., process the last **N events**)
  - **Pros:** N can be tuned to allow real-time performance on a CPU
  - **Cons:** no longer microsecond resolution (when is it needed?)

# How many events should be used?



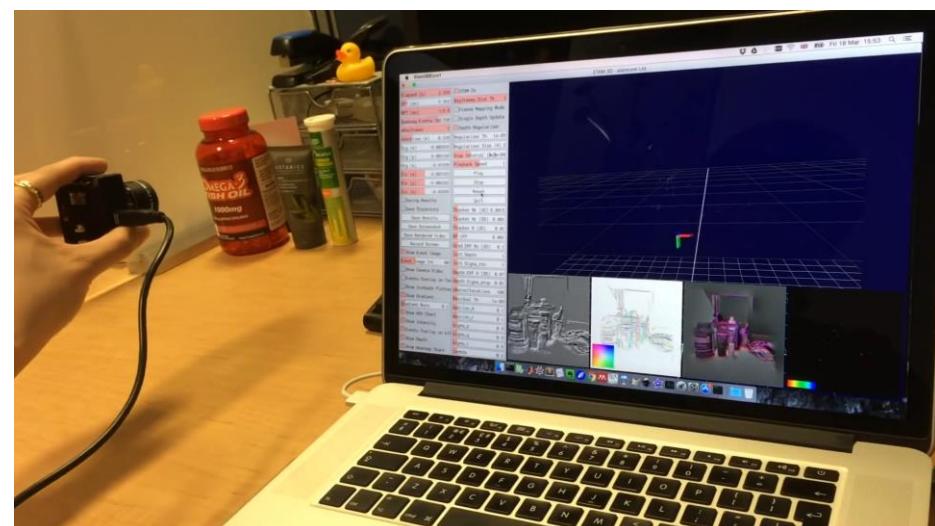
**Event-packet**



Rebecq et al., RA-L 2017 (CPU)

<https://youtu.be/bYqD2qZJlxE>

**Event-by-event**



Kim et al., ECCV 2016 (GPU)

<https://youtu.be/yHLyhdMSw7w>

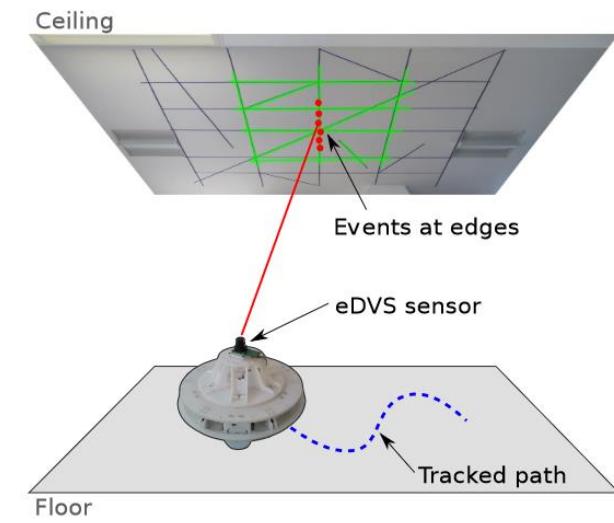
# Methods of Event Processing

# Taxonomy of Methods

- **Event-by-event:**
  - Filters (deterministic or probabilistic – Bayes filter)
  - ANNs - Spiking Neural Networks (SNNs)
- **Packets (groups) of events:**
  - Optimization or variational methods
  - Hand-crafted feature extractors
  - Modern DNNs on grid-based representations of events

# Event-by-event

- 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

# Event-by-event

- Examples:
  - Background activity **filter** for event noise removal
  - Per-pixel **temporal filter** for image reconstruction (ACCV'18)
  - Parallel tracking and mapping using the **Bayes filter** (EKF<sub>s</sub> and PF<sub>s</sub>): Weikersdorfer et al. ICVS 2013, Kim et al. BMVC 2014, Gallego et al. TPAMI 2018, etc.
  - **Multilayer ANNs** for object classification. Implemented in SNN<sub>s</sub>
- What's the **additional knowledge** on each system?  
(e.g., appearance information or scene “map” in SLAM)
- What's the inference **mechanism** used by each method to process the event and output a state update?  
(e.g., data fusion by EKF<sub>s</sub>, innovation)

# Packets of events

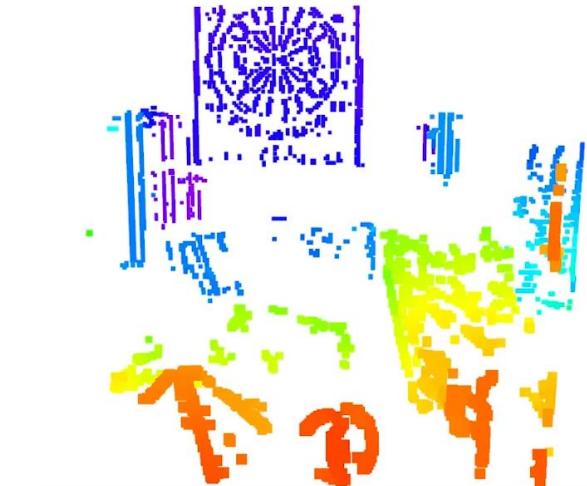
- Process **events in a packet / group / window**
- Aggregate information from multiple events.  
May not need additional knowledge.
- Events are processed differently depending on their representation. Specific methods for time surfaces, event frames, voxel grids, etc.

## • Advantages:

- Efficient event aggregation
- May be asynchronous, with adaptive rate
- Broader toolbox than event-by-event

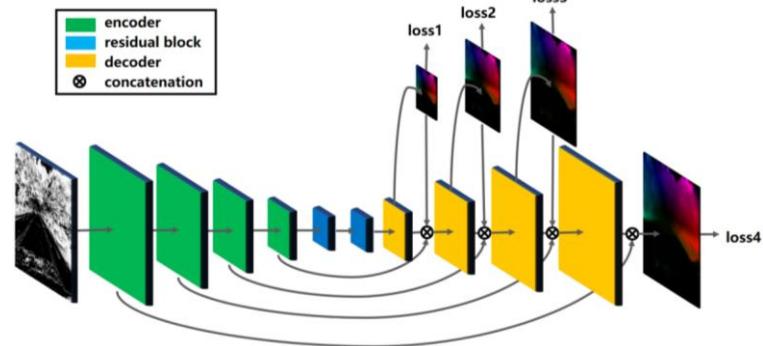
## • Disadvantages:

- Introduce some latency



Rebecq et al., EMVS IJCV 2018

# Packets of events



- Examples:
  - Conventional computer vision on **event frames or voxel grids**
    - Classical feature detection, extraction and tracking (e.g., Harris, KLT, etc)
    - Modern ANNs (CNNs, DNNs, encoder-decoder architectures, etc.)
  - Hierarchical feature extractors + shallow classifiers for **time surfaces**
  - Contrast / Focus maximization methods on **motion-compensated images**
- How is information in the event stream aggregated and filtered to produce a state update (the desired output)?
- How is additional knowledge, if any, considered?

# References

Reading:

- Section 3 of Gallego et al., [Event-based Vision: A Survey](#), TPAMI 2020

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

# Event Generation Model

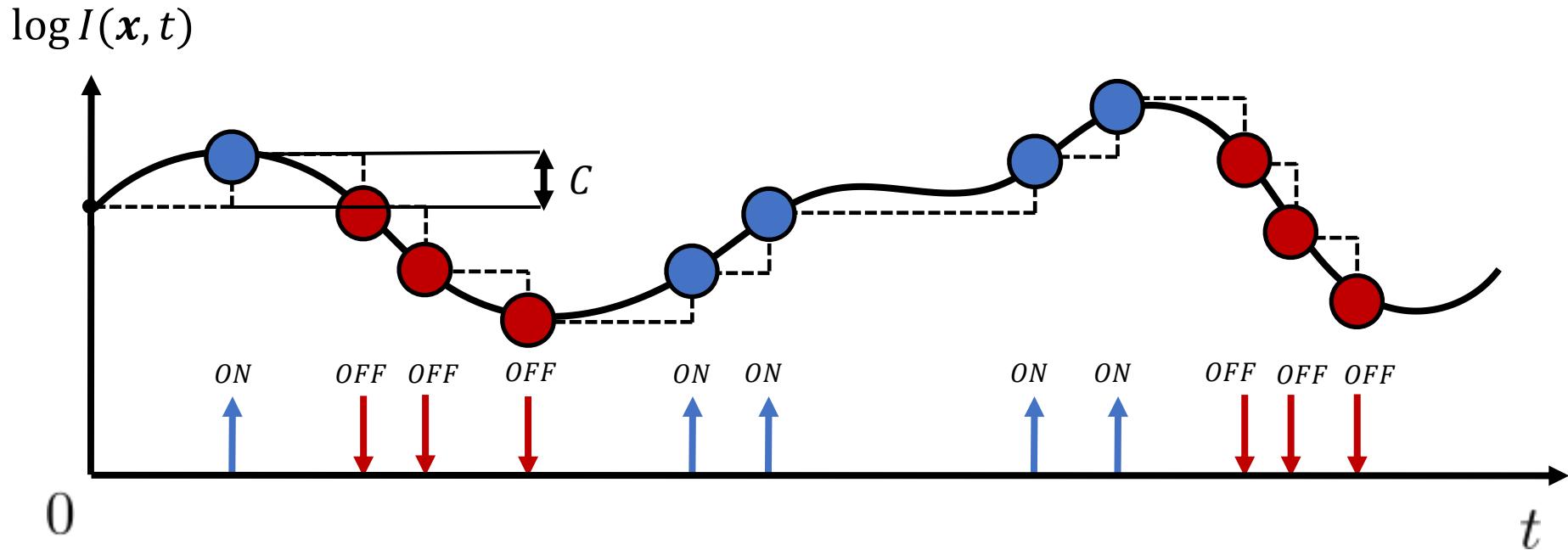
# Event Generation model(s)

- **Original** (pixelwise, non-linear, no noise)
- **Linearized** (using brightness constancy)
- More **realistic** (incorporating noise)
  - Bell-shaped, Gaussian. In which variable?
  - Mixture model (“good” and “bad” measurements)
  - Dependent on more variables (such as refractory period)

# Event Generation Model at a Pixel

An event is triggered at pixel  $x$  if:

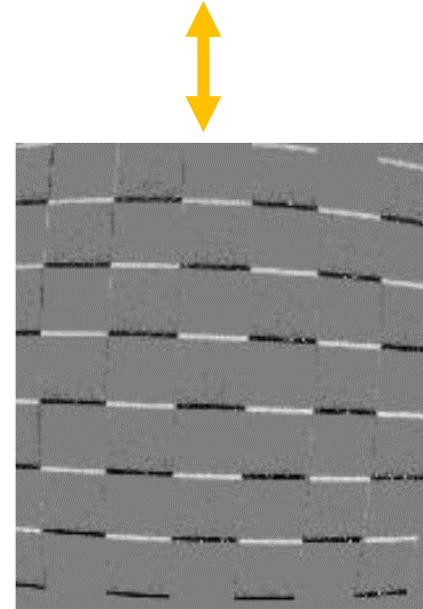
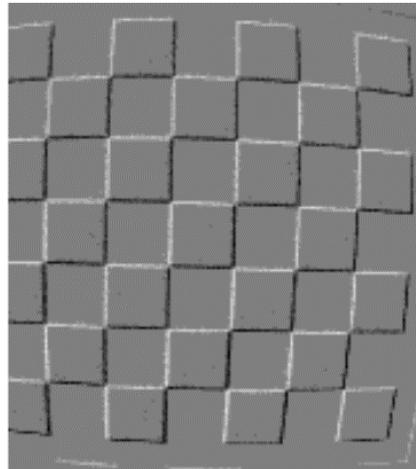
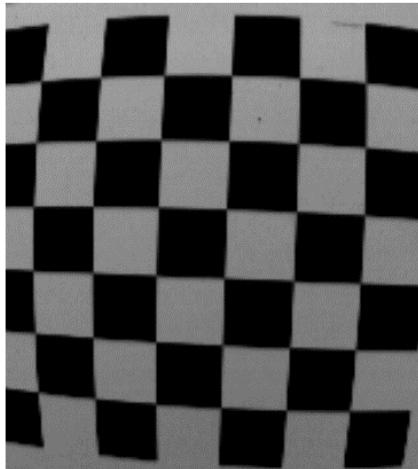
$$\log I(x, t) - \log I(x, t - \Delta t) = \pm C$$



Light ( $\log I$ ) is transduced into a stream of asynchronous events

# What causes the events?

- DVS pixels measure brightness changes (i.e., temporal contrast)  
$$\log I(x, t) - \log I(x, t - \Delta t) = \pm C$$
- These changes could be due to a **varying light** (e.g., LED)
- However, assuming **constant illumination**... what causes the events?  
⇒ **moving edges** in the scene
- Can we say more? Observe the **spatial** patterns of events:



# What causes the events?

Going from temporal contrast to spatial information:

- An event is generated if  $\Delta L := L(t_k) - L(t_k - \Delta t) = \pm C$
- For a short time:  $\Delta L \approx \frac{\partial L}{\partial t} \Delta t$
- Assuming **brightness constancy**

$$0 = \frac{dL(x(t), t)}{dt} = \nabla L \cdot \frac{dx}{dt} + \frac{\partial L}{\partial t} \quad (\text{this is just the chain rule})$$

where  $\dot{x} = dx/dt$  (a velocity  $v$ ), we may work out  $\frac{\partial L}{\partial t} = -\nabla L \cdot \dot{x}$

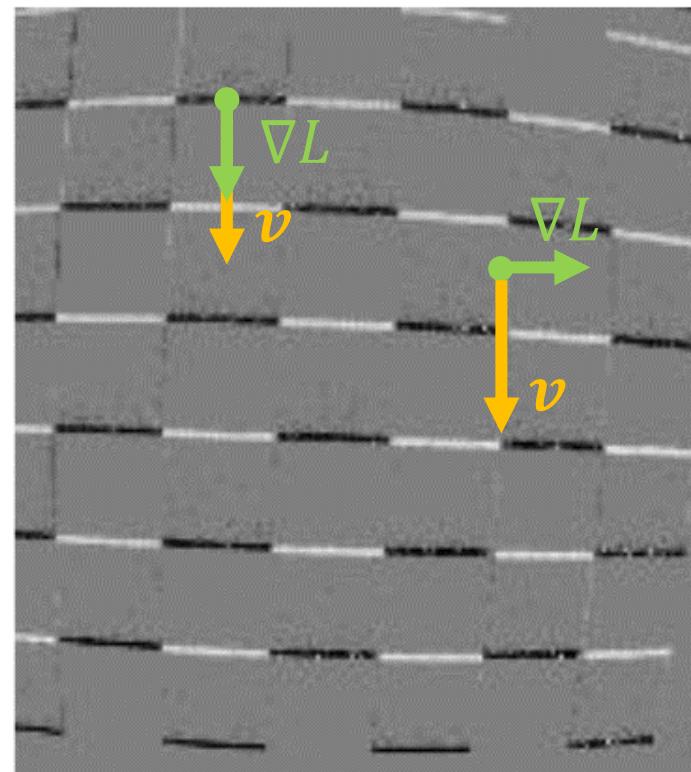
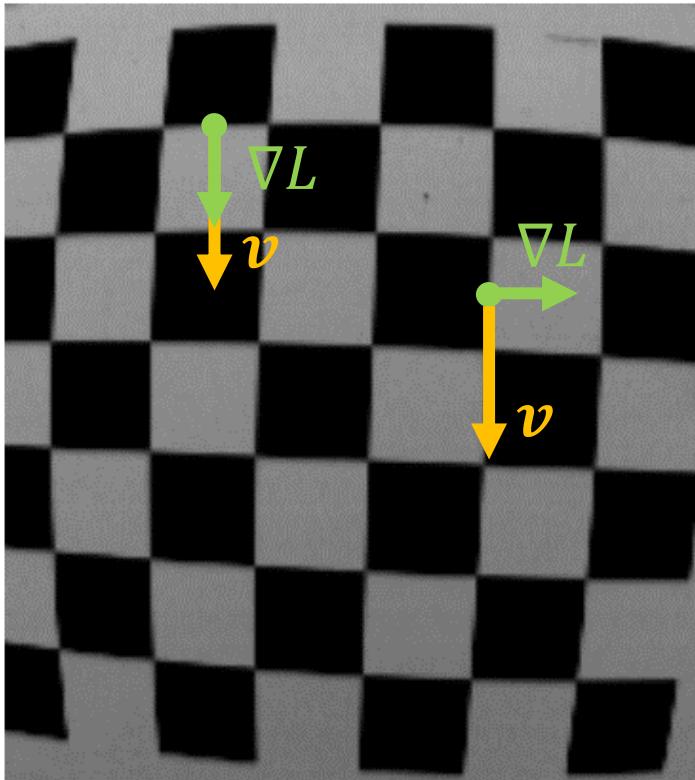
- Substituting:

$$\Delta L \approx -\nabla L \cdot v \Delta t$$

- Brightness change is **caused** by brightness gradient **moving** with a velocity  $v$  on the image plane, over a displacement  $\Delta x := v \Delta t$
- If brightness gradient  $\perp$  motion  $\Rightarrow$  no event is generated
- If brightness gradient  $\parallel$  motion  $\Rightarrow$  event is generated fastest

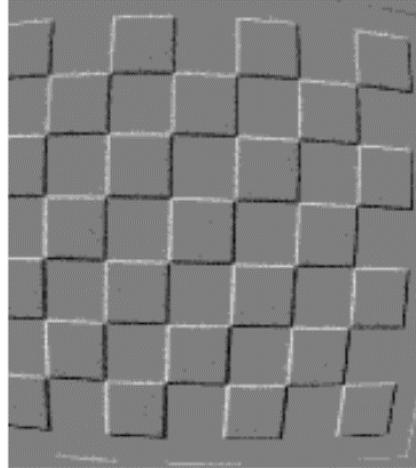
# Review the earlier example

- If brightness gradient  $\perp$  motion  $\Rightarrow$  no event is generated
- If brightness gradient  $\parallel$  motion  $\Rightarrow$  event is generated fastest



# Event Generation

- In conclusion, for constant illumination, events are caused by **moving edges**.
- When we accumulate the events pixelwise what we see is the **trace of the edge** as it **moved and generated events**.  
(This is not “motion blur”)



- Some methods exploit the linearized model; others do not

# More Realistic Event Generation Model

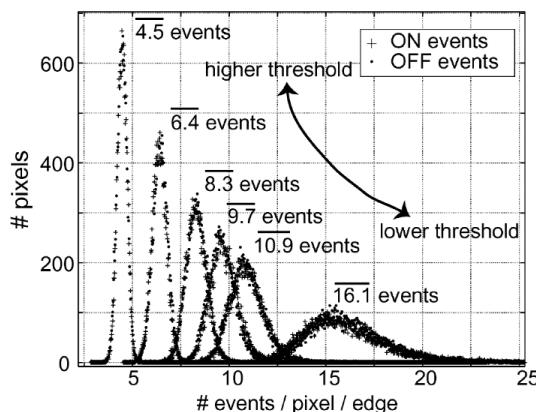
- A probabilistic model:

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

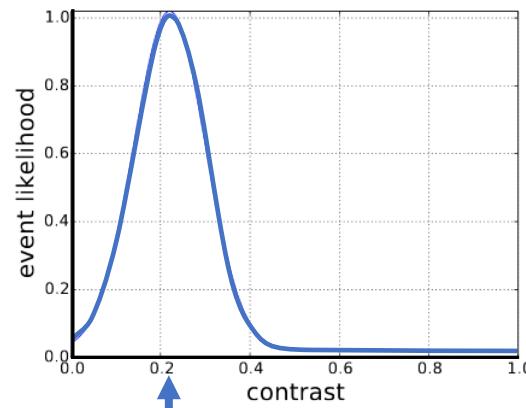


Replace this constant with  
a **probability distribution**

Measured JSSC 2008



Modeled BMVC 2014



$C$  “average” or “mode”

Fig. 6. Distributions in the number of events recorded per pass of the bar for 40 repetitions of the 15:1 contrast bar sweeping over the array, e.g., for the highest threshold setting there are an average of 4.5 ON and 4.5 OFF events per ON and OFF edge.

# References

- Section 2.4 of [Event-based Vision: A Survey](#), TPAMI 2020
- Papers on event noise measuring or modeling:
  - JSSC 2008 – Journal paper about DVS
  - BMVC 2014, arXiv 2015, ICCP 2017, TPAMI 2018
  - Further reading: “V2E: From video frames to realistic DVS event camera streams”, [arXiv](#) 2020.
- Classical computer vision books for the topic of [brightness constancy](#).

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

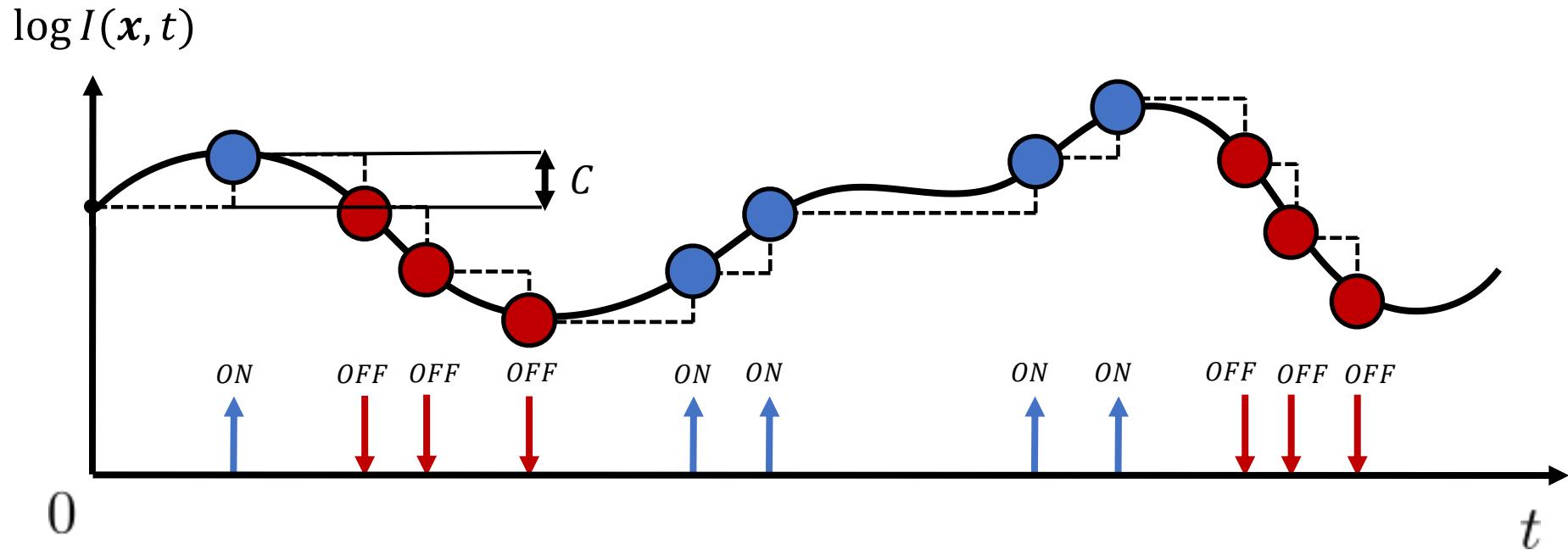
[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

# Image (intensity) Reconstruction

# Recall the Event Generation Model

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

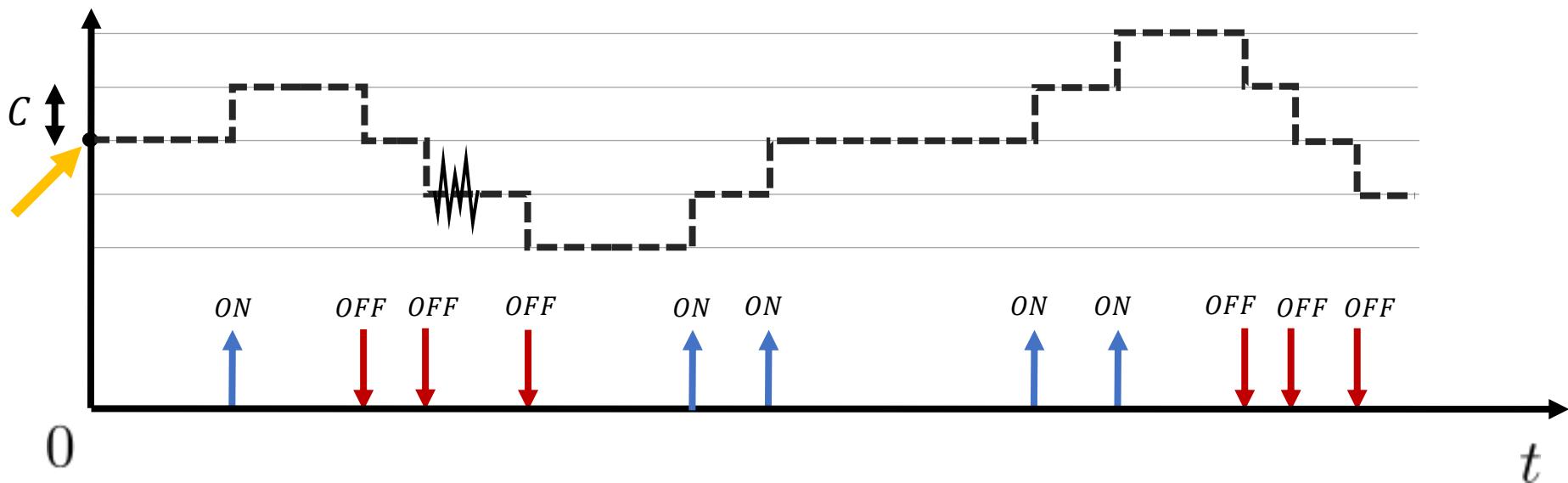


Light ( $\log I$ ) has been transduced into asynchronous events...

**Given the events, can we recover the absolute intensity?**

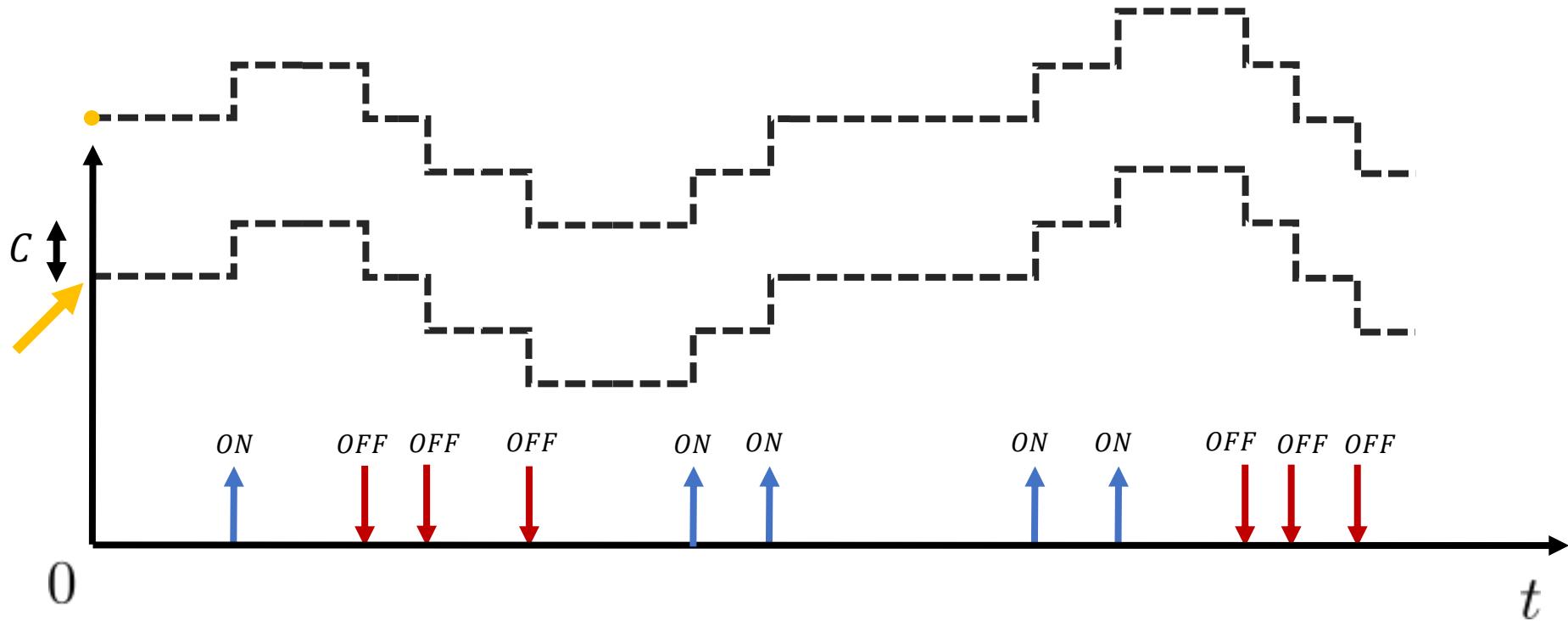
# Let's try to recover the pixel's intensity

- Events represent **intensity changes**  $\Rightarrow$  **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's try to recover the pixel's intensity



- Typically the **offset** (at  $t = 0$ ) is unknown...
- That's what happens at 1 pixel... and we need offsets on all image pixels to make a good (coherent) image

# Event Integration

- Estimated intensity. Intuition:

$$L(t) = L(0) + \underbrace{\int_0^t \frac{dL}{dt}(\tau) d\tau}_{\text{Increment } \Delta L := L(t) - L(0)}$$
$$\log \hat{I}(x, t) = \log I(x, 0) + \sum_{0 < t_k \leq t} p_k C \delta(x - x_k) \delta(t - t_k)$$

↑  
pixel

↑  
Intensity at  $t = 0$   
(offset)

↑  
0 <  $t_k \leq t$   
polarity

↑  
Kronecker delta (discrete variable)

↑  
Dirac delta (continuous variable)

↑  
Intensity increment  $\Delta \log I$

# Image Reconstruction sneak peek

- Isn't it magic? We can reconstruct some intensity **starting from zero offset** (initial condition), i.e., without knowing  $\log I(x, 0)$



- Did we notice how intensity (i.e., grayscale) information **appears as edges move** through the scene and then it fades away?
- Did we notice the background trees as “after image” while fading?

# References

Reading:

- E. Mueggler et al., *The Event-Camera Dataset and Simulator*, IJRR 2017, page 3.

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

# Image (intensity) Reconstruction

## History & Evolution

# Brief history of Brightness Reconstruction



Belbachir et  
al., CVPRW  
Tuco3D



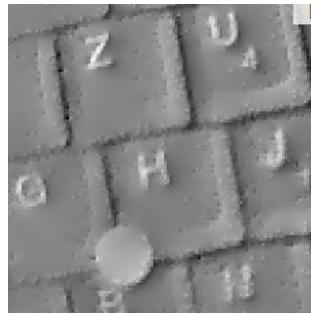
Barua et al.,  
WACV



Reinbacher  
et al. BMVC

2011

Cook et al. IJCNN



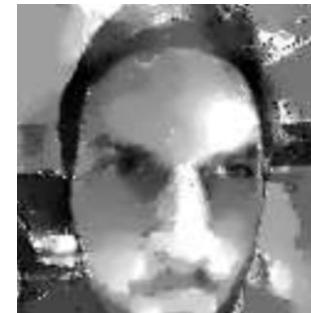
2014

Kim et al. BMVC



2016

Bardow et al.  
CVPR



Kim et al.  
ECCV



# Brief history of Brightness Reconstruction



Scheerlinck et al,  
AACV



Mostafavi et al.  
CVPR



Scheerlinck et al,  
WACV

2017

Rebecq et al. RAL

2018

2020

2019

Rebecq et al. CVPR

Rebecq et al. PAMI



# Brief history of Color Reconstruction

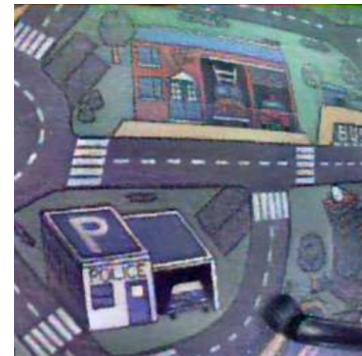


Scheerlinck et al,  
CVPRW

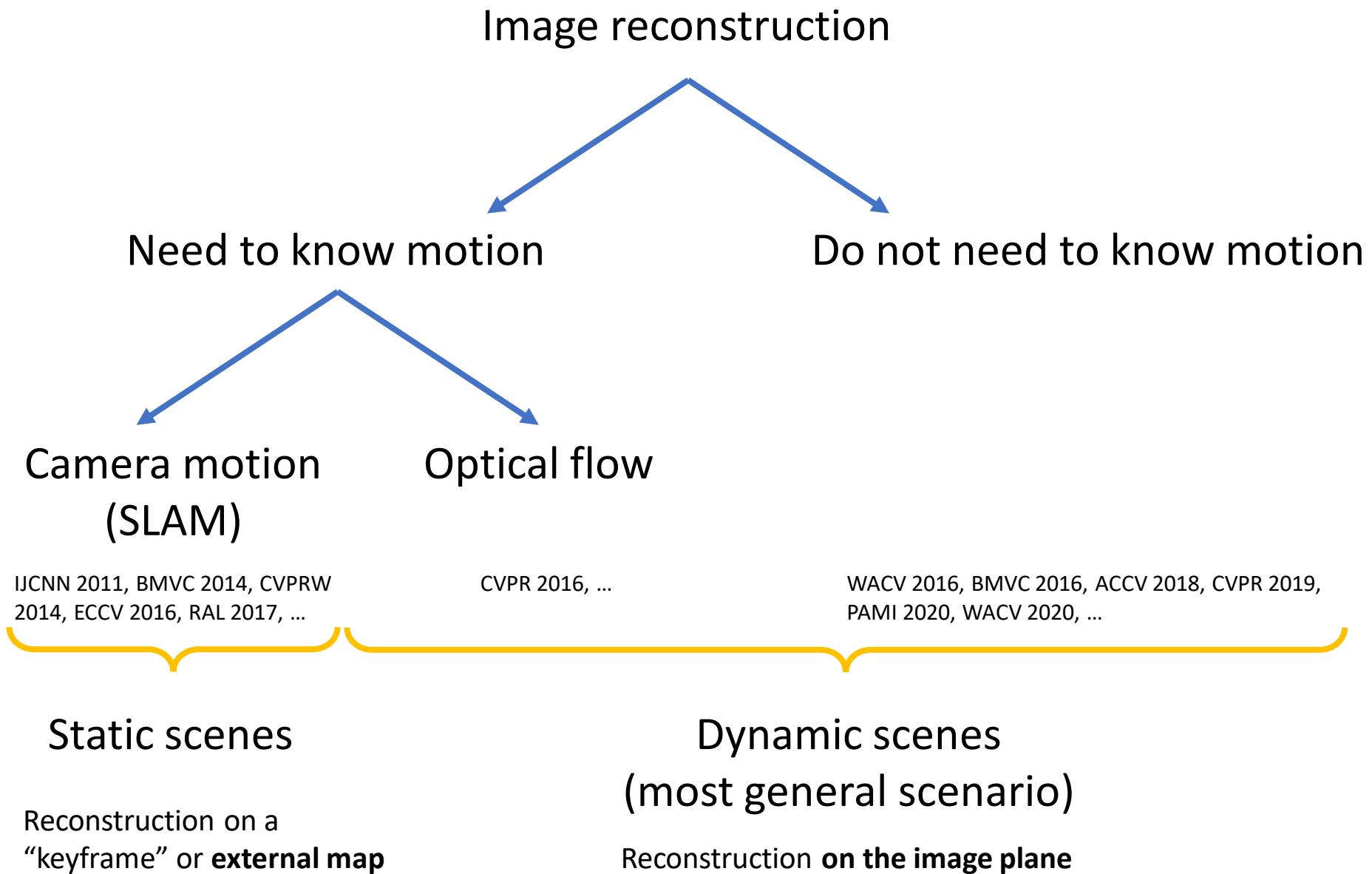


Moeys et al. ISCAS

Rebecq et al. PAMI



# Assumptions & Scenarios



# Classification of Methods

- Need to know **ego-motion** (e.g., SLAM)
  - Work on static scenes: Cook et al IJCNN 2011, Kim et al. BMVC 2014 & ECCV 2016, Rebecq et al. RAL 2017, ...
- Need to know **apparent motion** (e.g., optical flow)
  - Relaxed conditions compared to SLAM: Bardow CVPR 2016
  - Can work on dynamic scenes
- **Do not need to know motion.**  
Work for arbitrary scenes (most general scenario)
  - **Model-based:** Reinbacher (BVMC 2016), Scheerlinck (ACCV 2018)
  - **Learning-based:** Barua (dictionary learning), Rebecq (U-Net), Mostafavi (GANs), Bardow's thesis (GANs), etc.

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

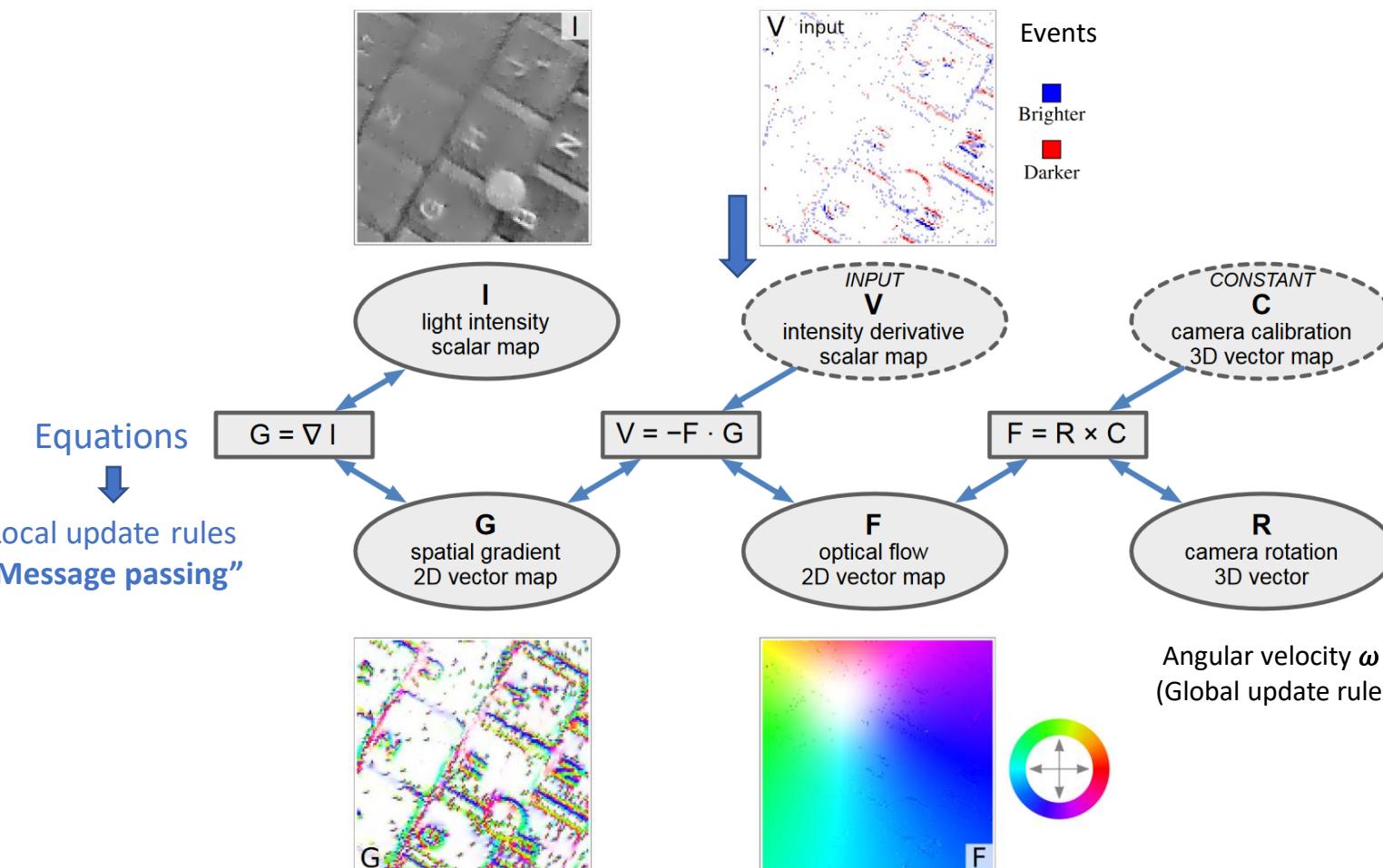
<http://www.guillermogallego.es>

# Image (intensity) Reconstruction

## Literature Review

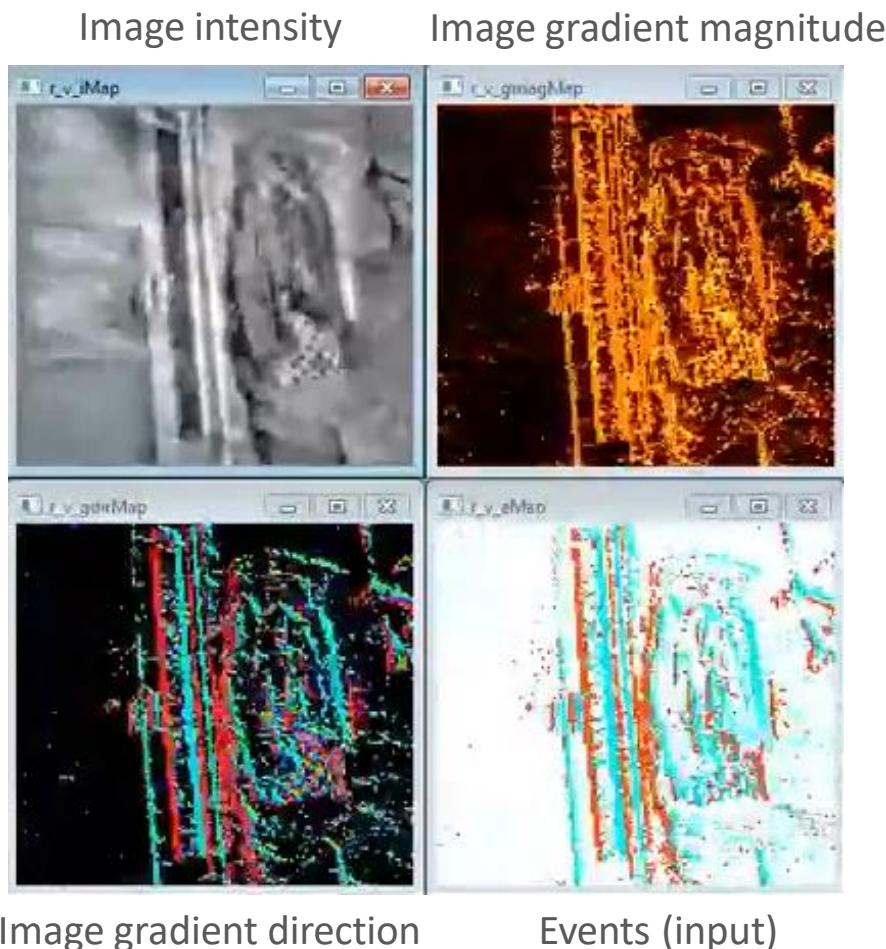
# “Interacting Maps”

- Simultaneous estimation of multiple visual quantities with a *rotating* event camera



# “Interacting Maps”

- Simultaneous estimation of multiple visual quantities with a *rotating* event camera



Inspired by the primary visual cortex.

From the events, jointly estimate:

- Rotation (3 DOF ego-motion)
- Optical flow
- Image gradient
- Intensity reconstruction ←

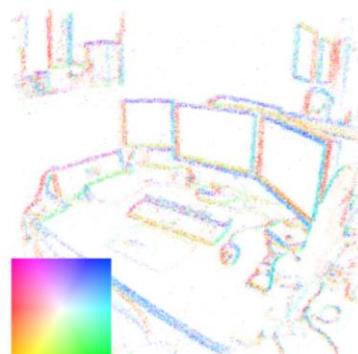
Pure rotation: no translation or depth

# Simultaneous mosaicing and tracking

- Parallel tracking and mapping
- *Rotating event camera (no translation or depth)*



- **Mapping** = mosaicing (panoramic imaging)
  - Get intensity gradient  $g$  using pixel-wise EKF:  
$$h(e|R) = \frac{g \cdot v}{c}$$
 should equal the pixel event rate  $\frac{1}{\Delta\tau}$   
(uses linearized event generation model)
  - Poisson reconstruction to obtain intensity



Intensity gradient

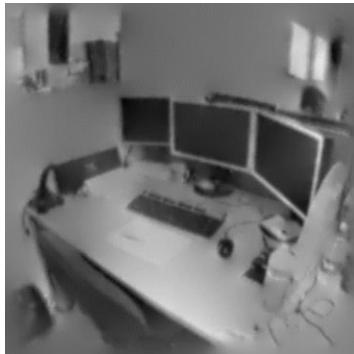
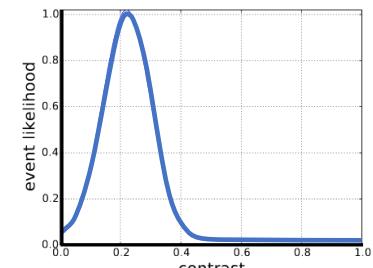
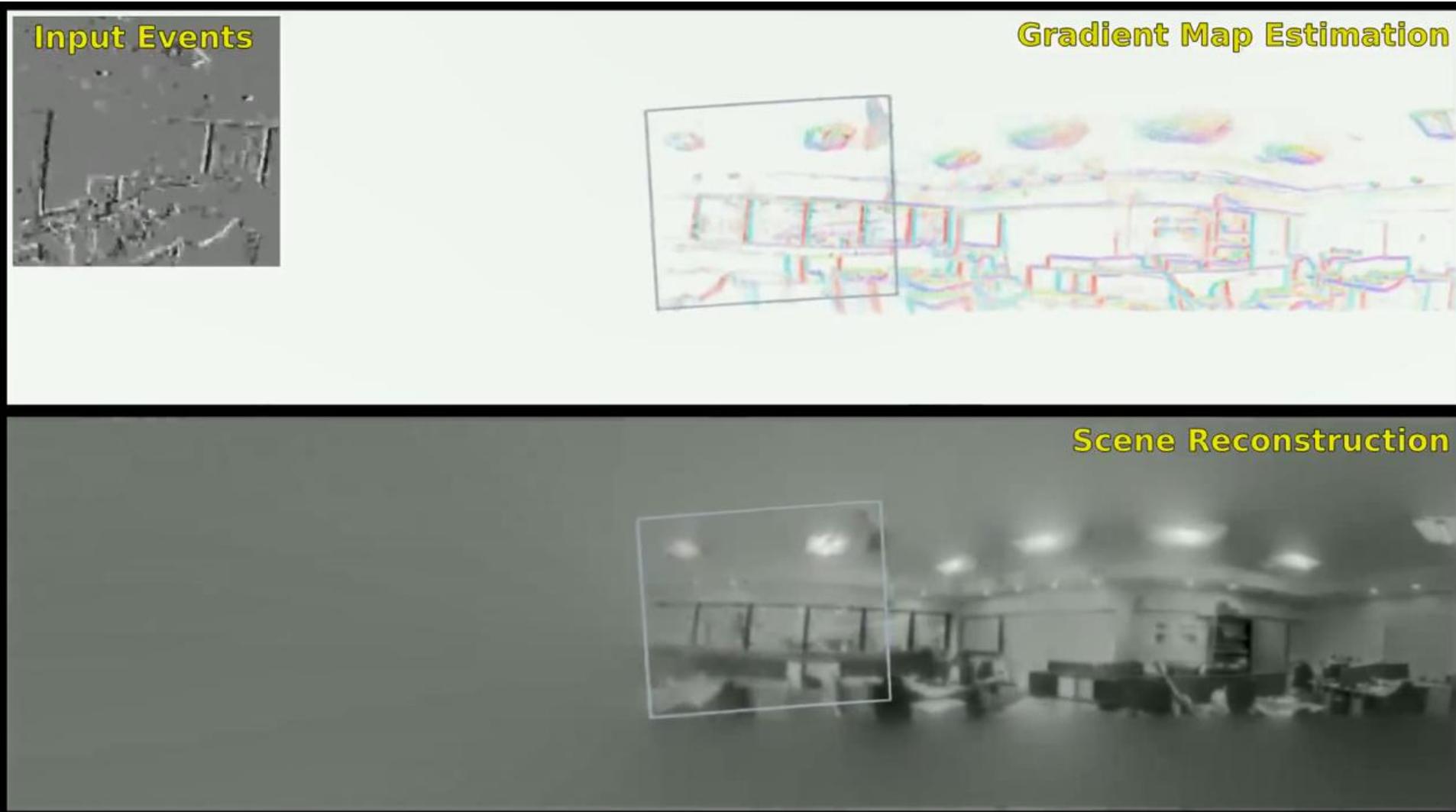


Image intensity

- **Tracking** (ego-motion estimation)
  - Random diffusion in motion space
  - Particle filter: particle weights updated using the map: contrast =  $|\log I(t) - \log I(t - \Delta t)|$



# Simultaneous mosaicing and tracking

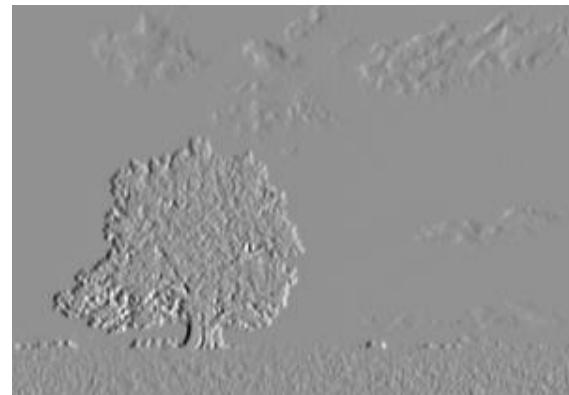


# Image reconstruction by Poisson integration

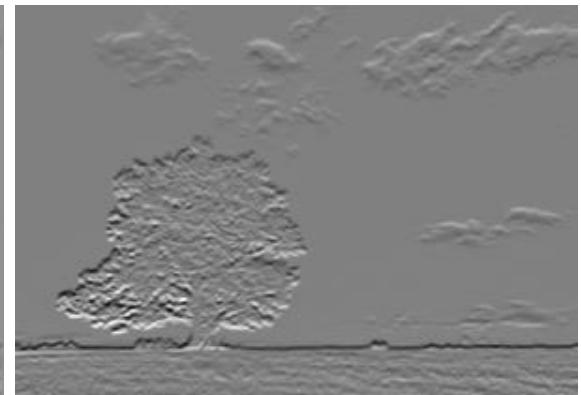
Integrate gradient map  $\mathbf{g}$  to get absolute brightness  $M$



Original Image



Gradient in  $x$  direction  
 $(g_x = \partial_x I)$

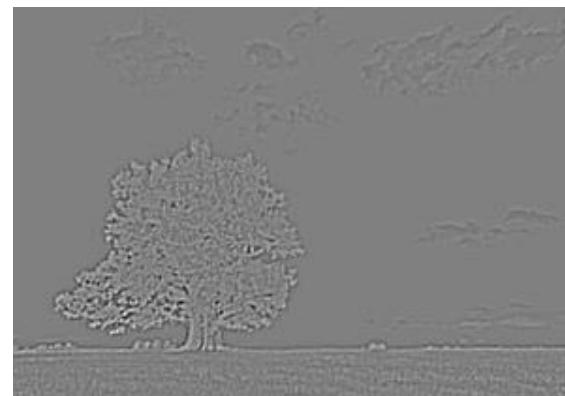


Gradient in  $y$  direction  
 $(g_y = \partial_y I)$



Reconstructed Image

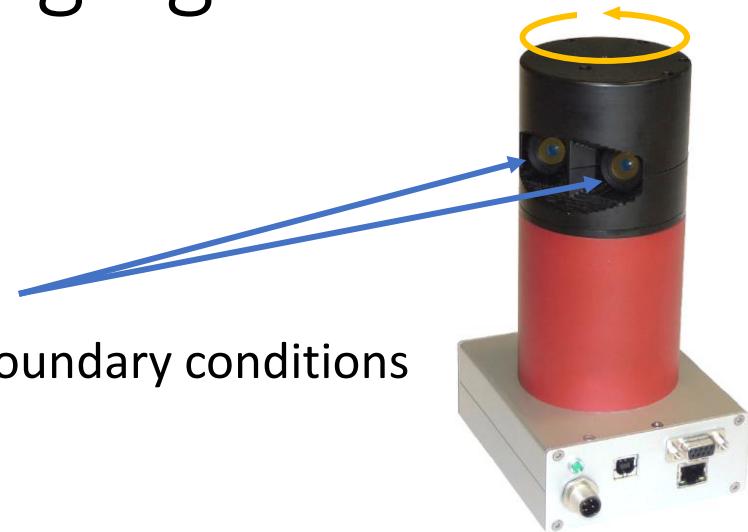
2D integration  
←  
Solve Poisson eq:  
 $(\Delta \tilde{I} = \operatorname{div} \mathbf{g})$   
fast using the FFT



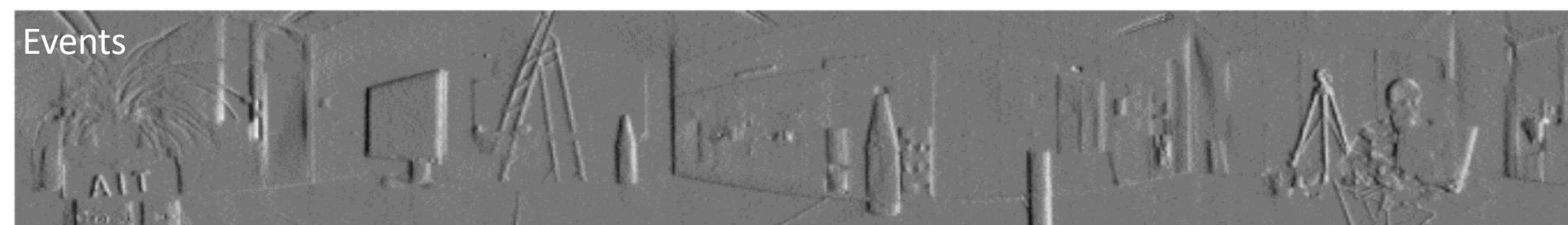
Divergence  
 $(\operatorname{div} \mathbf{g} = \partial_x g_x + \partial_y g_y)$

# TUCO-3D Panoramic Imaging

- 360 deg Panoramic Imaging
- Exploit constrained motion
  - Two rotating 1D event cameras (stereo)
  - Event integration must satisfy periodic boundary conditions
- It also provides depth (3D)



TUCO-3D by AIT

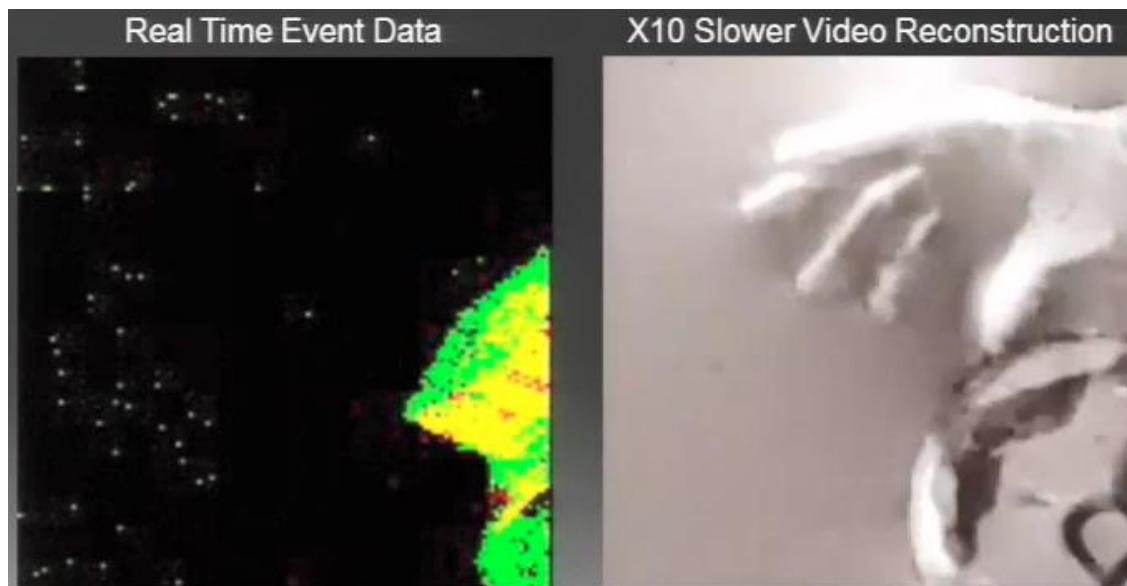
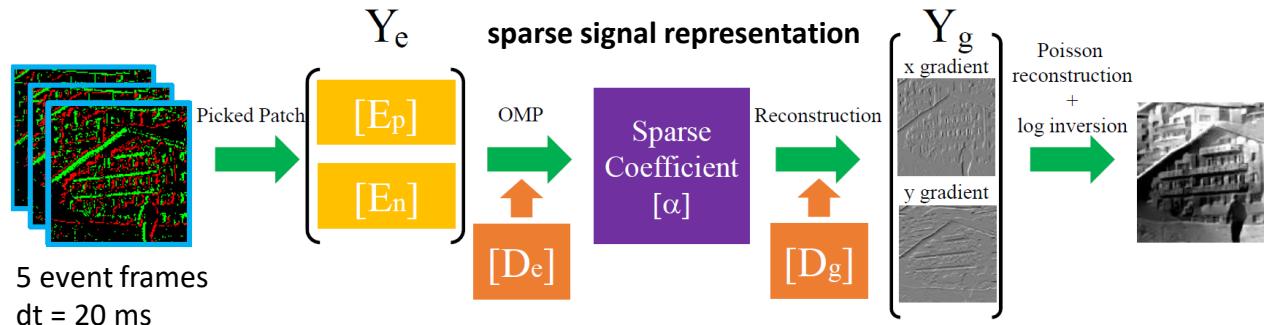


↓ Integrate brightness changes, line by line



# Image Reconstruction using a Sparse Dictionary

- Shows that there is **no need to estimate motion** for reconstruction
- **Patch-based dictionary of events** learnt from simulated data



Green: positive

Red: negative

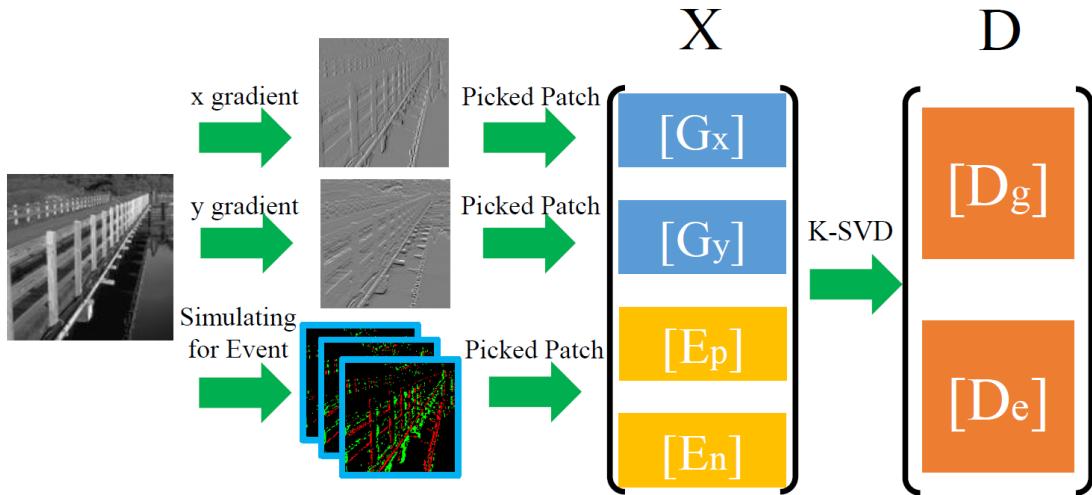
Yellow: both events

Video at **2 kHz**

Patches of  $9 \times 9$  pixels

# Image Reconstruction using a Sparse Dictionary

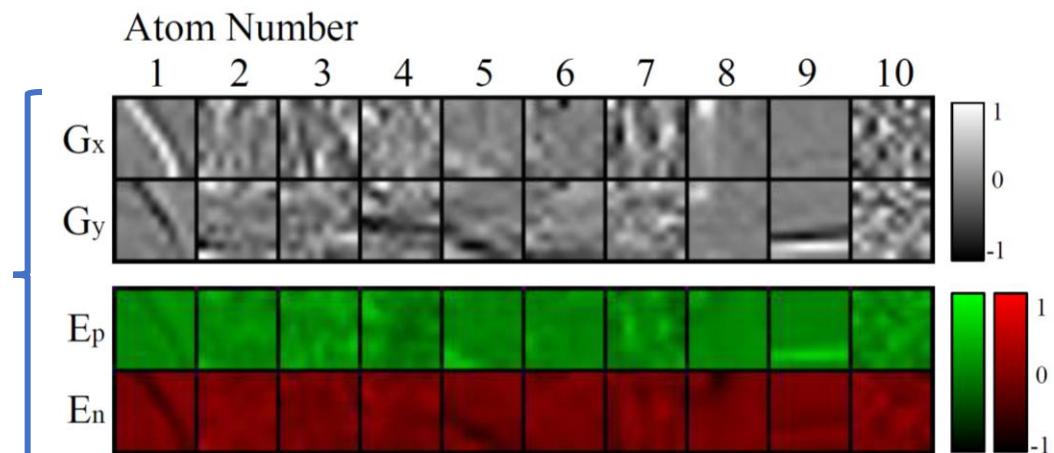
- How is the dictionary computed (i.e., learned)?



**K-SVD** is a generalization of the **k-means clustering** method (i.e., **unsupervised**)

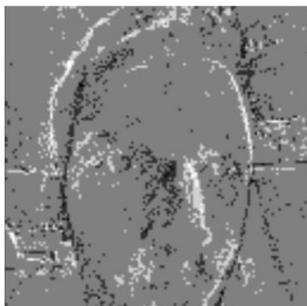
These are the 4 components in the dictionary.

Given positive ( $E_p$ ) and negative ( $E_n$ ) events, we have corresponding  $G_x$ ,  $G_y$

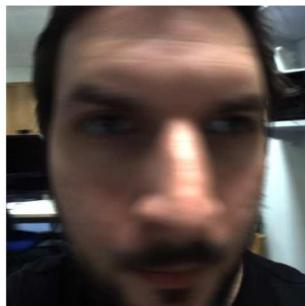


# SOFIE: Simultaneous Optical Flow & IE

- **Joint optimization** over image reconstruction and **optical flow** to explain a volume of events (voxel grid)
- More relaxed than SLAM methods; just need optical flow, i.e., works on **dynamic scenes** with **little assumptions about motion**.
- Solve a variational optimization problem:



(a) Raw event camera output



(b) Standard camera image



(c) Intensity estimate from events



(d) Optical flow from events

$$\begin{aligned} \min_{\mathbf{u}, L} \int_{\Omega} \int_T & \left( \lambda_1 \|\mathbf{u}_{\mathbf{x}}\|_1 + \lambda_2 \|\mathbf{u}_t\|_1 + \lambda_3 \|L_{\mathbf{x}}\|_1 + \right. \\ & \left. \lambda_4 \|\langle L_{\mathbf{x}}, \delta_t \mathbf{u} \rangle + L_t\|_1 + \lambda_5 h_{\theta}(L - L(t_p)) \right) dt d\mathbf{x} \\ & + \int_{\Omega} \sum_{i=2}^{|P(\mathbf{x})|} \|L(t_i) - L(t_{i-1}) - \theta \rho_i\|_1 d\mathbf{x}, \end{aligned}$$

Smoothness terms

Optical flow term (brightness constancy)

No-event term

Event term

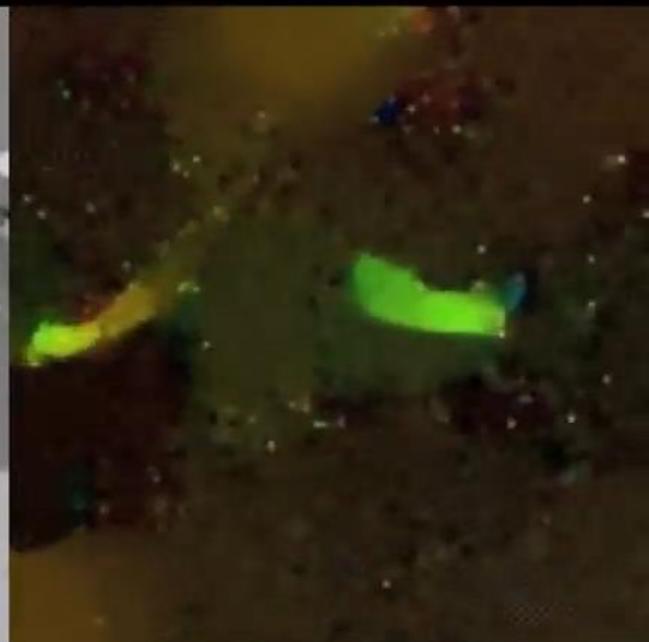
# SOFIE: Simultaneous Optical Flow & IE



Camera Image



Reconstruction

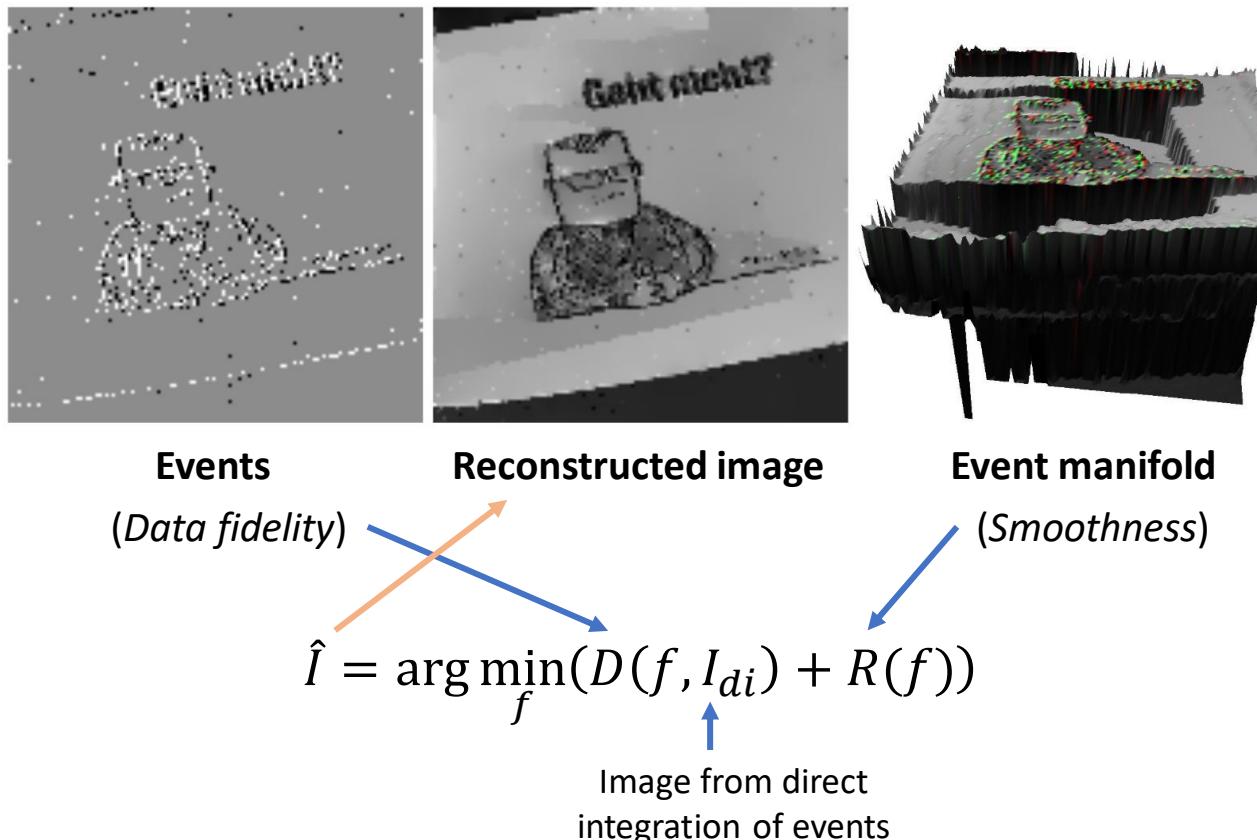


Optical Flow

1 frame = 4ms

# Reconstruction using “Manifold Regularisation”

- Does not need to estimate motion
- Reconstruction is posed as **variational nonlinear image denoising**, using the time surface (event timestamps) to guide the denoising



# Reconstruction using “Manifold Regularisation”

- Does not need to estimate motion
- Reconstruction is posed as **variational nonlinear image denoising**, using the time surface (event timestamps) to guide the denoising

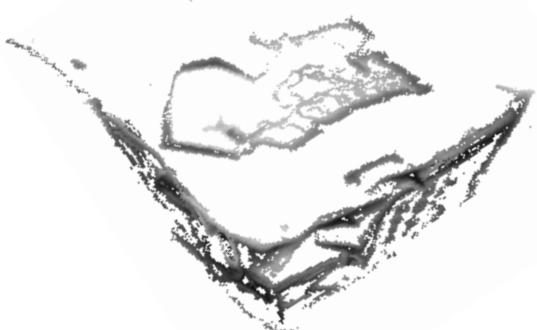


# Event-based 6-DOF SLAM with 3 parallel filters

## Parallel 6 DOF Tracking & Mapping in real time on a GPU



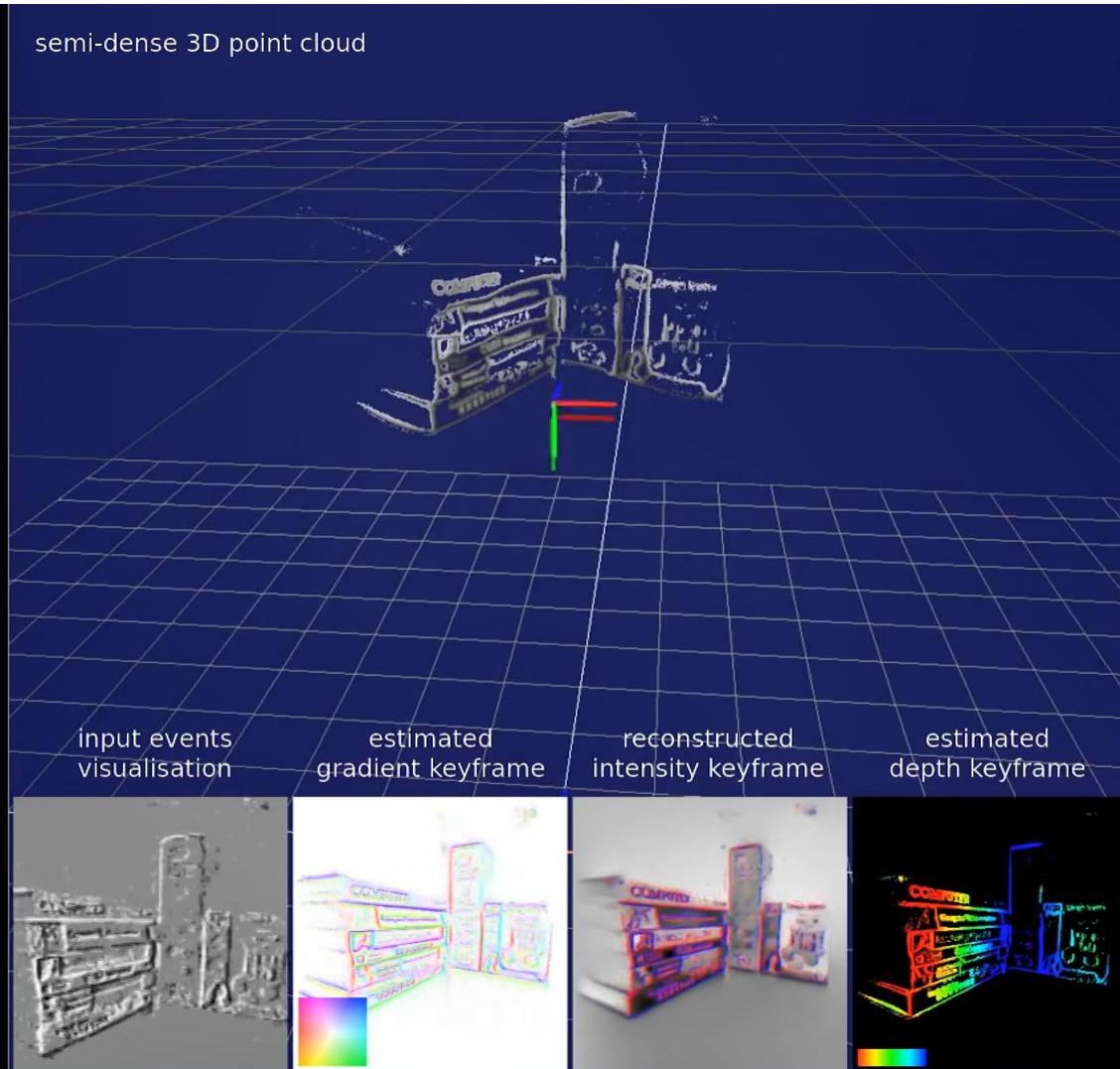
3D map of the scene



- **3 Kalman Filters** running in parallel.  
Each filter needs the output from the others.
- **Tracking:** EKF in 6 DOF pose
  - Motion model: constant position
  - Use contrast  $h_{\mathbf{x}}(\mathbf{x}^{(t|t-\tau)}) = \mathbb{I}_l \left( \mathbf{p}_w^{(t)} \right) - \mathbb{I}_l \left( \mathbf{p}_w^{(t-\tau_c)} \right)$  to update pose (needs depth and intensity)
- **Intensity reconstruction:** pixel-wise EKF like Kim'14 & robust Poisson (Huber norm)
- **Mapping:** pixel-wise EKF on inverse depth, using contrast  $h_{\rho} = \mathbb{I}_l \left( \mathbf{p}_w^{(t)} \right) - \mathbb{I}_l \left( \mathbf{p}_w^{(t-\tau_c)} \right)$

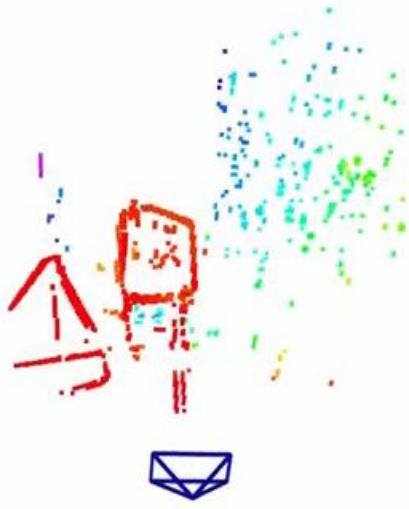
# Event-based 6-DOF SLAM with 3 parallel filters

Parallel 6 DOF Tracking & Mapping in real time on a GPU



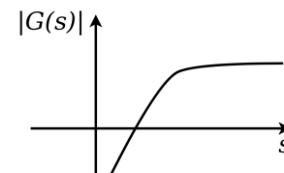
# Event-based 6-DOF SLAM on a CPU

HDR image reconstruction from the output of SLAM



# Reconstruction by Temporal Filtering

- Replace **pixel-wise** direct integration with a **high-pass temporal filter** to remove accumulated event noise
- No spatial filtering needed



Direct integration of events



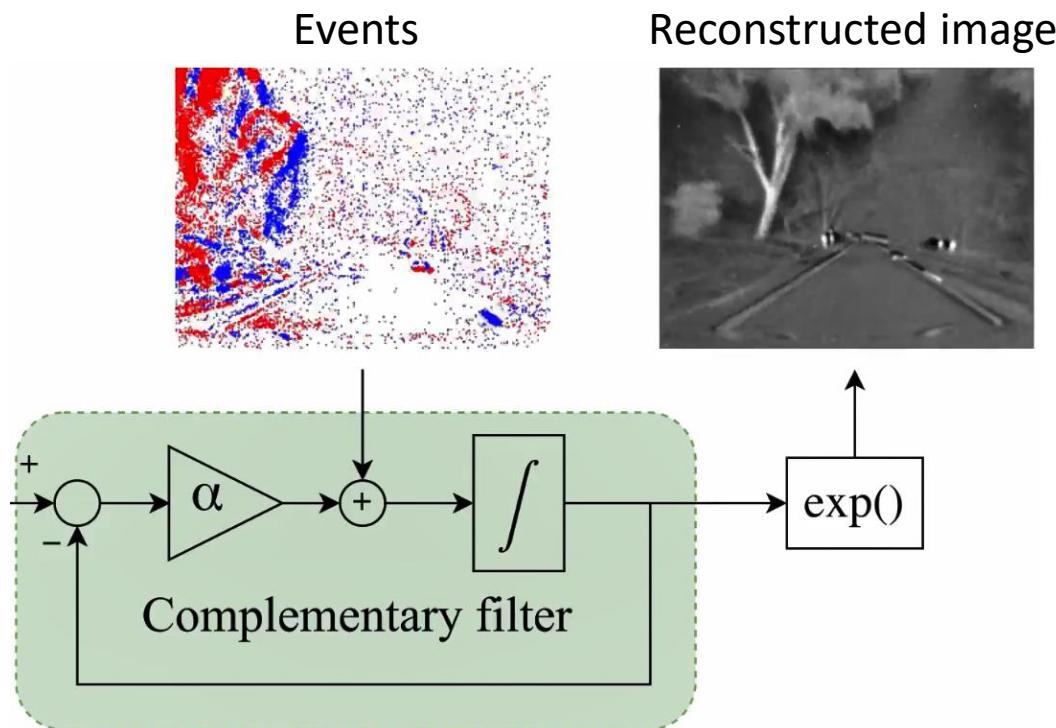
High-pass filtered (in time)



But slow-time information (low freq.)  
(static background) is lost.  
⇒ fuse with grayscale frames from DAVIS  
(complementary filter)

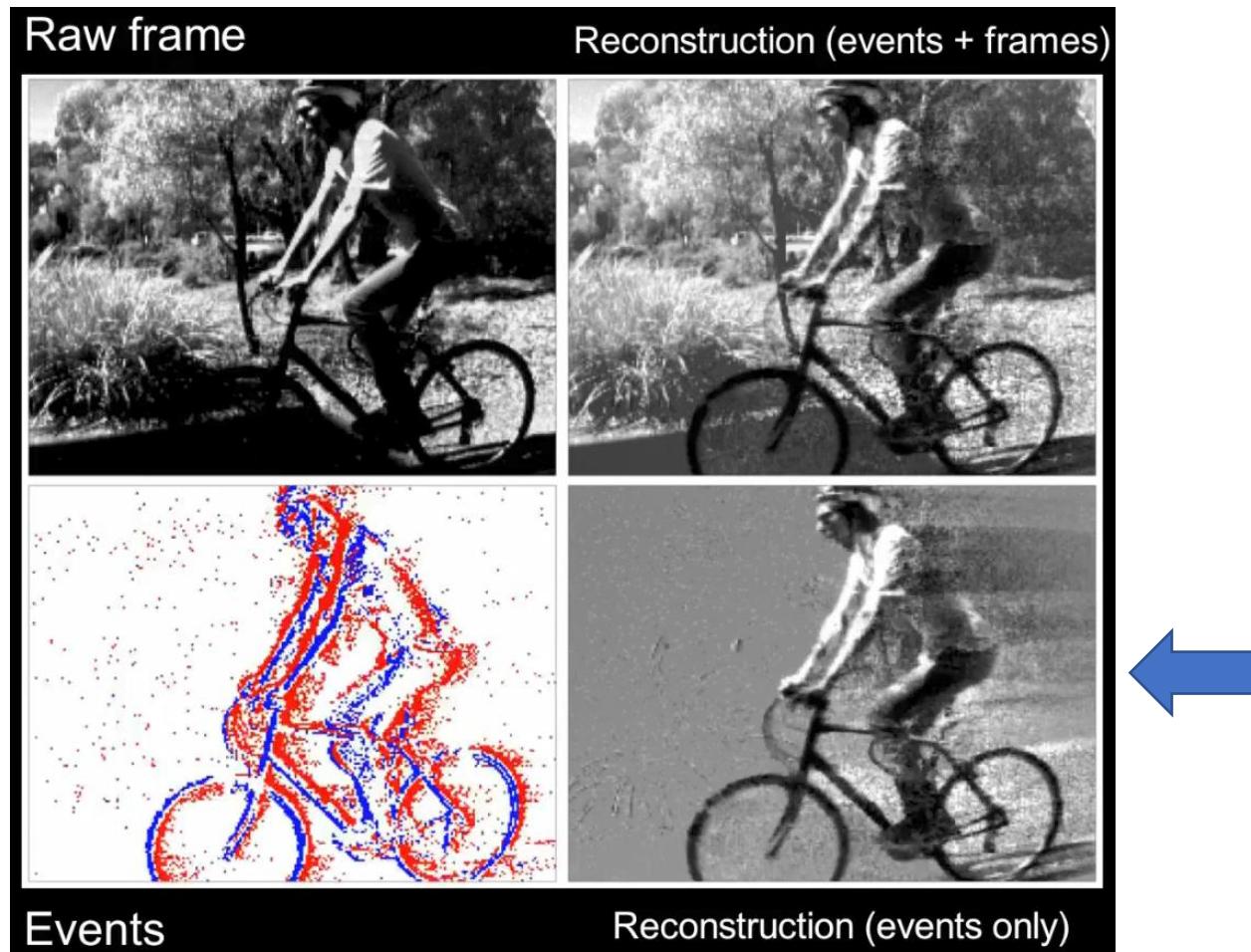
# Reconstruction by Temporal Filtering

- Replace **pixel-wise** direct integration with a **high-pass filter** to remove accumulated event noise
- The internal **state of the filter** is an **image** that is updated **asynchronously, per-pixel** with each incoming event



# Reconstruction by Temporal Filtering

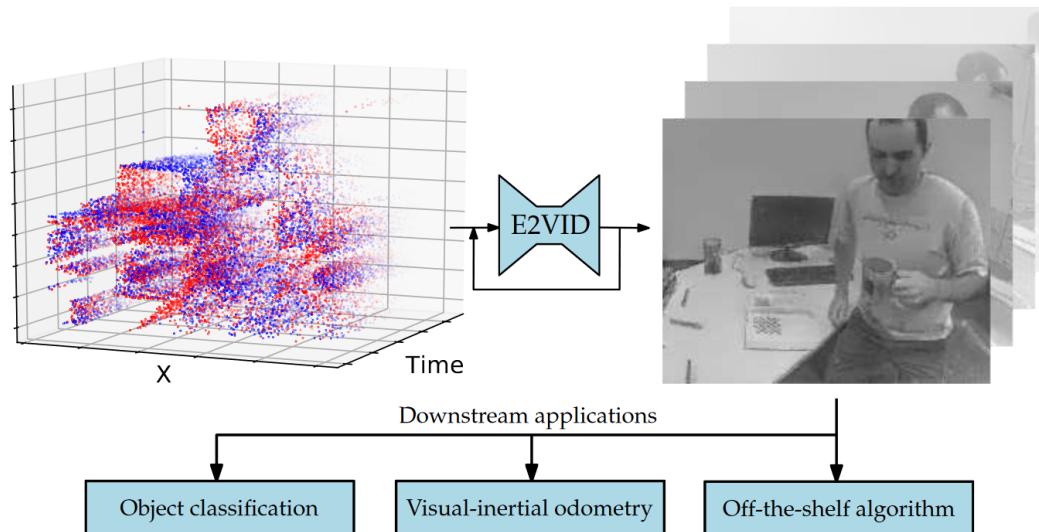
- Replace **pixel-wise** direct integration with a **high-pass filter** to remove accumulated event noise



# Reconstruction using Deep-Learning

- **Events-to-Video (E2VID)**

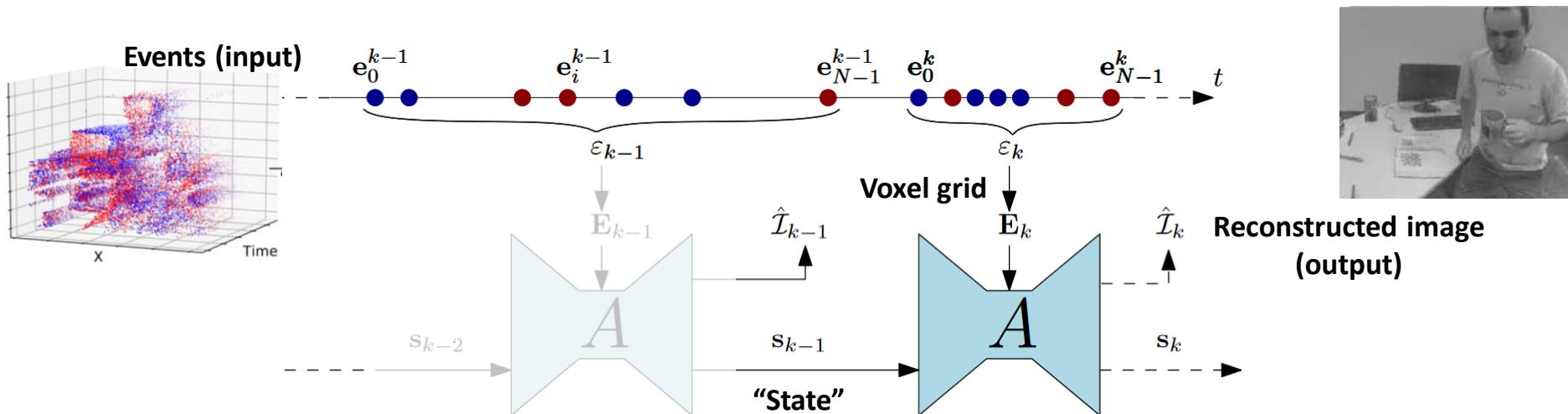
- Deep Learning method: **recurrent network** (with a U-Net)
- **Loss function: perceptual** (LPIPS) + temporal consistency
- Trained on simulation, transfers well to real-world data
- Shows a **big improvement** with respect to previous methods
- Shows reconstructed images can be used on **off-the-shelf computer vision methods** designed for image data



# Reconstruction using Deep-Learning

- **Events-to-Video (E2VID)**

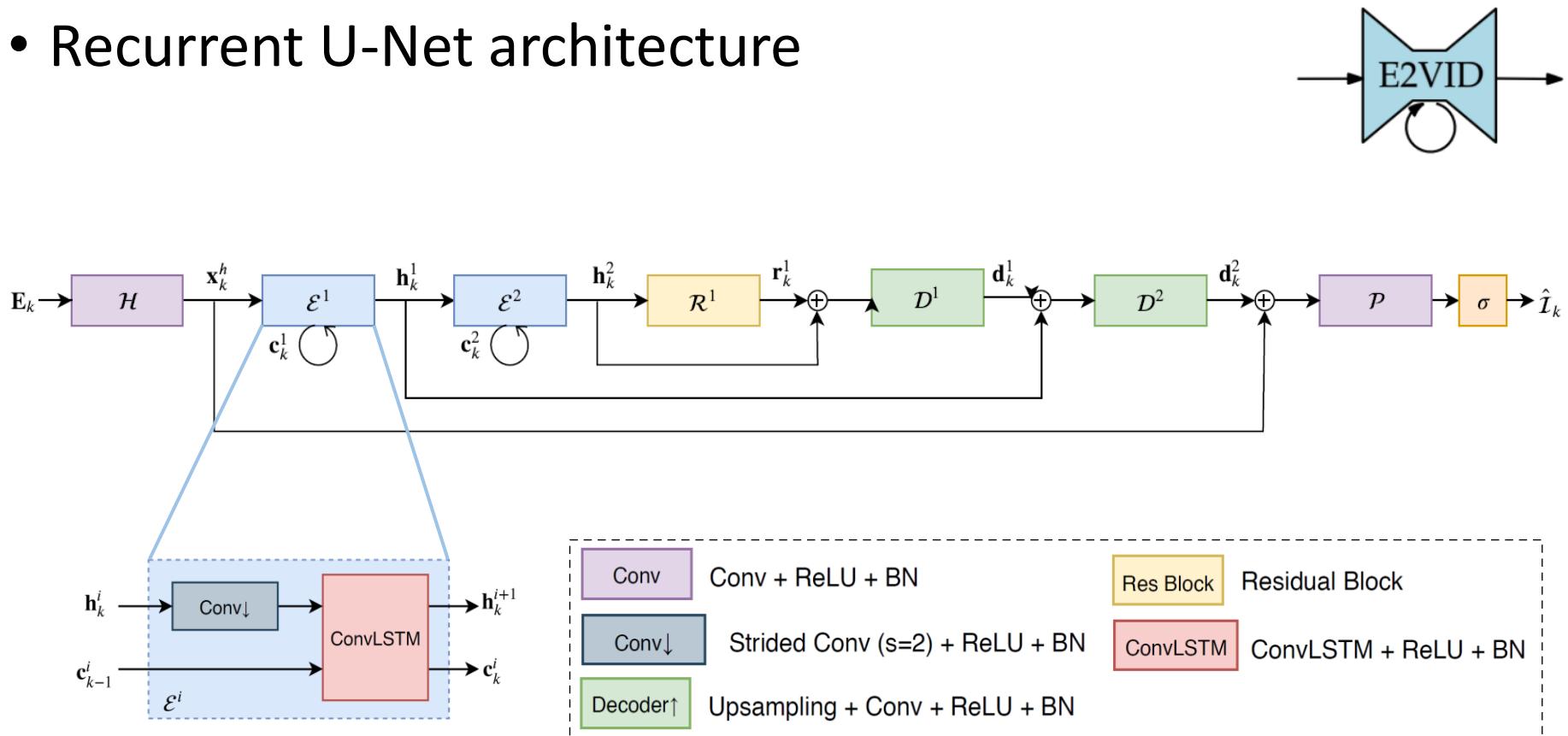
- Deep Learning method: **recurrent network** (with a U-Net)
- **Loss function: perceptual (LPIPS) + temporal consistency**
- Trained on simulation, transfers well to real-world data
- Shows a **big improvement** with respect to previous methods
- Shows reconstructed images can be used on **off-the-shelf computer vision methods** designed for image data



Reconstructed image  
(output)

# Network architecture

- Recurrent U-Net architecture



# E2VID - Results



Huawei P20 Pro (240 FPS)



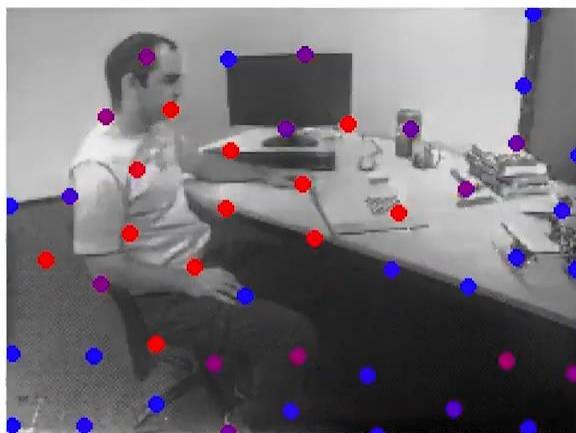
Our reconstruction (5400 FPS)

100 x slow motion

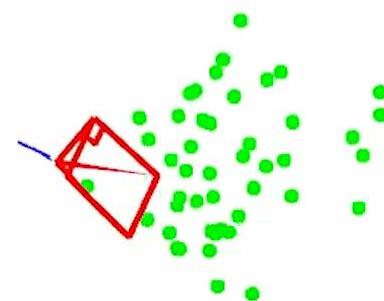
# E2VID - Applications of Reconstructed images



Events



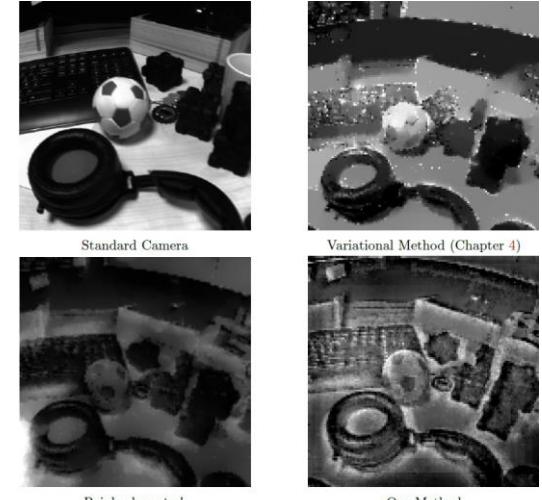
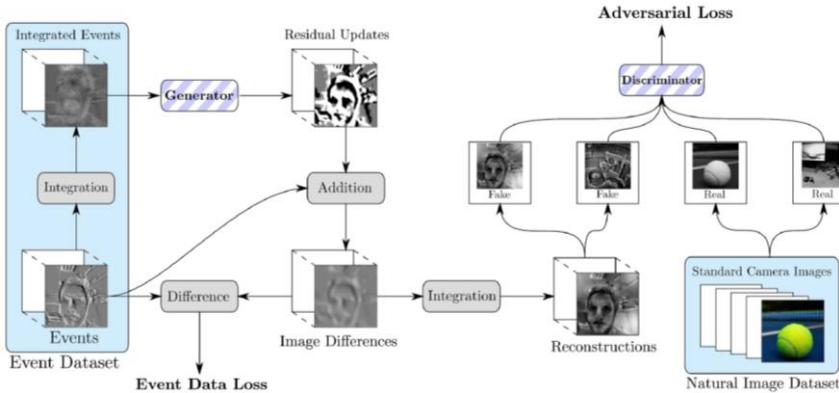
Our reconstruction  
+ tracked features



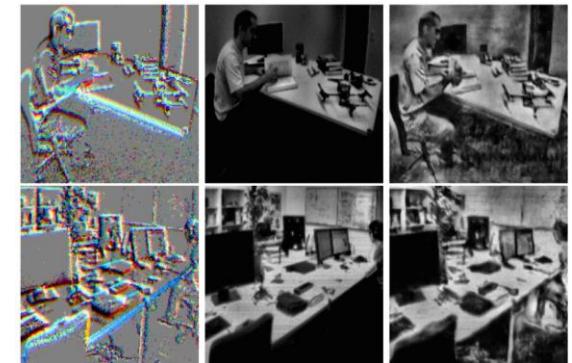
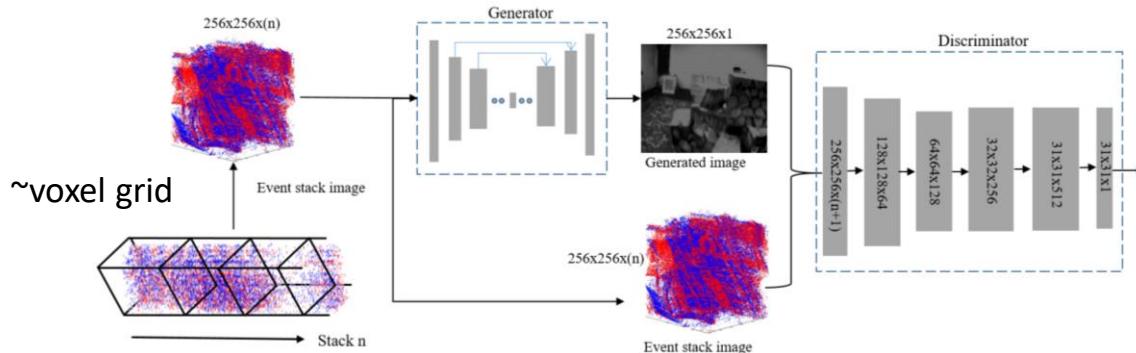
VINS-Mono running on our reconstruction  
from events

# Unsupervised Deep Learning, using GANs

- Bardow (Ch. 6): Reconstruction using only Natural Image Priors



- Mostafavi CVPR 2019



# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

# Discussion

# What enables image reconstruction?

Smoothing or “**Regularization**” (also known as “Prior”)

- In **space**:
  - Neighboring pixels are processed together.
- In **time**:
  - In ACCV 2018 pixels are processed independently, only filtering in time.
- In **space-time**: most methods do this, implicitly.
  - What is the (best) **regularizer / prior**?
    - Hand-crafted? TV-L1, some norm
    - Learned from natural images? LPIPS...
  - What’s the accuracy vs. computational effort **trade-off**?
- As event cameras improve, the image reconstructions will also improve (first sensors were very noisy and of low resolution)

# A variety of Representations and Methods

Just by looking at the task of image reconstruction we find a variety of:

- **Event representations**
  - Event packets, event frames, voxel grids, stacks of events, ...
- **Event processing methods**
  - Temporal filters, EKFs, variational methods, artificial neural networks, GANs, ...

Exercise: go over the paper and identify these (event representations and processing methods used)

# What is image reconstruction good for?

- For **visualization**:
  - Visual feedback (if camera does not output grayscale)
  - Camera Calibration (geometric)
  - High Dynamic Range (HDR) imaging
  - High-speed video generation
- To show the amount of **information contained in the event stream**
- For “**transferability**”: Off-the-shelf computer vision algorithms work very well on the reconstructed images
  - Object classification
  - Visual odometry, depth estimation
  - Labels from image-based datasets can be transferred to events
- As a **baseline** (comparison): how well does my method perform using events directly, as compared to using standard computer vision methods on reconstructed images?

# Is image reconstruction needed?

- It depends on the **task**
  - For ego-motion estimation (SLAM), it is not needed: RAL-17
  - For optical flow estimation it is not needed
  - ...
  - (We will see examples in upcoming lectures)
- But it may be **useful**, as we have seen
  - For video generation, it is essential

# References

## Reading:

- **Section 4.6 of Event-based Vision: A Survey, TPAMI 2020.**
- E. Mueggler et al., *The Event-Camera Dataset and Simulator*, IJRR 2017, page 3.
- List of Event-based Vision Resources, section on “Image reconstruction”:  
[https://github.com/uzh-rpg/event-based\\_vision\\_resources#image-reconstruction](https://github.com/uzh-rpg/event-based_vision_resources#image-reconstruction)

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

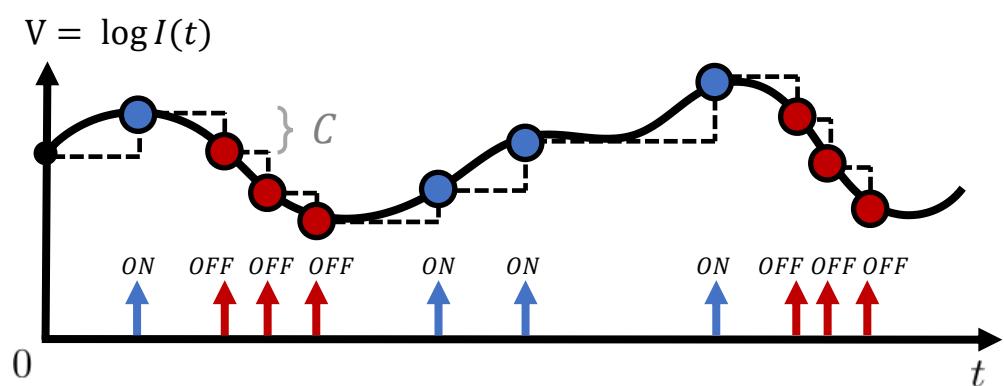
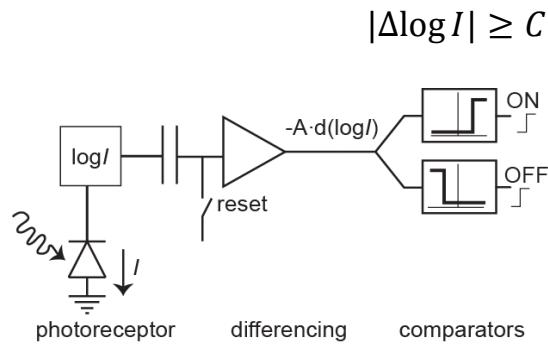
# Case study

Image reconstruction from events in case of rotating event camera with known motion

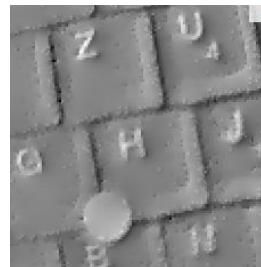
Reference: Kim et al. BMVC 2014

# Image Reconstruction

- Recall: Events are generated any time a single pixel sees a change in brightness larger than  $C$



The intensity signal at the event time can be reconstructed by **integration** of  $\pm C$



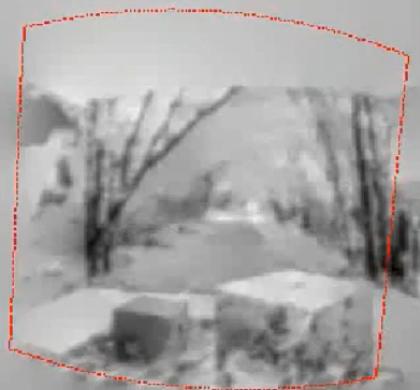
Cook et al., IJCNN 2011



Kim et al., BMVC 2014

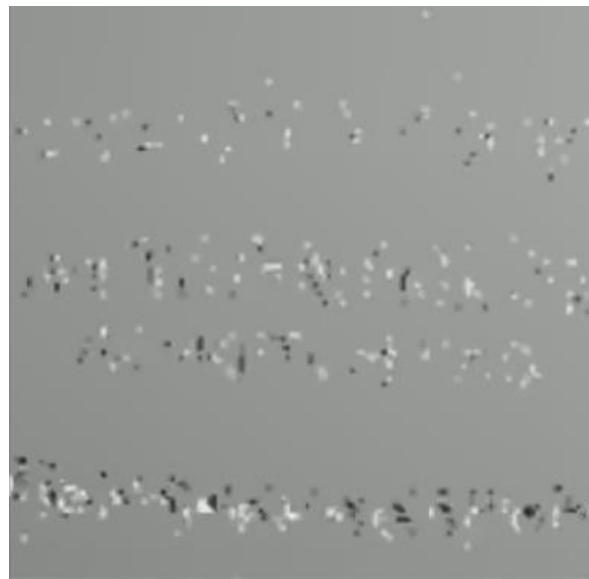
# Image reconstruction

Given the **events** and the **camera motion** (rotation),  
recover the **absolute brightness**



# Image reconstruction

High Dynamic Range property comes “for free” with the sensor



Input events



**HDR** reconstructed intensity



Standard camera  
(narrow dynamic range)

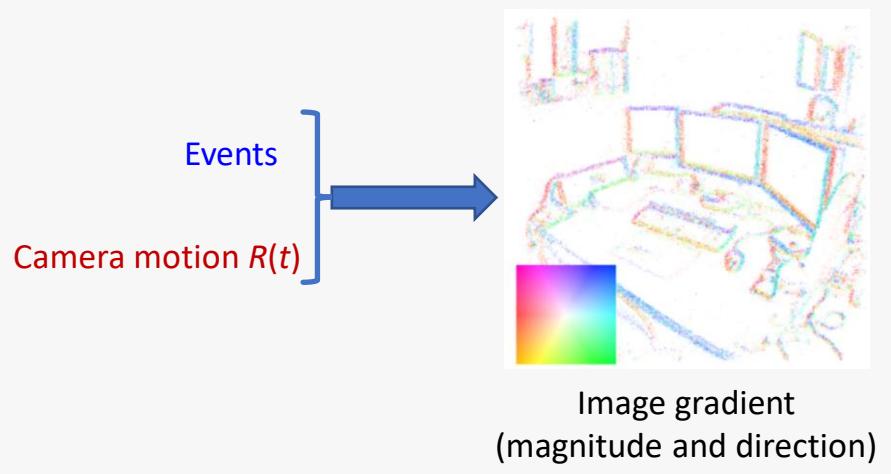
# Image reconstruction

Given the **events** and the **camera motion** (rotation), recover the **absolute brightness**

- How is it possible?
- Intuitive explanation: an event camera naturally responds to edges, hence, if we know the motion, we can relate the events to “world coordinates” to get an edge/gradient map. Then, integrate the gradient map to get absolute intensity.

Steps:

1. Recover the gradient map of the scene

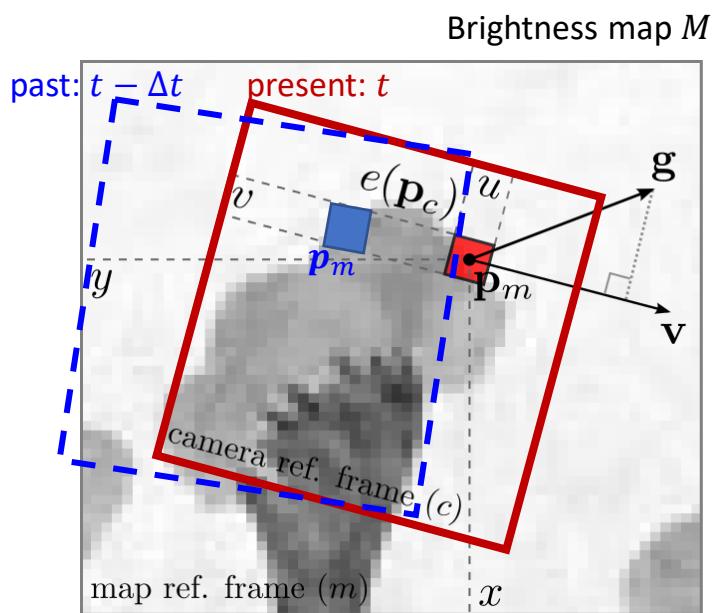


2. Integrate the gradient to obtain brightness



# Image reconstruction

## Step 1: compute gradient map



Event generated due to brightness change of size  $C$

Let  $L = \log I$ ,

$$\Delta L(t) \equiv L(t) - L(t - \Delta t) = C$$

In terms of the brightness map  $M(x, y)$  (panorama):

$$M(\mathbf{p}_m(t)) - M(\mathbf{p}_m(t - \Delta t)) = C$$

Using Taylor 1<sup>st</sup> order approximation:

$$M(\mathbf{p}_m(t)) - M(\mathbf{p}_m(t - \Delta t)) \approx \mathbf{g} \cdot \mathbf{v} \Delta t$$

where brightness gradient  $\mathbf{g} = \nabla M(\mathbf{p}_m(t))$

displacement  $\mathbf{v} \Delta t = (\mathbf{p}_m(t) - \mathbf{p}_m(t - \Delta t))$

# Image reconstruction (detailed)

## Step 1: compute gradient map

Idea: use the **event generation model** and set up an **EKF for every map pixel** (mosaic)

- **Event generation model:**

An event is fired at a pixel  $\mathbf{p}_c$  if there is a brightness change of size  $C$ .

Letting  $L = \log I$ ,

$$\Delta L(t) \equiv L(t) - L(t - \Delta t) = C$$

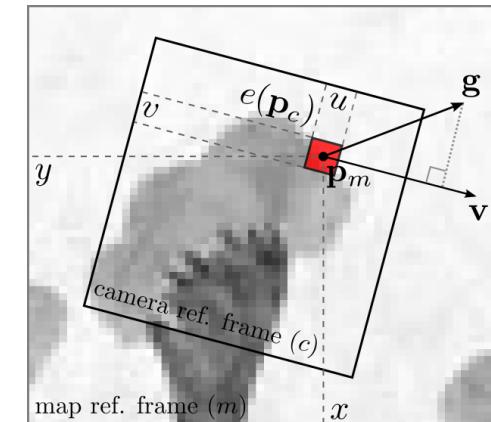
- This is an eq. on the image plane.

Let us **write it in map coordinates**:

- Let  $M(x, y)$  be the brightness of the world map (mosaic)
- At  $t$ , the pixel  $\mathbf{p}_c$  sees the map point  $\mathbf{p}_m(t)$ ,  
where the brightness is  $L(t) = M(\mathbf{p}_m(t))$
- At  $t - \Delta t$ , the pixel  $\mathbf{p}_c$  saw the point  $\mathbf{p}_m(t - \Delta t)$ ,  
where the brightness was  $L(t - \Delta t) = M(\mathbf{p}_m(t - \Delta t))$

Hence, the brightness change in terms of the map is:

$$C = \Delta L = \Delta M = M(\mathbf{p}_m(t)) - M(\mathbf{p}_m(t - \Delta t))$$



# Image reconstruction (detailed)

## Step 1: compute gradient map

### Event generation model

The brightness change in terms of the map is:

$$\Delta M = M(\mathbf{p}_m(t)) - M(\mathbf{p}_m(t - \Delta t)) = C$$

### Linearization:

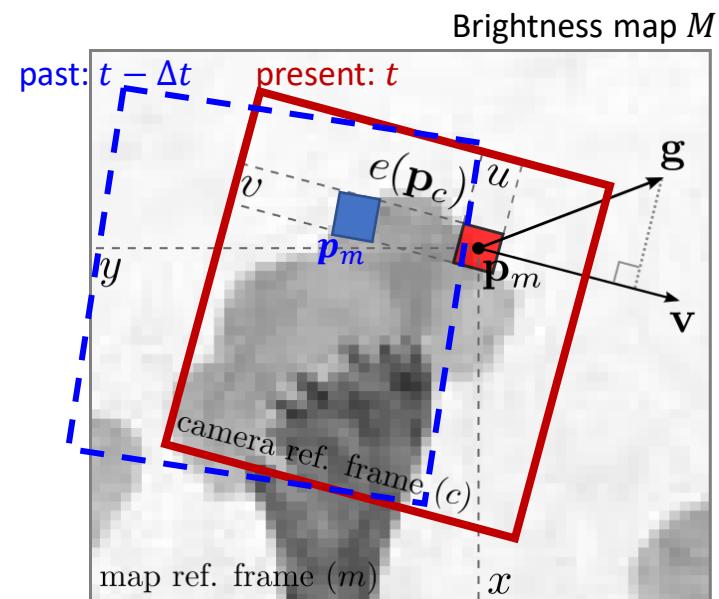
To first order, the brightness change can be computed using the map slope (i.e., gradient). By Taylor's approximation,

$$M(\mathbf{p} + \Delta\mathbf{p}) \approx M(\mathbf{p}) + \nabla M(\mathbf{p}) \cdot \Delta\mathbf{p}$$

$$M(\mathbf{p}_m(t)) - M(\mathbf{p}_m(t - \Delta t)) \approx \underbrace{\nabla M(\mathbf{p}_m(t))}_{g} \cdot \underbrace{(\mathbf{p}_m(t) - \mathbf{p}_m(t - \Delta t))}_{\Delta\mathbf{p}}$$

Assuming a linear motion between the map points,  $\Delta\mathbf{p} \approx \mathbf{v}\Delta t$   
Hence, the event generation in terms of the map is

$$\Delta M = g \cdot v \Delta t \approx C$$



# Image reconstruction

## Step 1: compute gradient map

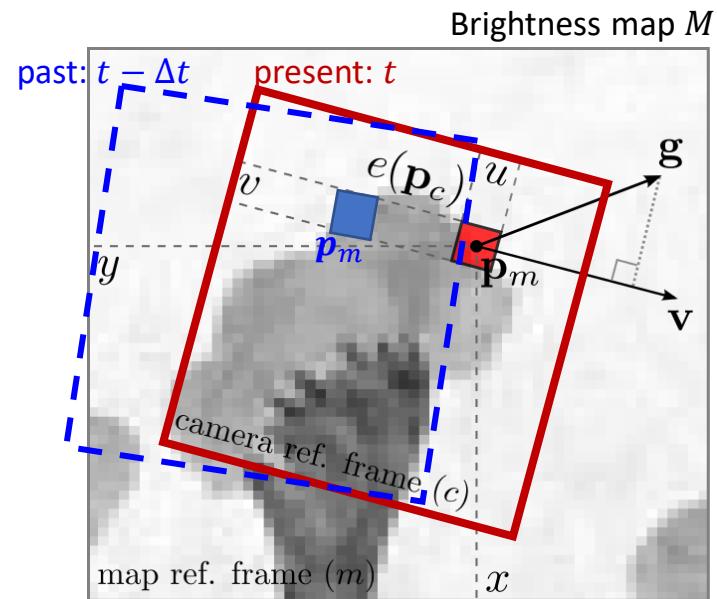
### Event generation model

The event generation in terms of the map is

$$\Delta M = \mathbf{g} \cdot \mathbf{v} \Delta t \approx C$$

Similar interpretation as on the image plane:  
the contrast  $\Delta M \propto \mathbf{g} \cdot \mathbf{v}$  is proportional to  
the dot product between:

- the brightness gradient  $\mathbf{g}$
- the “motion flow”  $\mathbf{v}$  of points on the map



Extreme cases:

- $\mathbf{g}$  and  $\mathbf{v}$  are perpendicular  $\rightarrow \Delta M = 0$ . No event is generated
- $\mathbf{g}$  and  $\mathbf{v}$  are parallel  $\rightarrow \Delta M = C$ , events are triggered the fastest (minimum  $\Delta t$ )

# Image reconstruction

## Step 1: compute gradient map

### Extended Kalman Filter

The event generation in terms of the map is

$$\mathbf{g} \cdot \mathbf{v} \Delta t \approx C$$

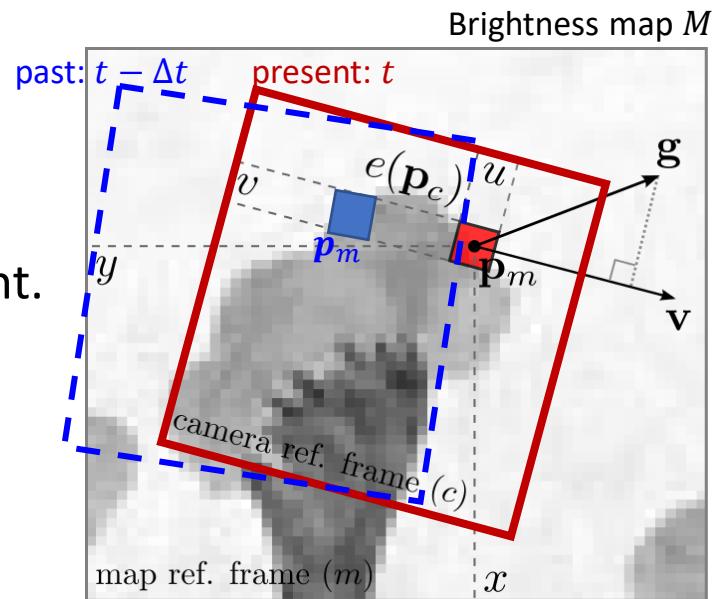
In this equation, only  $\mathbf{g}$  is unknown, and it is constant.

Let us use an EKF to estimate this constant vector.

(one EKF per map point)

- State eq:  $\mathbf{g}_k = \mathbf{g}_{k-1}$
- Observation equation:  $h_k = \frac{\mathbf{g}_k \cdot \mathbf{v}_k}{C}$

Thus, we use the “event rate” at the map point  $p_m$  as observation:  $z_k = \frac{1}{\Delta t}$



# Image reconstruction

## Step 1: compute gradient map

### Extended Kalman Filter

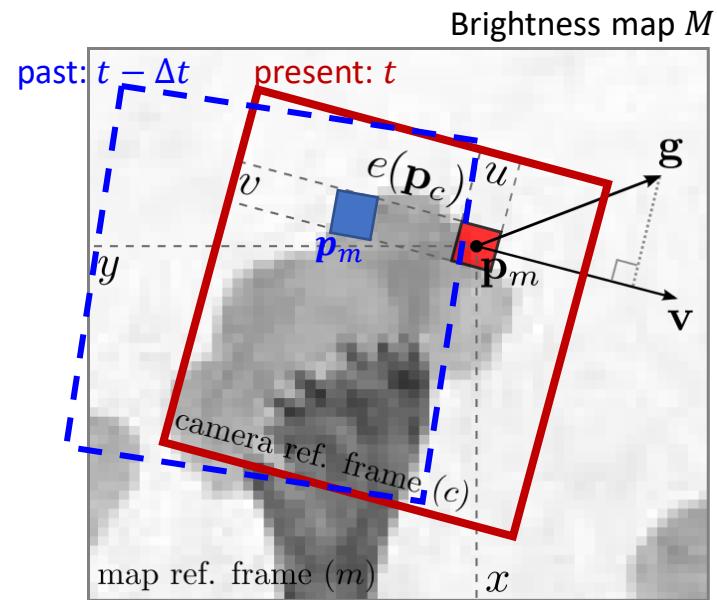
Iteration equations:

- Innovation:  $v_k = z_k - h_k$
- Innovation covariance:  $S_k = H_k P_{k-1} H_k^\top + R_k$ 
  - Measurement matrix is the Jacobian:

$$H_k = \frac{\partial h_k}{\partial g_k} = \frac{v_k}{c}$$

- $R_k$  is the covariance of the measurement  $z_k$
- Kalman gain:  $K_k = P_{k-1} H_k^\top S_k^{-1}$

- Gradient update:  $\mathbf{g}_k = \mathbf{g}_{k-1} + K_k v_k$
- Covariance of gradient update:  $P_k = P_{k-1} - K_k S_k K_k^\top$

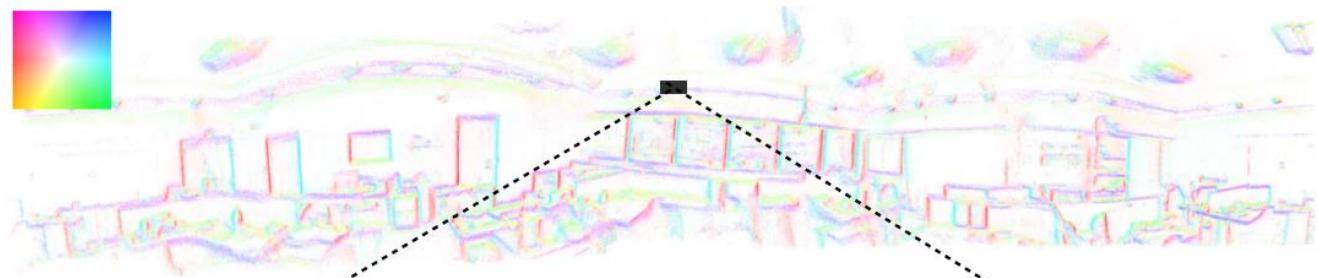


# Image reconstruction

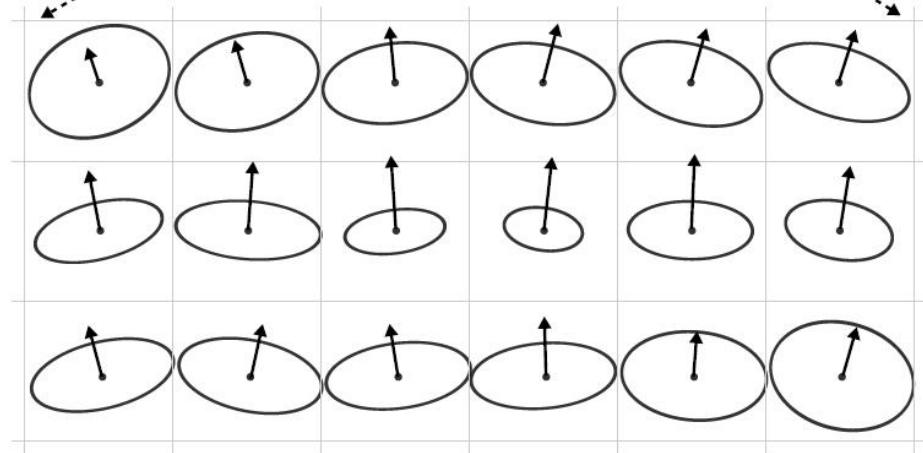
## Step 1: compute gradient map

**Extended Kalman Filter** (one per map pixel)

Gradient map:  
magnitude (radius)  
and direction (color)



Arrow: gradient vector  $\mathbf{g}$   
Ellipse: covariance  $P$

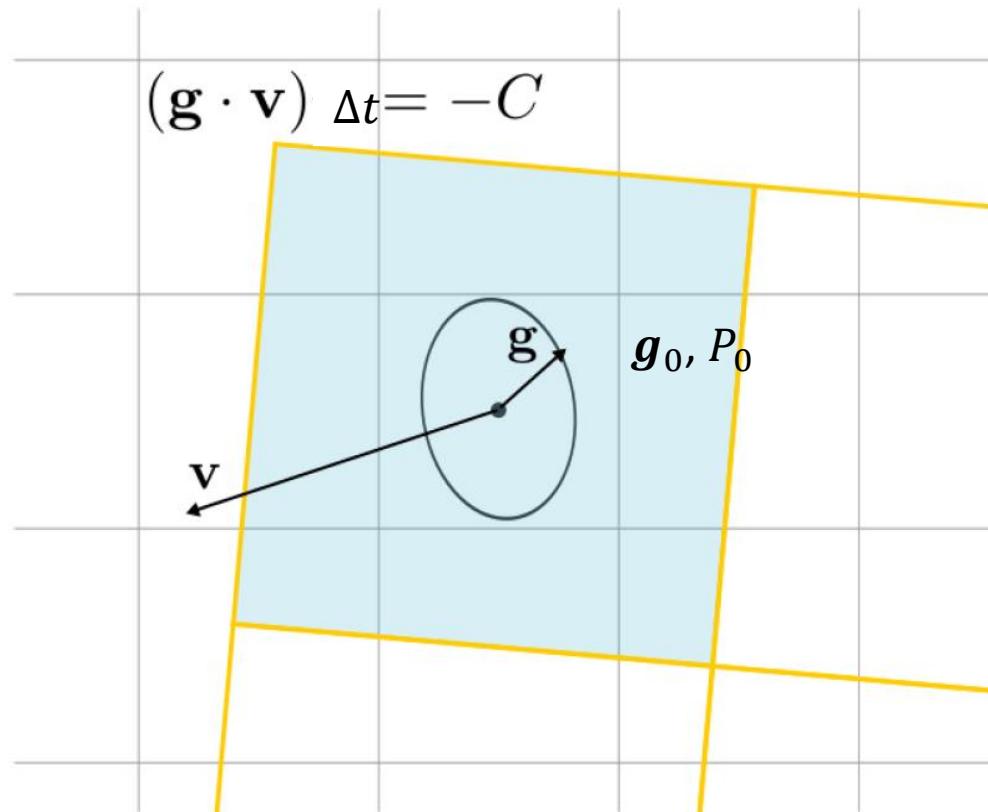


# Image reconstruction

Step 1: compute gradient map

**Extended Kalman Filter** (one per map pixel)

Iterations:

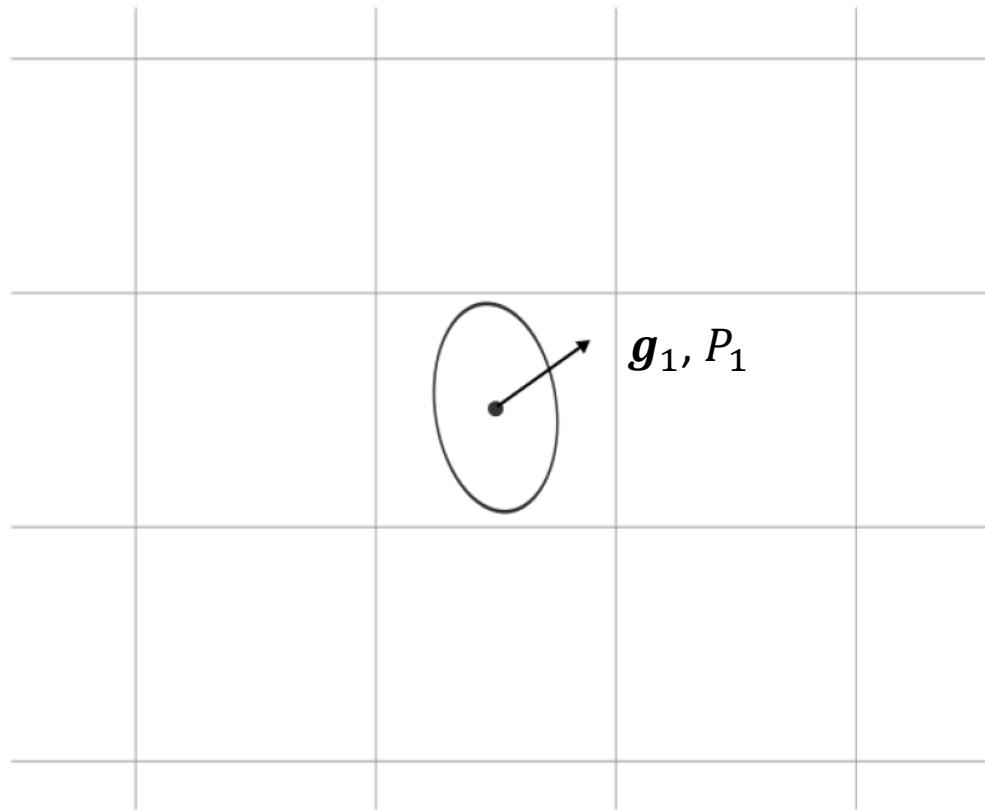


# Image reconstruction

Step 1: compute gradient map

**Extended Kalman Filter** (one per map pixel)

Iterations:

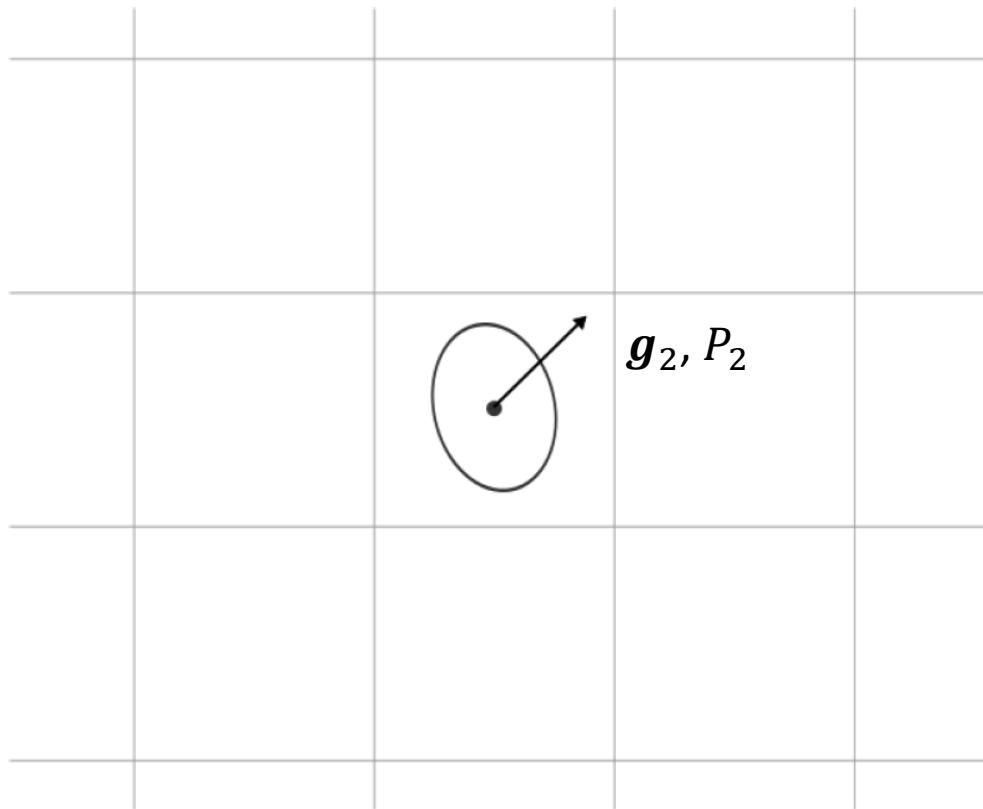


# Image reconstruction

Step 1: compute gradient map

**Extended Kalman Filter** (one per map pixel)

Iterations:

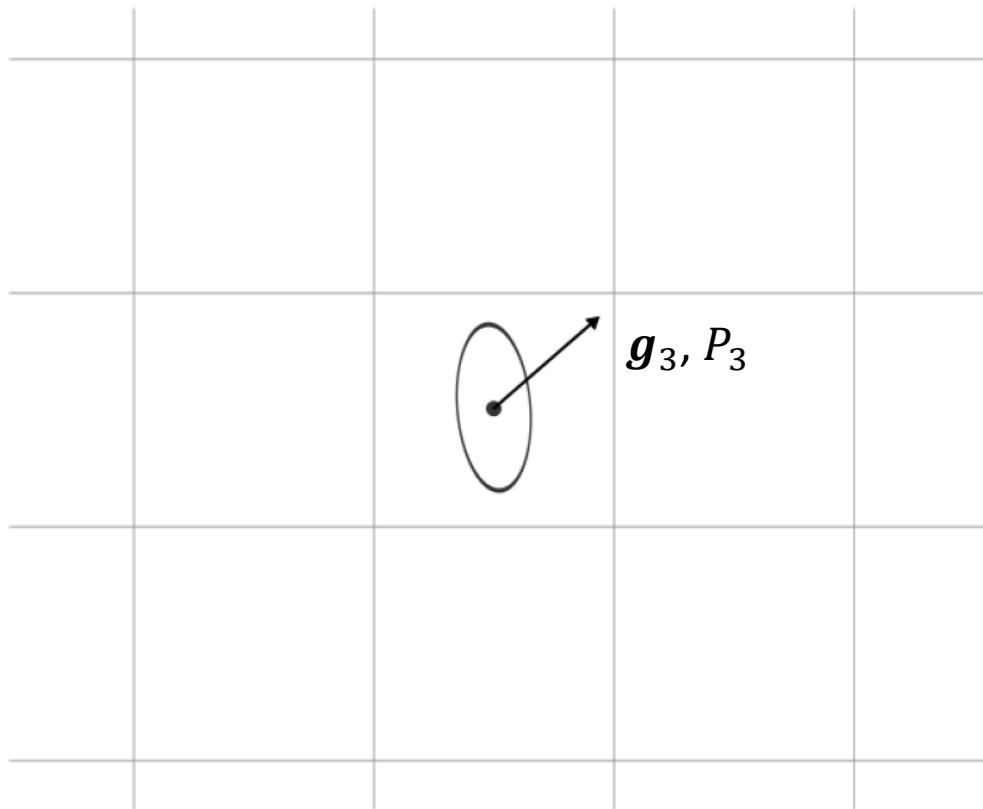


# Image reconstruction

Step 1: compute gradient map

**Extended Kalman Filter** (one per map pixel)

Iterations:

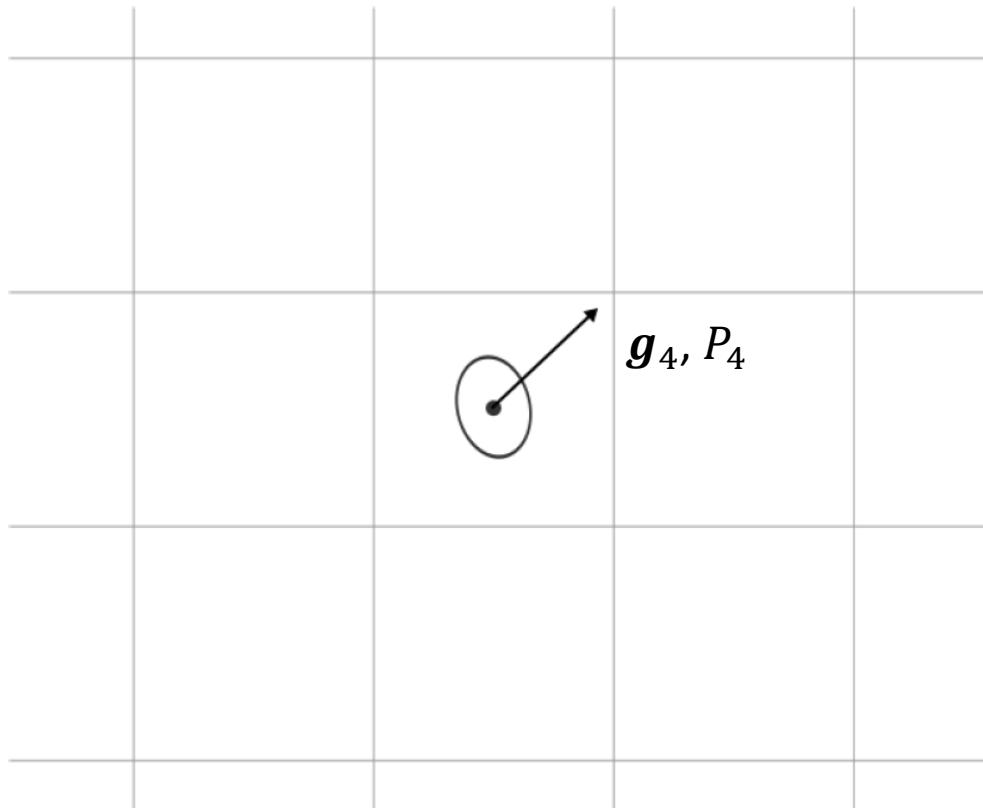


# Image reconstruction

Step 1: compute gradient map

**Extended Kalman Filter** (one per map pixel)

Iterations:

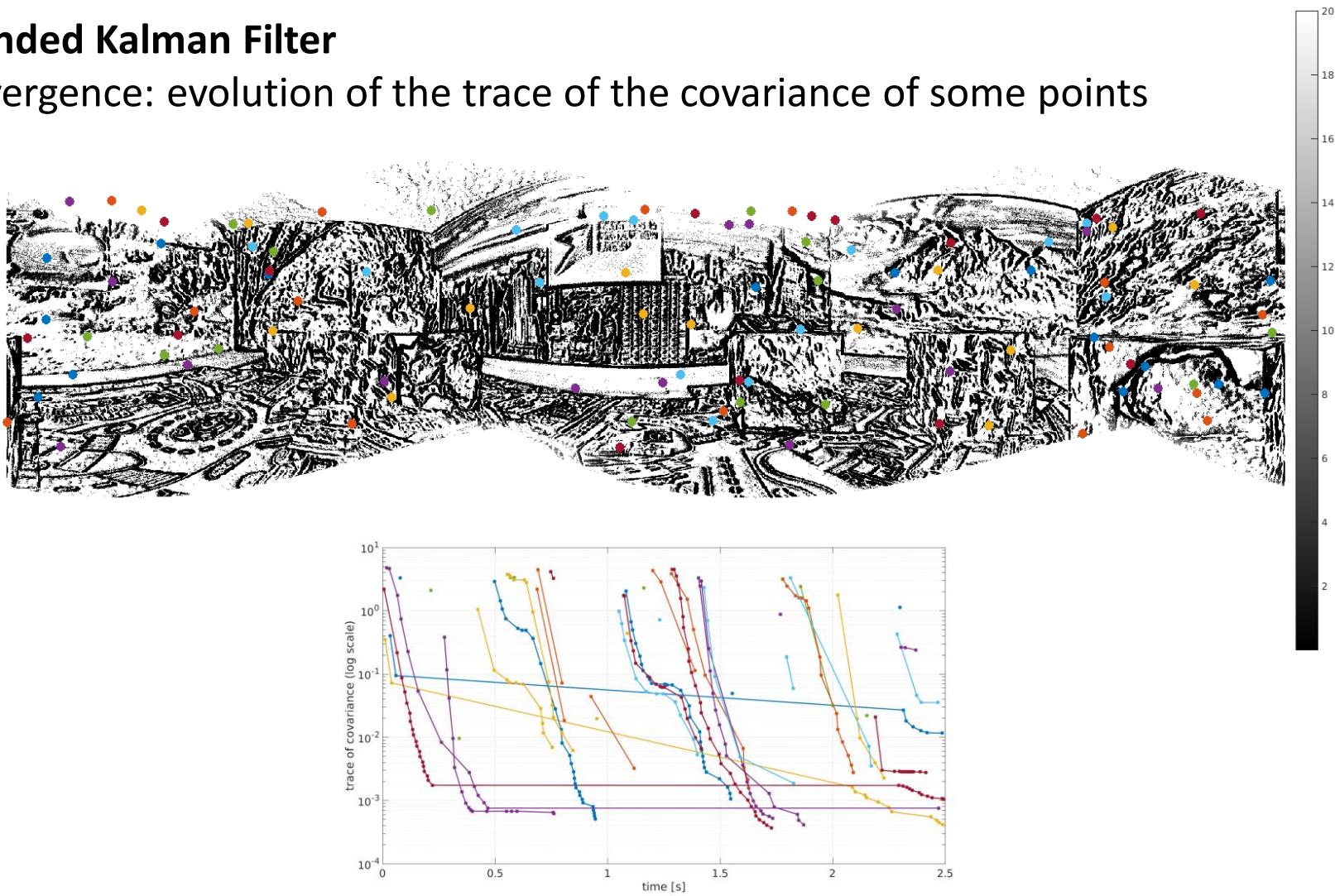


# Image reconstruction

## Step 1: compute gradient map

### Extended Kalman Filter

Convergence: evolution of the trace of the covariance of some points



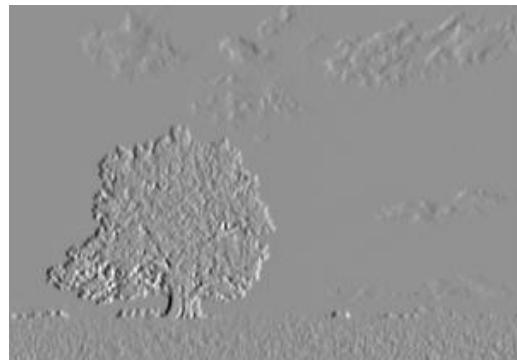
# Image reconstruction

## Step 2: Poisson integration

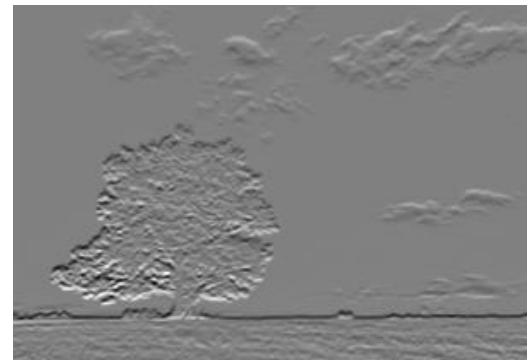
Integrate gradient map  $\mathbf{g}$  to get absolute brightness  $M$



Original Image



Gradient in x direction  
 $(g_x = \partial_x I)$



Gradient in y direction  
 $(g_y = \partial_y I)$

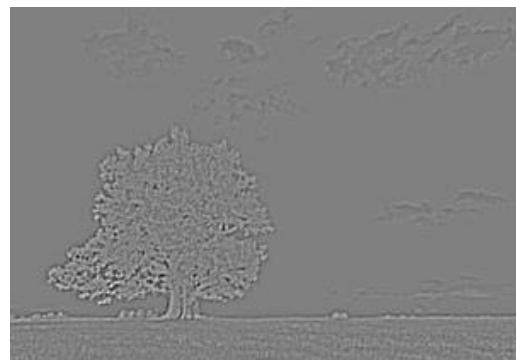


Reconstructed Image

2D Integration

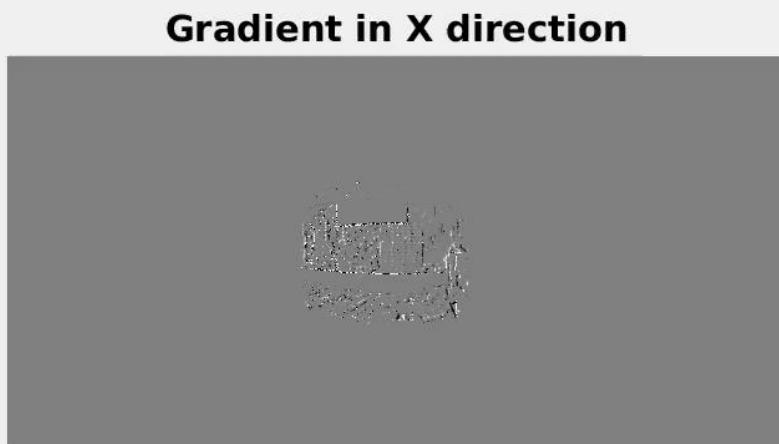
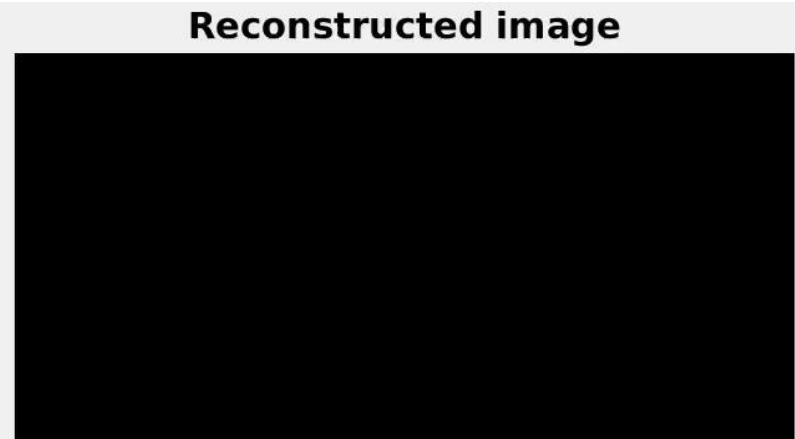
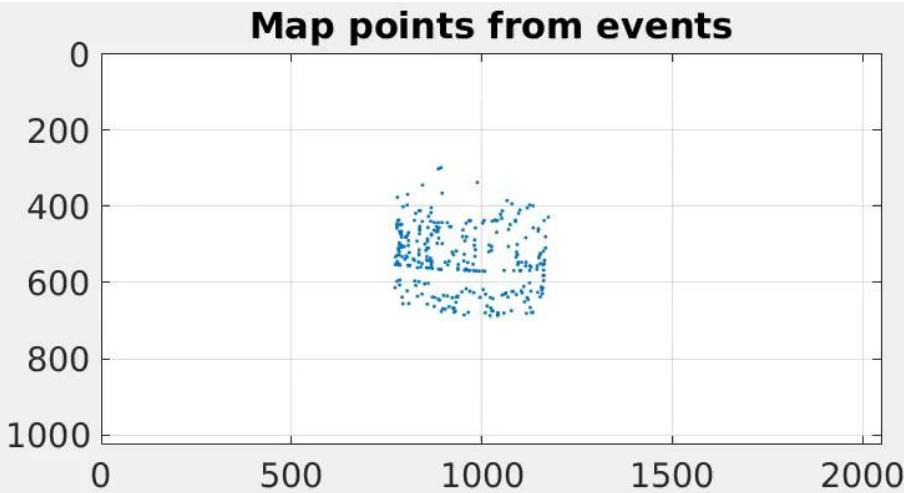


Solve Poisson eq:  
 $(\Delta \tilde{I} = \operatorname{div} \mathbf{g})$   
fast using the FFT



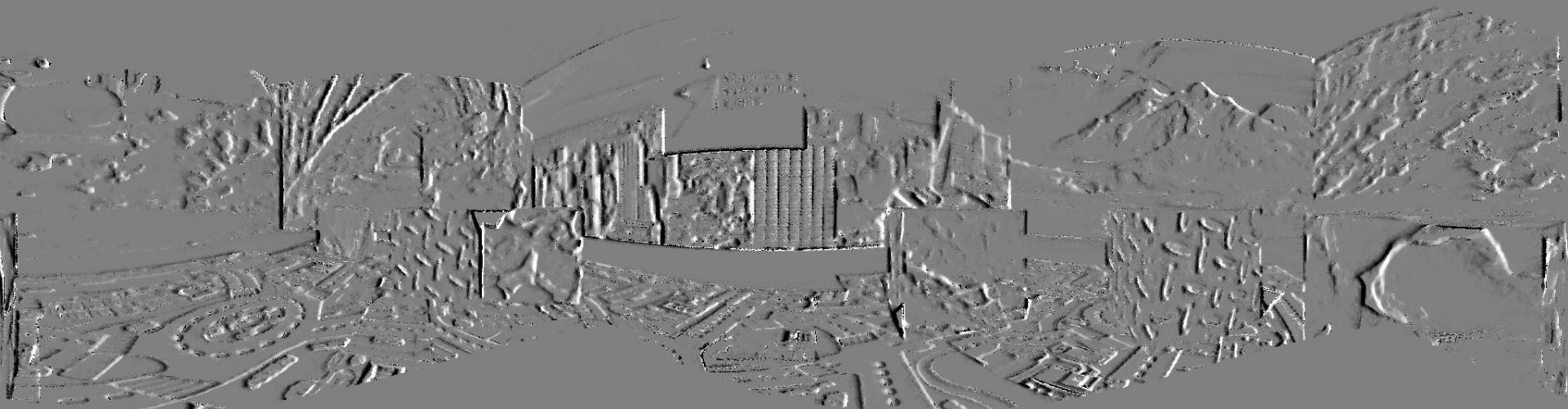
Divergence  
 $(\operatorname{div} \mathbf{g} = \partial_x g_x + \partial_y g_y)$

# Image reconstruction demo



# Image reconstruction demo

Gradient  $\mathbf{g}$  in X direction



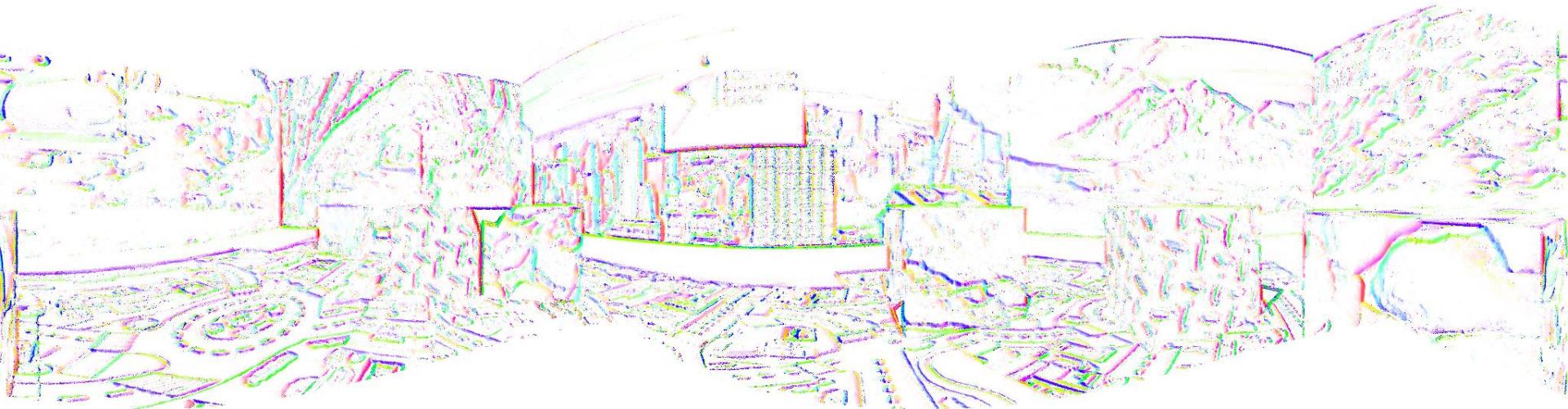
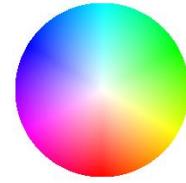
# Image reconstruction demo

Gradient  $g$  in Y direction



# Image reconstruction demo

Gradient  $g$ : magnitude (radius) and orientation (color)



# Image reconstruction demo

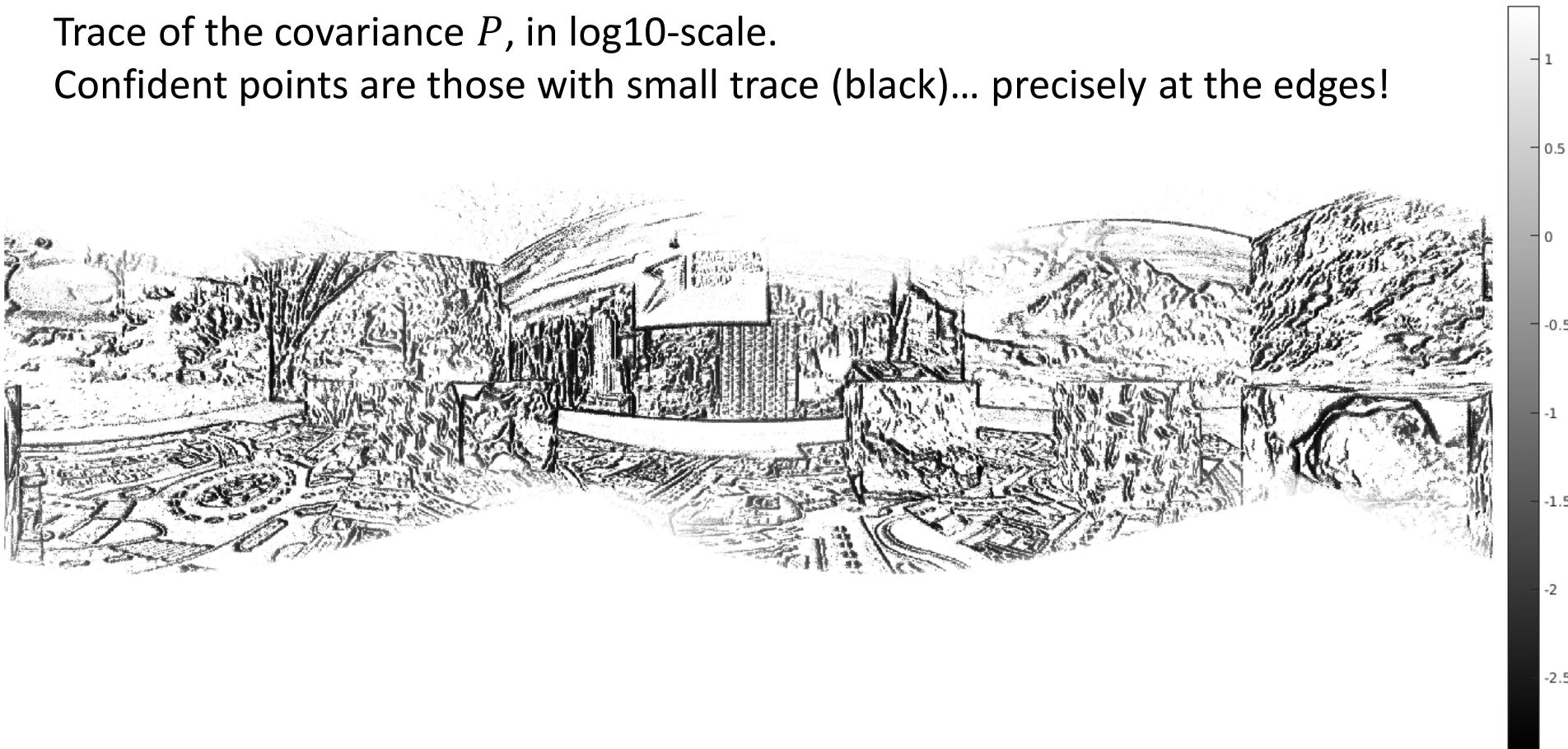
Reconstructed map  $M$ , in log-brightness



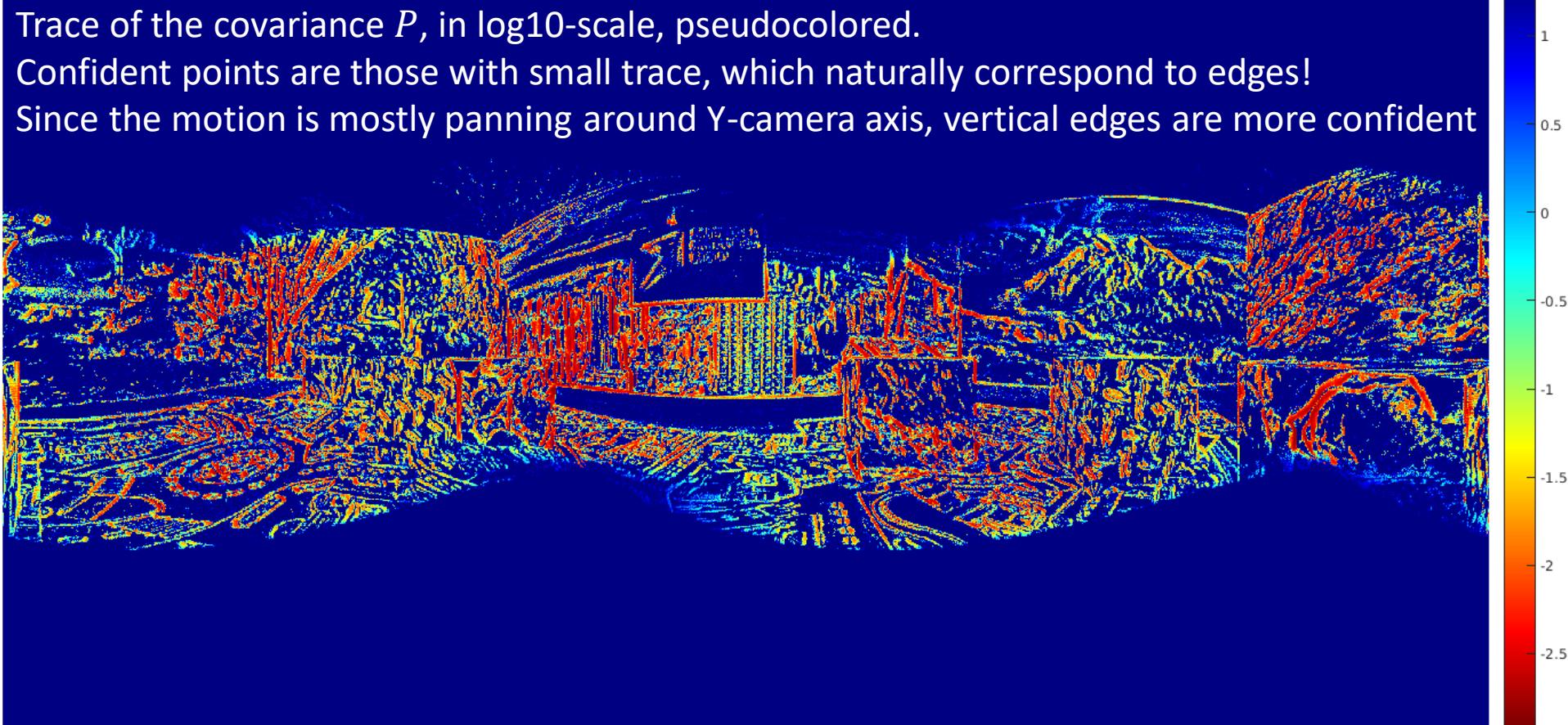
# Image reconstruction demo

Trace of the covariance  $P$ , in log10-scale.

Confident points are those with small trace (black)... precisely at the edges!



# Image reconstruction demo



# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

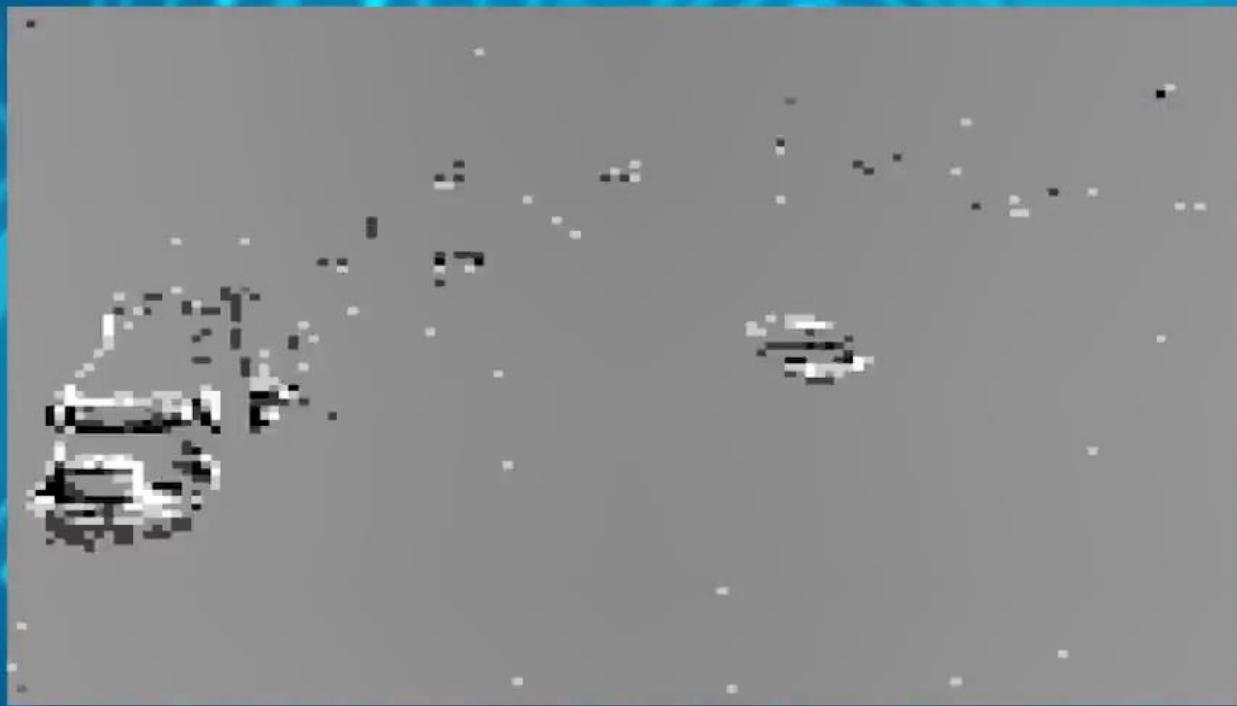
[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

# Blob / Cluster Tracking

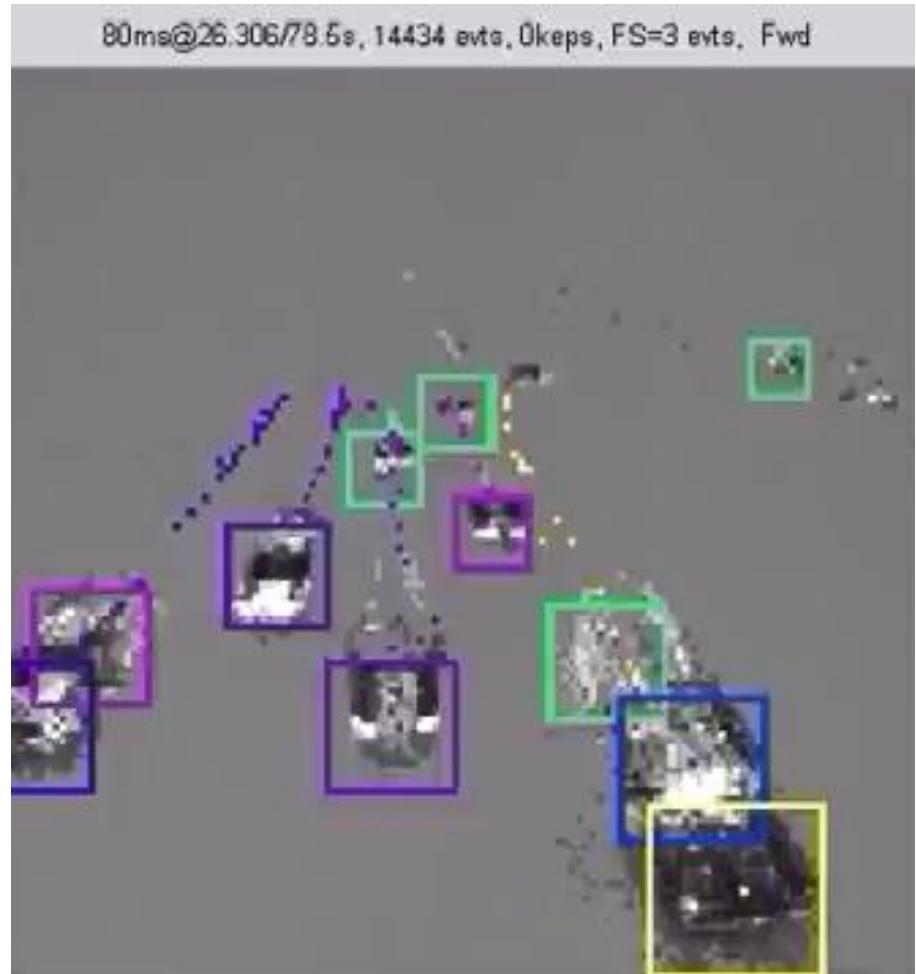
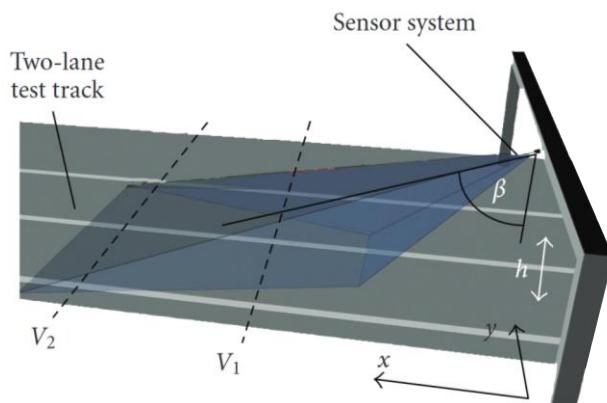
# Blob Tracking for Surveillance

- Application: counting cars and measuring their speed



# Blob Tracking for Surveillance

- Application: counting cars and measuring their speed



# Blob / Cluster Tracking

- It is low-power → Applications in Surveillance and IoT



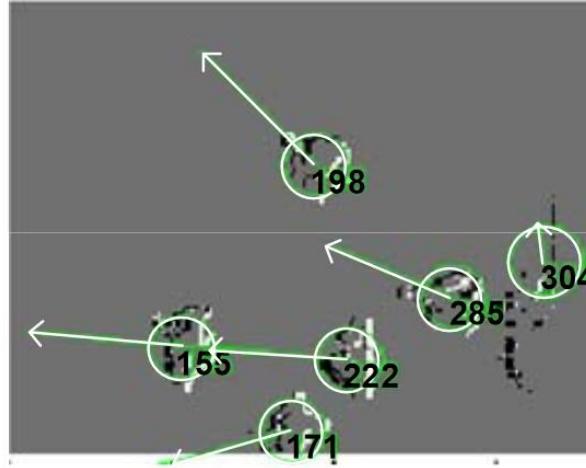
Tracking clusters of events (red and green dots)

# Blob / Cluster Tracking

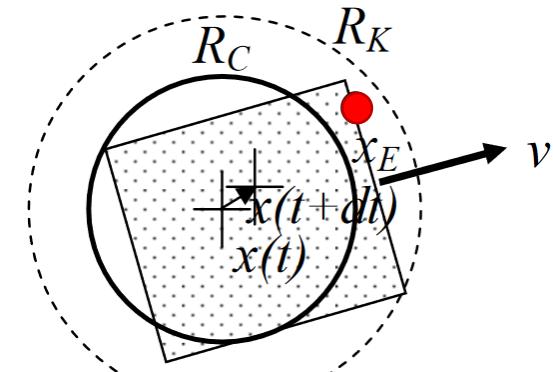
- Objects (cars, people..) are crudely **represented** by Gaussians, boxes.



People (top view)



Blobs tracked using events



Blob & Event in  $[t, t + \Delta t]$

- **How are events processed?**
  - Typically, **one by one**
  - Events are **assigned** to clusters based on spatial vicinity
  - Then, cluster parameters (center, size, velocity) are **updated** to fit the events. The events “drag” the centroid of the cluster.
  - It is like a filter (e.g., Kalman). State: blob parameters

# How to track?

There are **two key ingredients** that are also present in other trackers:

## 1. Data association

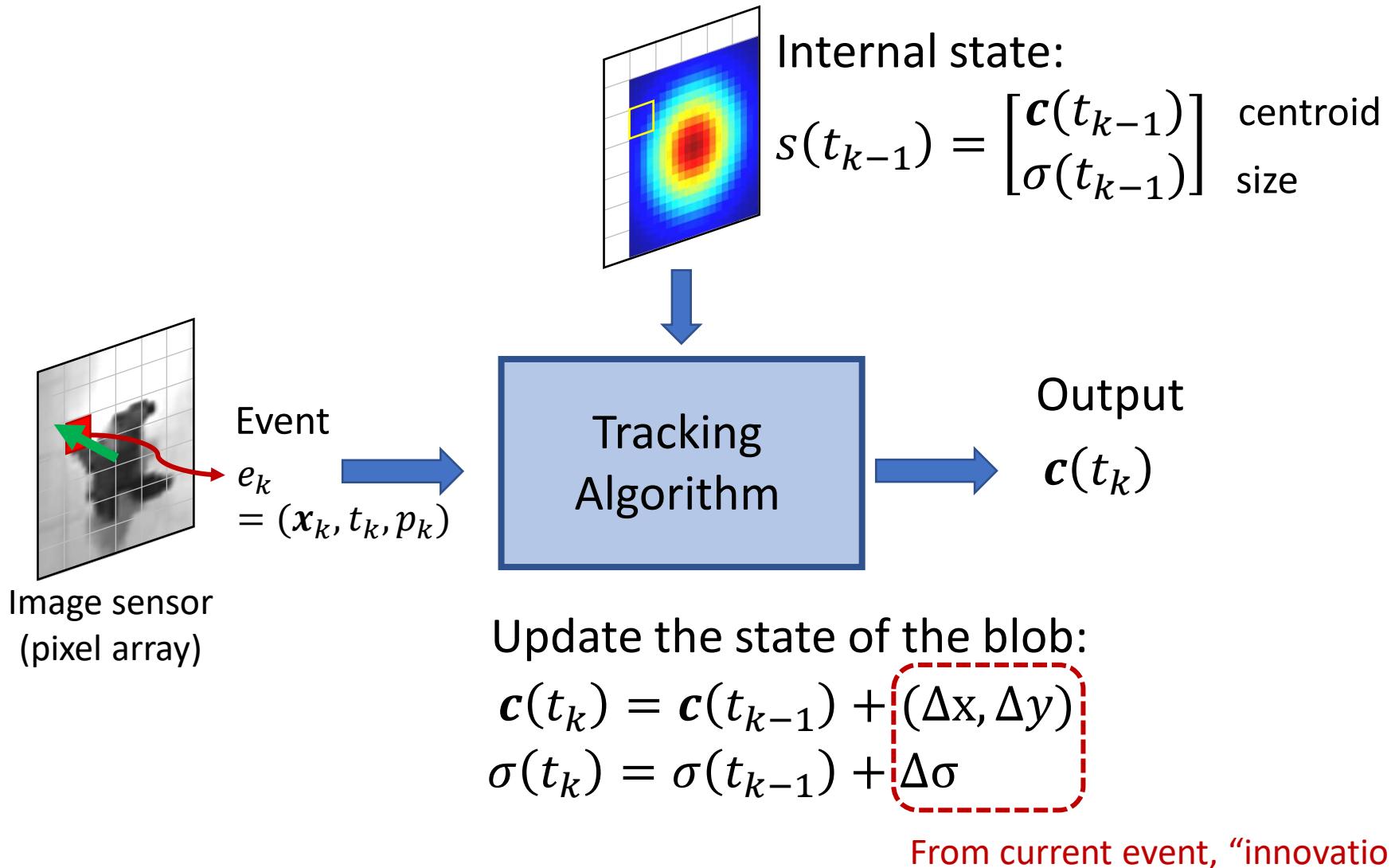
- Q: *To which cluster does this event belong?*
- A: A function that assigns events to tracked objects / clusters

## 2. Update rule

- Q: *How are the object / model parameters updated to assimilate the information given by the new event?*
- A: A function that updates the model and/or its parameters, which are the desired output of the tracker

# The paradigm: Event-by-event Processing

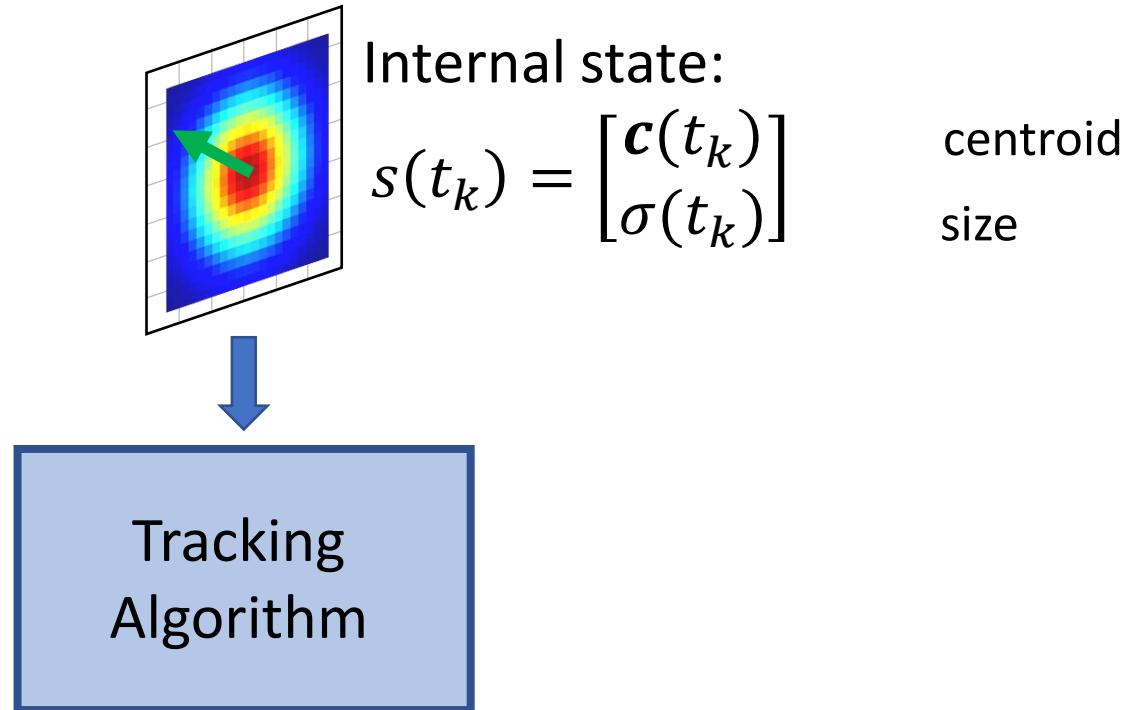
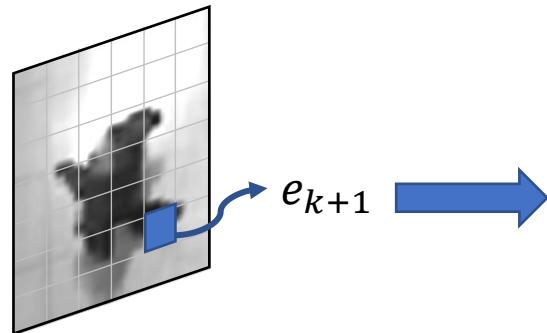
- Case Study: Blob/Cluster Tracking



# The paradigm: Event-by-event Processing

- Case Study: Blob/Cluster Tracking

Ready for next event:



# Video of car-slot racing tracker

- Tracking can be tailored to the problem and very efficient



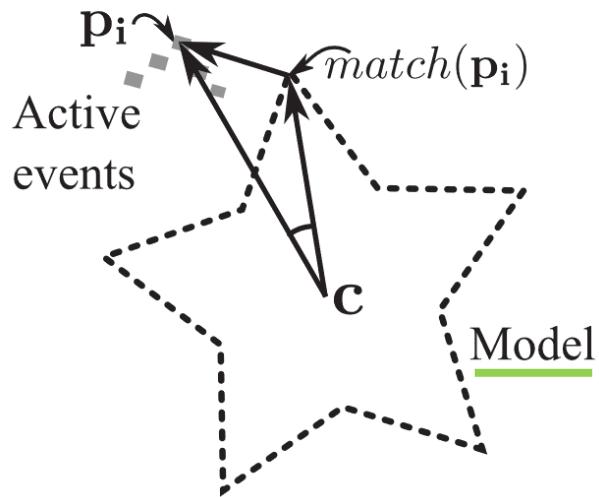
# Blob / Cluster Tracking

- **Pros:**
  - Very efficient. Do not waste effort on pixels with no events
  - Easy to implement
  - Low latency (event-by-event)
- **Cons:**
  - Rough approximation / simplification to shape.  
There is no concept of object; no clear “boundaries” unless enforced by a constrained scenario
  - Jitter or wiggling hinders accuracy. It should be damped
  - Problems arise if objects cross each other (occlusions...)

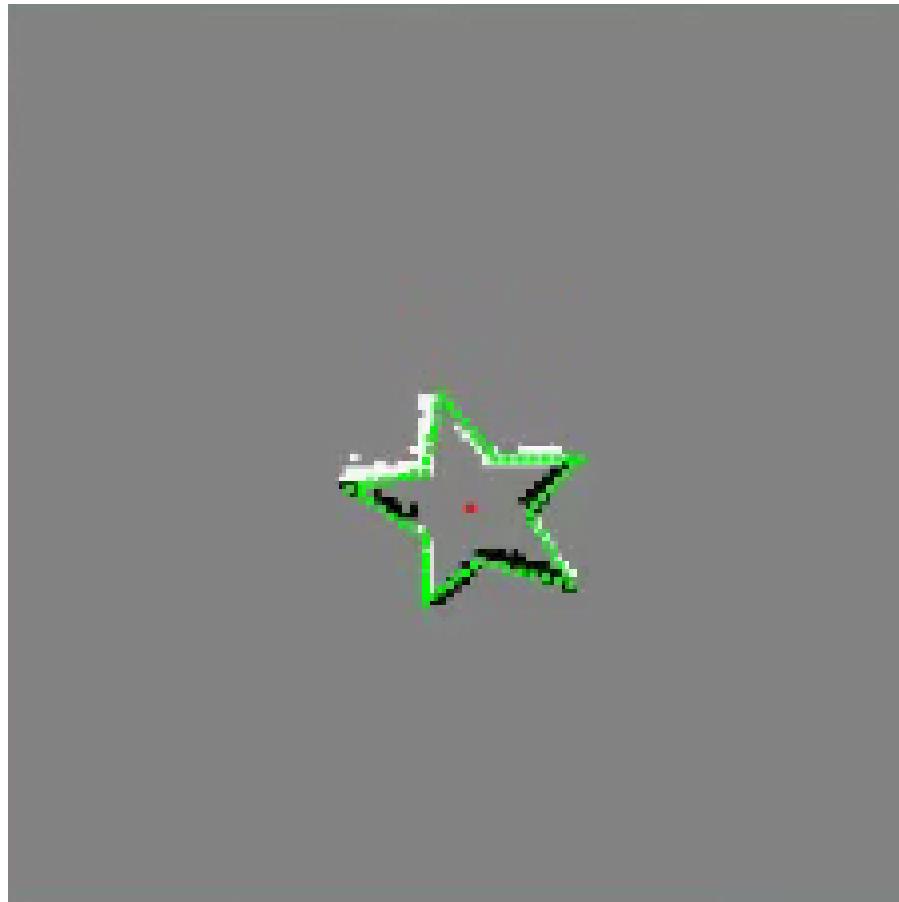
More complex shapes?

# Event-based Pattern Tracking

- Tracking by **registration of two sets of points**:
  - **Data**: events  $\{e_k\}$
  - **Model**: A point set of the 2D shape to be tracked.



**Objective:** Minimize the Euclidean distance between two sets of points  
(data & model)

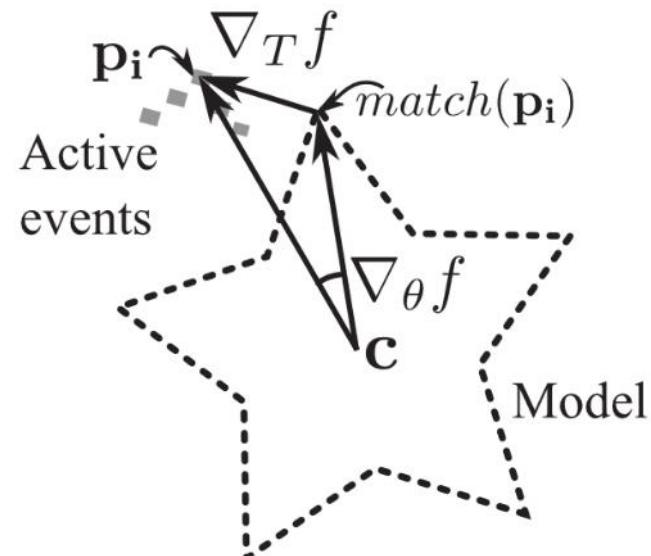


# Event-based Pattern Tracking

- How is each event processed?
  1. A matching function **associates** the event to the model shape (e.g., closest point)
  2. A **registration objective** provides the rules to **geometrically transform** the model shape in order to **best fit** the data. Model parameters  $\theta$ : position, orientation, etc.
    - What transformations are enabled? Translations, rotations, affine...

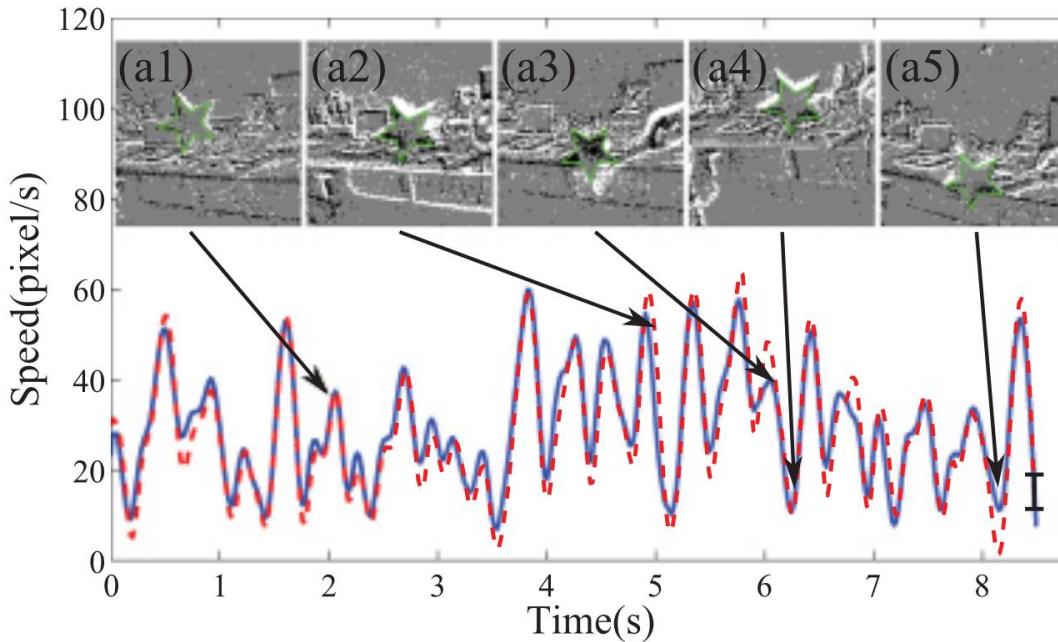
$$\min_{R(\theta) \in SO(2), T \in \mathbb{R}^2} \sum f_i$$
$$f = \| q_i - R(\theta)(\text{match}(q_i)) - T \|^2$$

(Euclidean distance)



- **Input:** Events and 2D shape model
- **Output:** parameters  $\theta(t_k)$  describing geometric transformation of the model

# Event-based Pattern Tracking



Event frames just for visualization

- **Pros:**

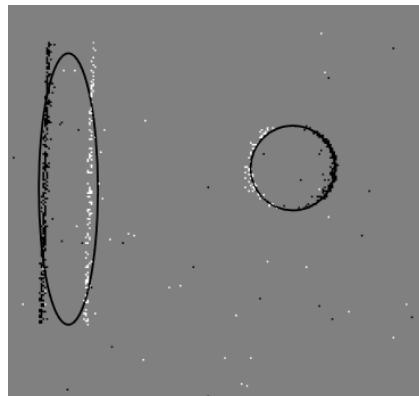
- Works on events directly
- Very efficient and local computations (better than ICP in tests)
- Tracking demonstrated even if camera moves

- **Cons:**

- Shapes are predefined, simple and of high contrast.  
Limited results on natural scenes.

# Tracking shapes specified by kernel masks

- Geometric Primitives of Objects:
  - **User-defined shapes** (“kernels”). Example: ellipse (pos, size, dir)
  - Considers events as points caused by the kernel as it moves

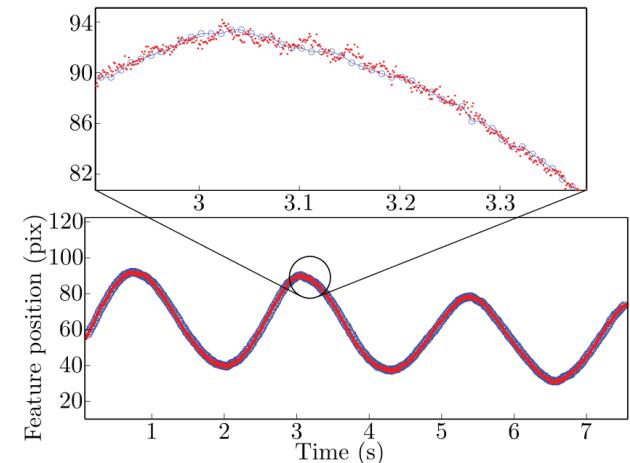


Kernels



Events

- Method:
  - **Event-by-event** processing. Low-latency & real time
  - Two layers: “activation” (for initialization) and “visible” (tracking)
  - Tracking = update parameters of the kernel. Accuracy: 1 pix
  - Position and orientation are handled separately



# Tracking shapes specified by kernel masks

- Tracking using oriented Gaussian-like shape primitives



More natural shapes?

# Feature Detection & Tracking with the DAVIS

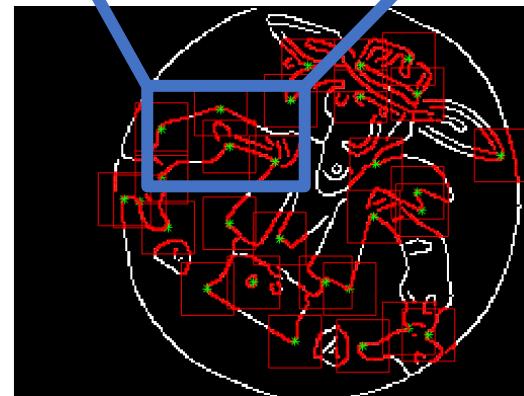
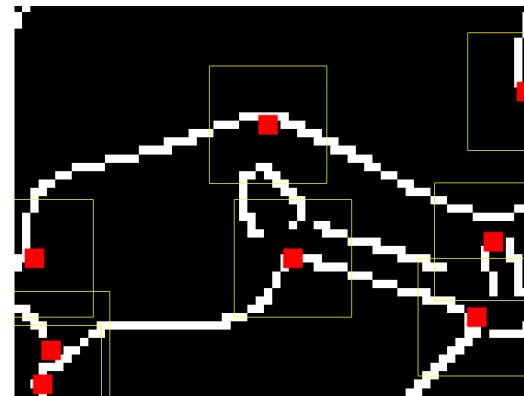
- Instead of predefined shapes... → arbitrary edge patterns
- Use frames from a DAVIS to build / **extract** such “shape model”



Corner  
detection



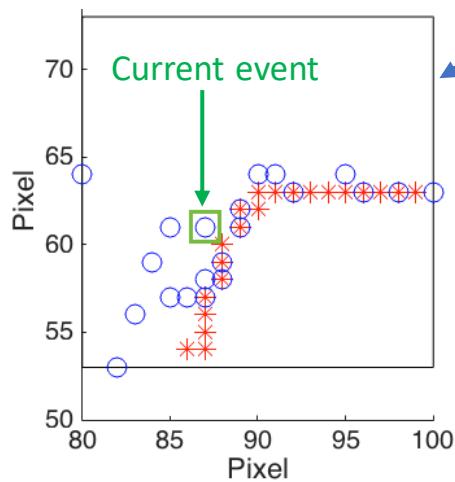
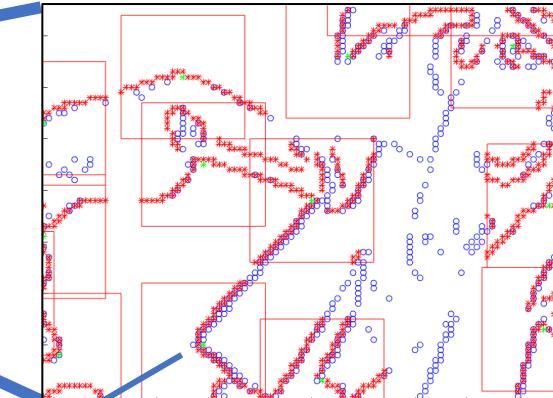
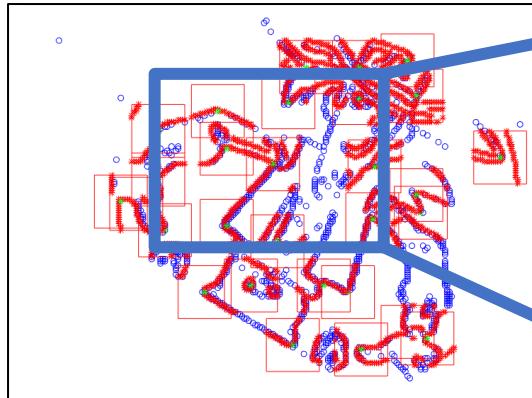
Edge detection  
around corners



# Feature Tracking using Events

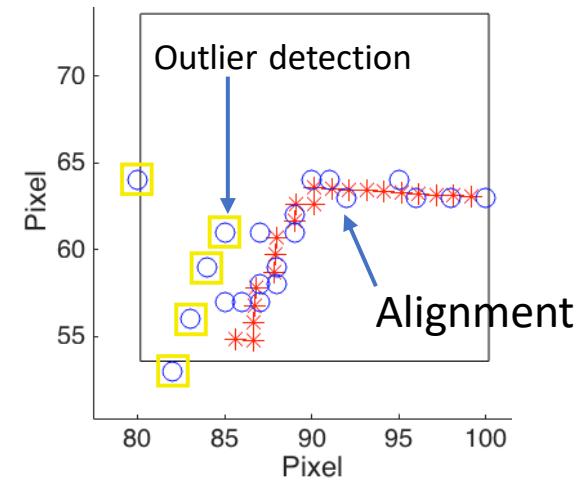
- After extracting edge patterns (“features”), track them using events

Shape models vs Events

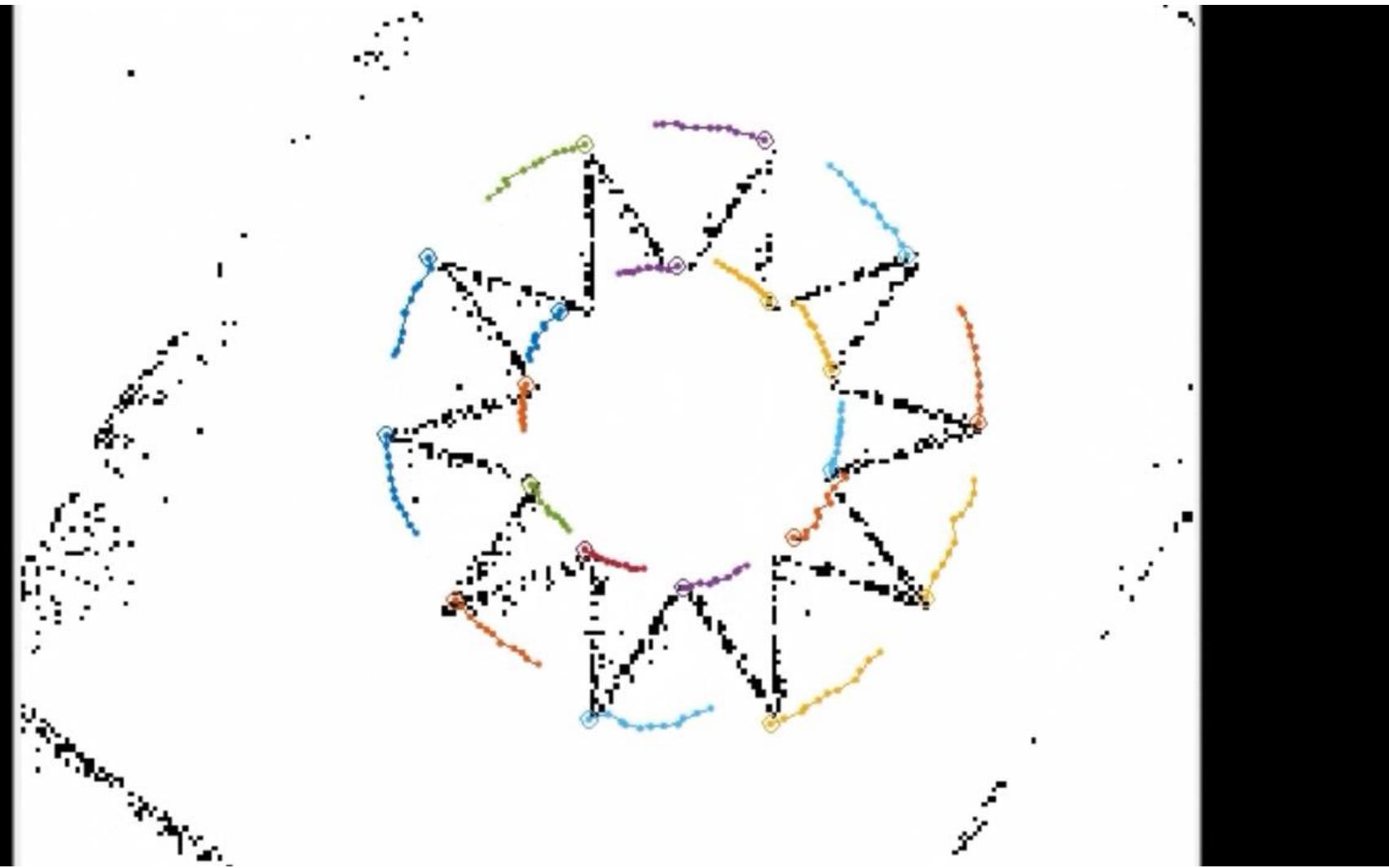


**Registration:**  
Iterative Closest  
Points (ICP)

$$\arg \min_{\mathbf{A}} \sum_{(\mathbf{p}_i, \mathbf{m}_i) \in \text{Matches}} \|\mathbf{A}(\mathbf{p}_i) - \mathbf{m}_i\|^2$$



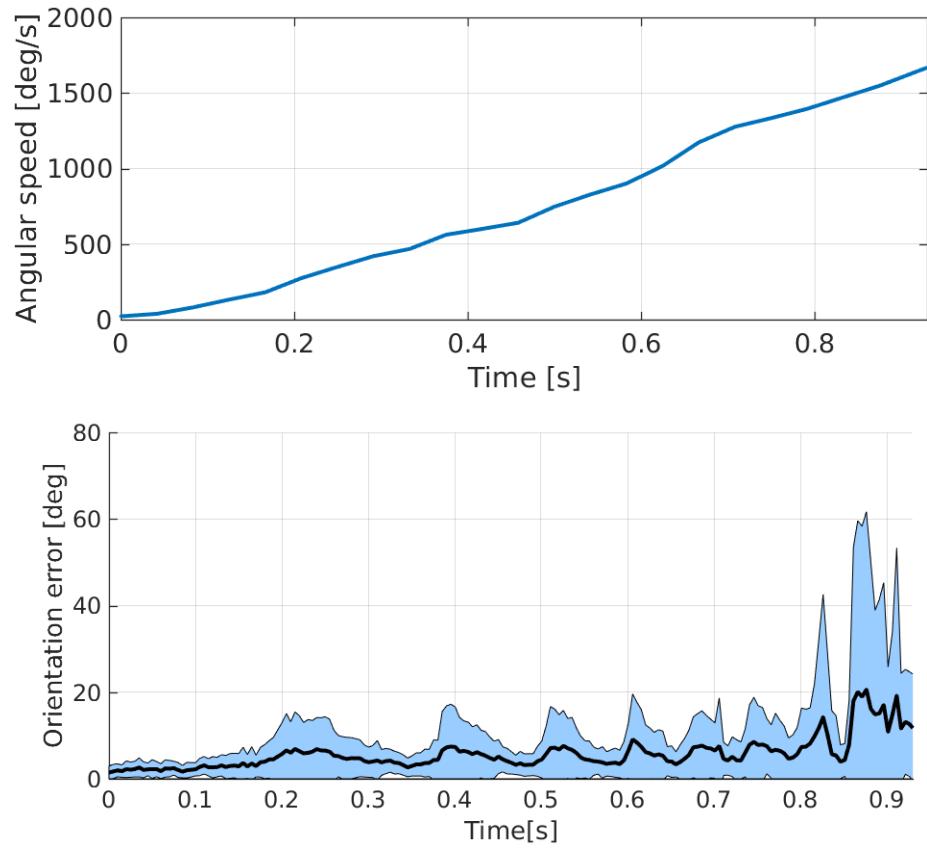
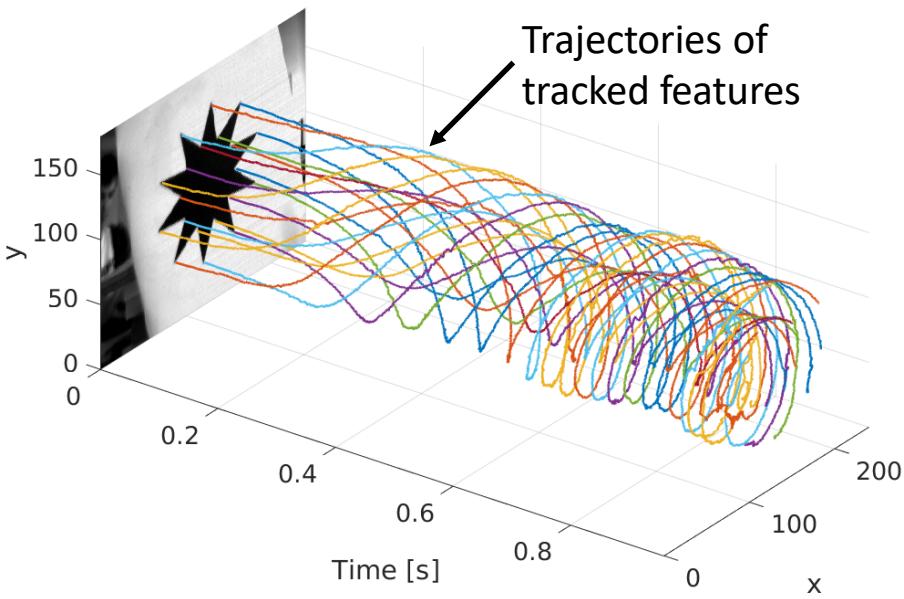
# High Speed Tracking. Even with just two edges!



# Feature Detection & Tracking with the DAVIS

## Results on Rotating Star

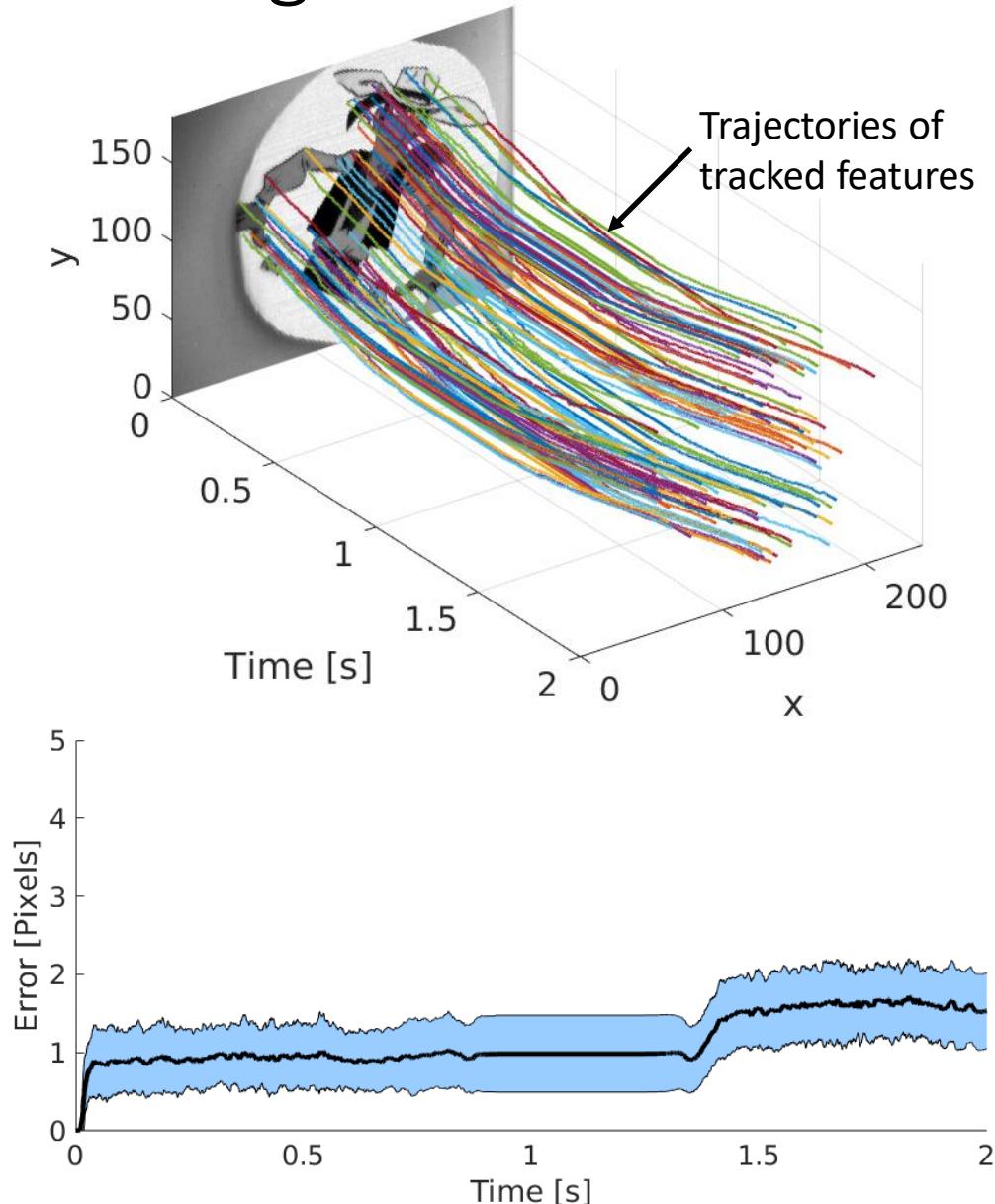
- Acceleration from rest to  $1.600^\circ/\text{s}$
- 1.800 pixels/s on image plane
- Mean error:  $6.3^\circ$



# Feature Detection & Tracking with the DAVIS

## Results on Lucky Luke

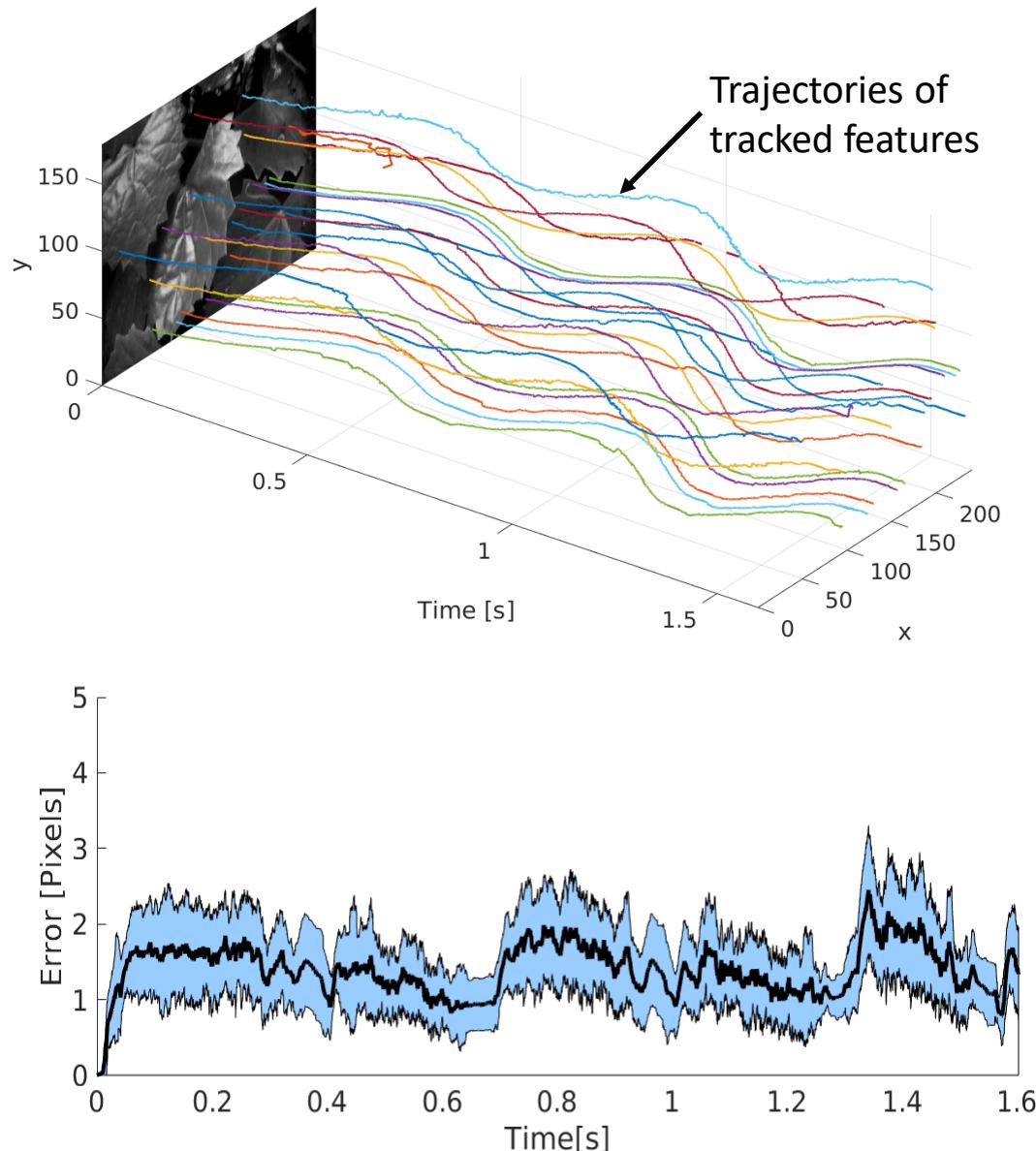
- Cartoon scene
- Back-and-forth translation
- 81 features
- Mean error: 1.22 pixels
- Ground truth obtained from frame-based feature tracking



# Feature Detection & Tracking with the DAVIS

## Results on Leaves

- Natural scene
- Oscillatory motion
- 20 features
- Mean error: 1.48 pixels
- Ground truth obtained from frame-based feature tracking



# Feature Detection & Tracking with the DAVIS

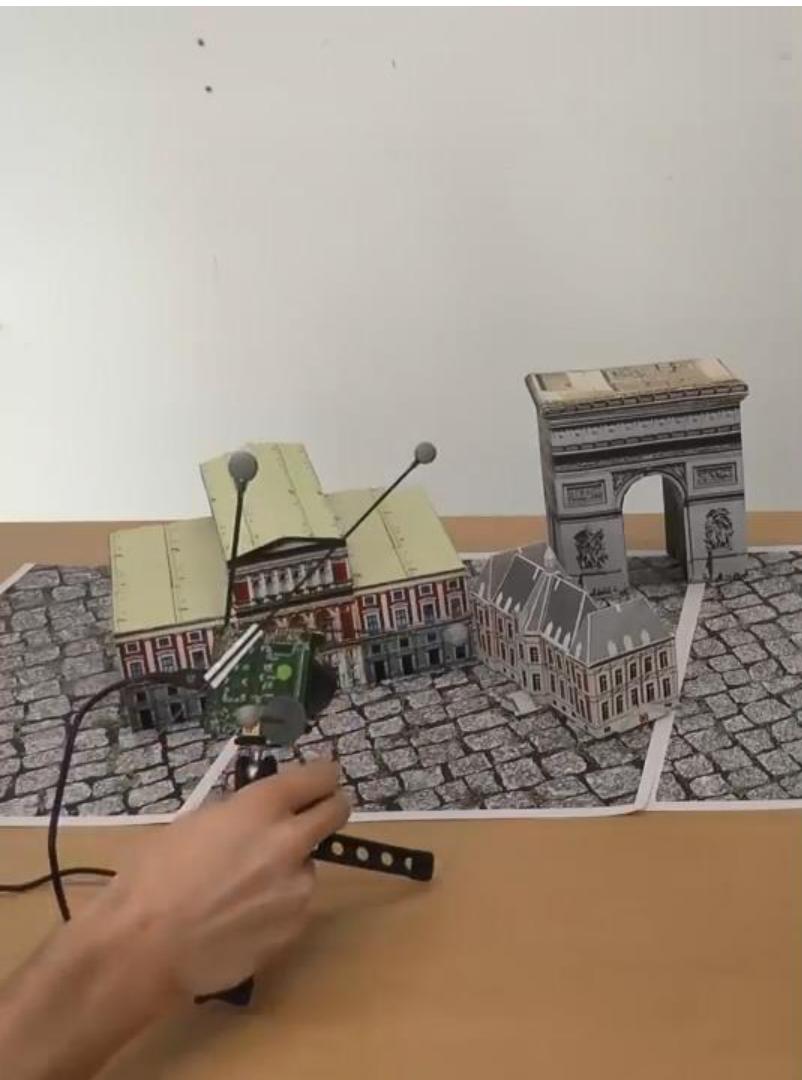
- **Pros:**

- Works on the **events directly** (no event frames) → Efficiency: only pixels with events are processed
- Allows for **high-speed** tracking ( $> 1 \text{ kHz}$ )
- Can track arbitrary edge patterns (**natural** scenes)
- Can track for considerable **duration** (2+ seconds)

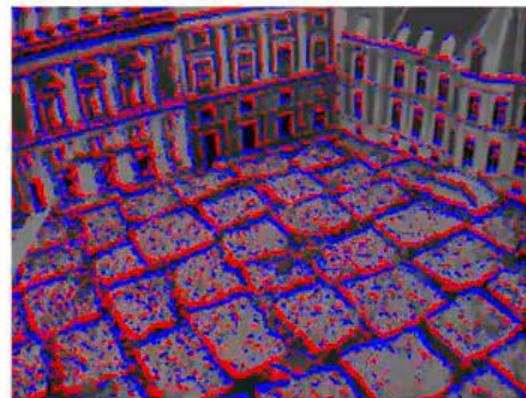
- **Cons:**

- **Jitter:** tracking is not as stable & accurate as with frames.  
Why? Only edge information is available (no absolute intensity).
- Depending on the **cost of update** per event, it can be inefficient.  
Alternative: process the last N events.
- Initialization based on **frames** (suffer from blur, low range) → Extract features from events or reconstructed intensity frames

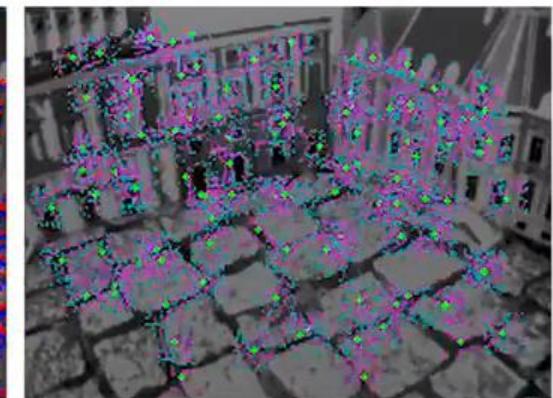
# Application to Visual Odometry



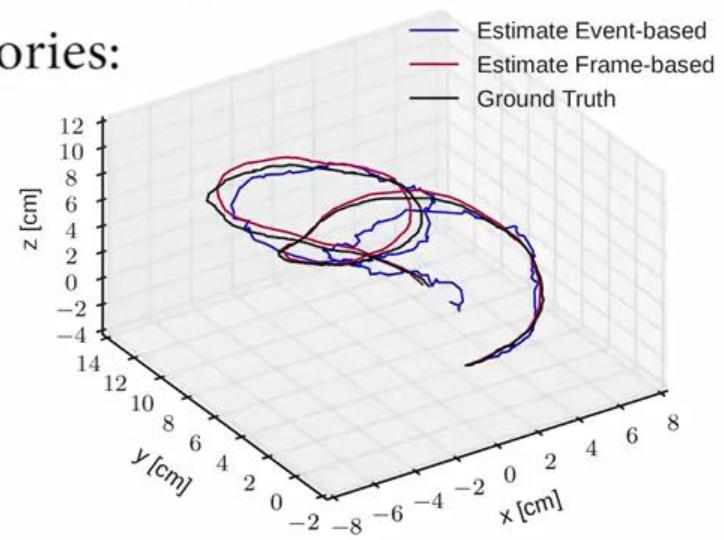
Raw Data:



Event-based Features:



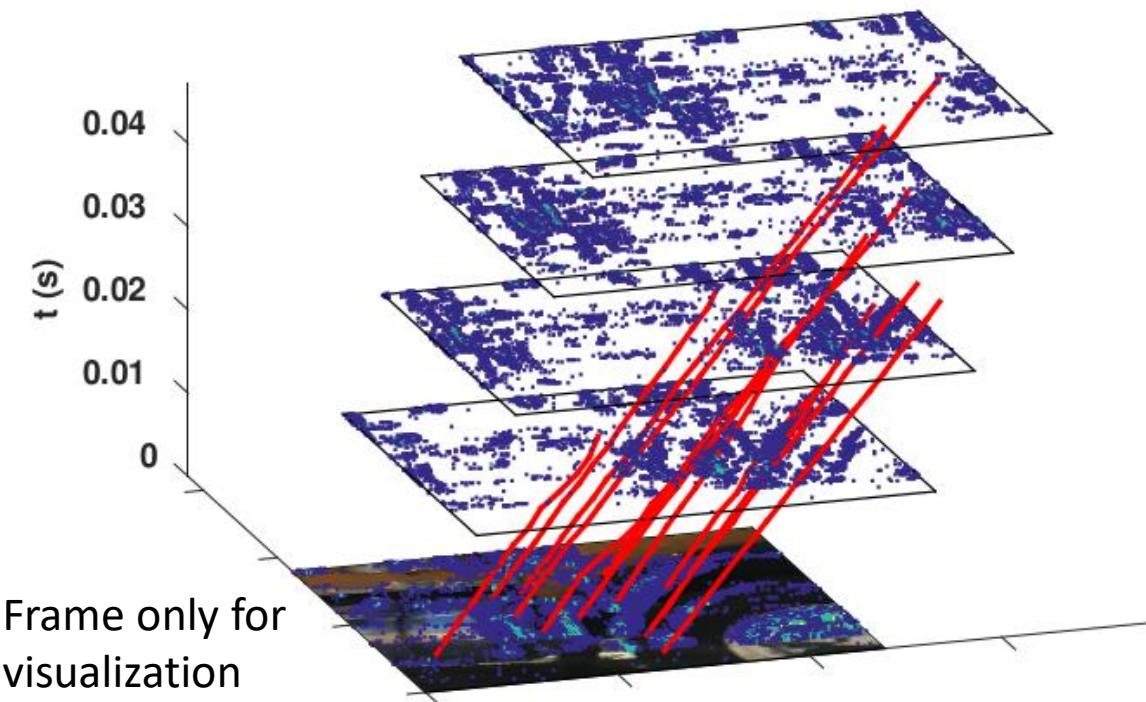
3D Trajectories:



Natural shapes from events?

# Feature Tracking using Motion-compensated Point Sets

- Natural **edge patterns** built using **events only**
- Use motion compensation to estimate **flow** and get a “sharp” feature
- Registration via Expectation-Maximization ICP on point sets, i.e., **soft data association**. Previous methods used “hard” associations

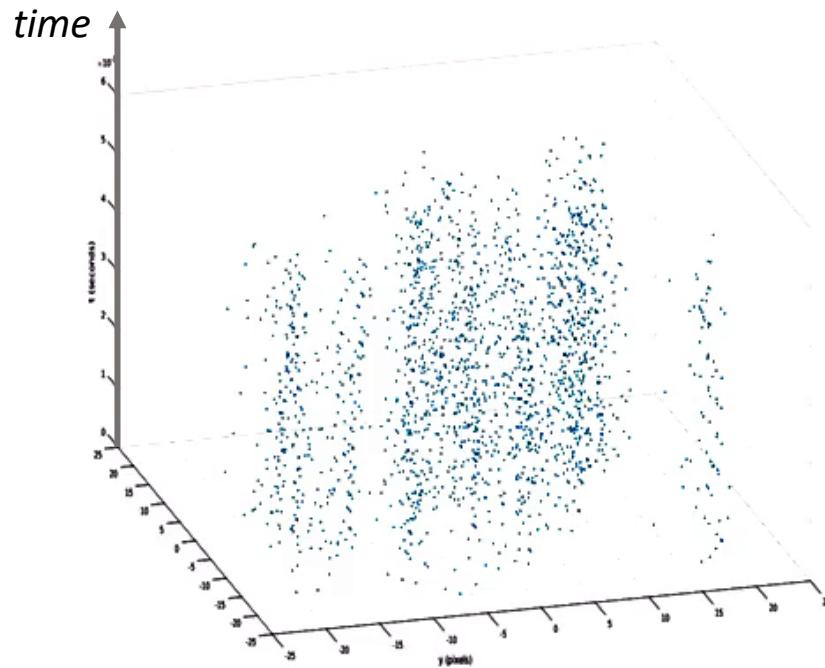


# Feature Tracking using Motion-compensated Point Sets

Main idea:

1. **Build a feature (“template”) by motion compensation of events, interpreted as point sets**

Space-time (polarity not used)



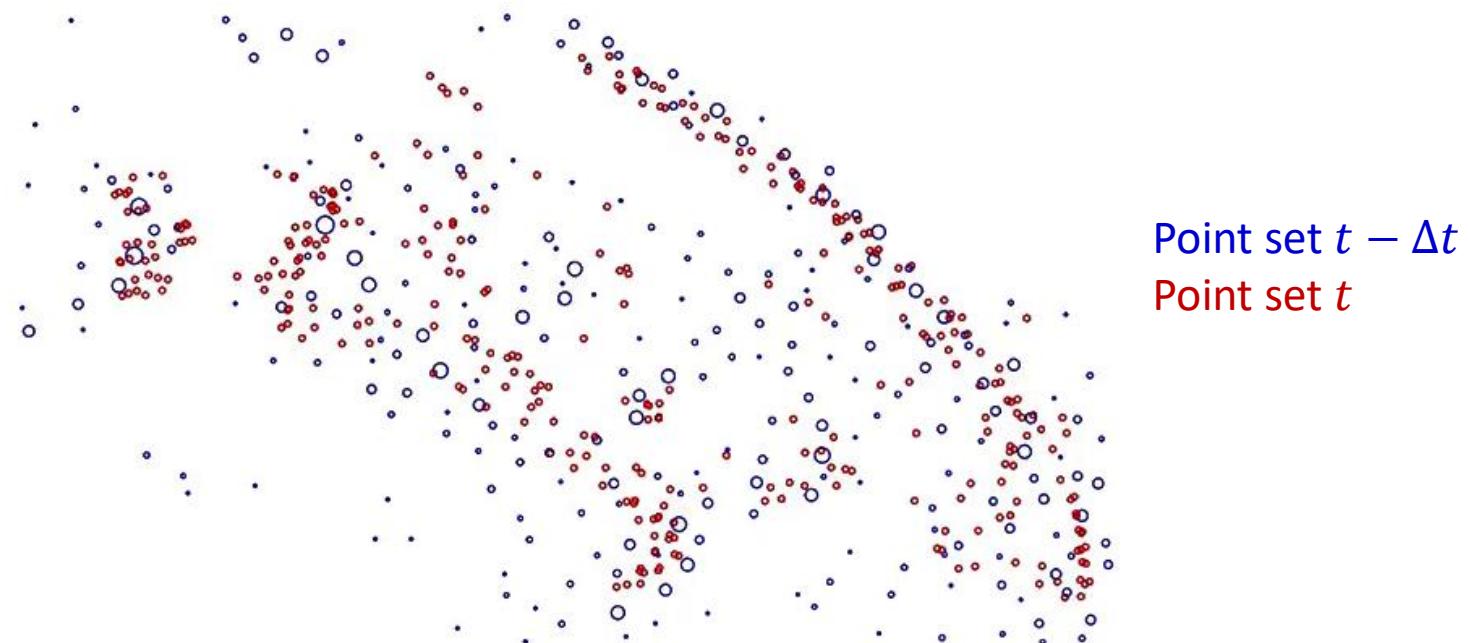
Motion compensated 2D point set  
(like a top-view of space-time)



# Feature Tracking using Motion-compensated Point Sets

Main idea:

1. **Build a feature** (“template”) by **motion compensation** of events, interpreted as point sets
2. **Register** feature w.r.t. previous one by EM-ICP (with affine warping)



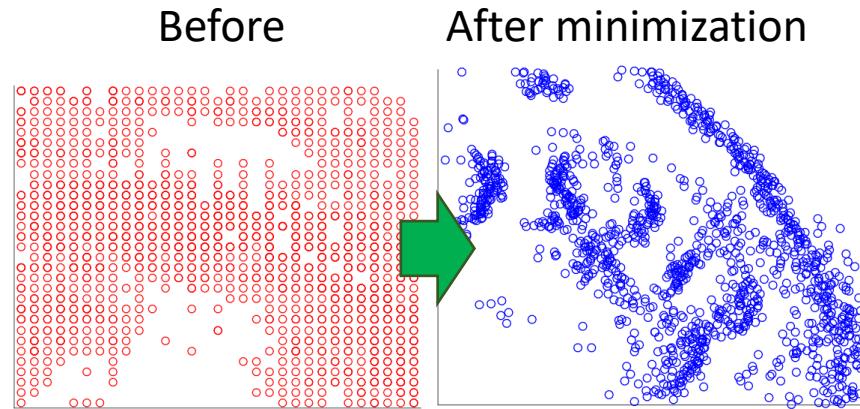
# Mathematical Details

- **Soft associations:** probability that event  $i$  originated in 3D point  $j$

$$r_{ij} := \mathbb{P}(\pi(i) = j)$$

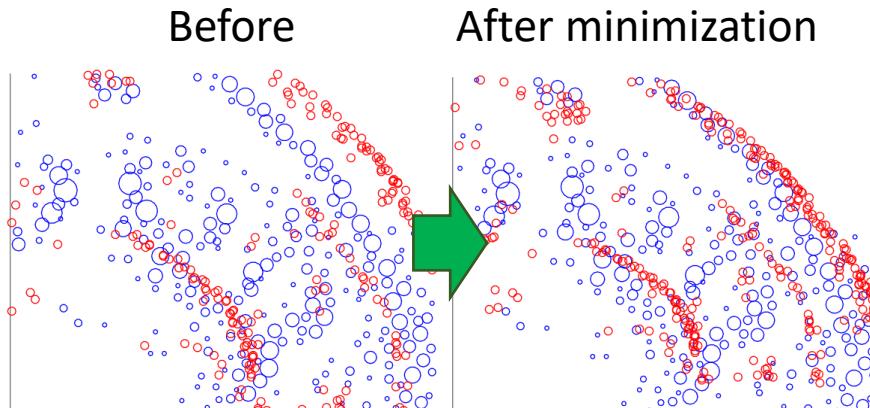
- EM Optical Flow Estimation

$$\min_v \sum_{i=1}^n \sum_{k=1}^n \left[ \sum_{j=1}^m r_{ij} r_{kj} \right] \|(x_i - t_i v) - (x_k - t_k v)\|^2$$



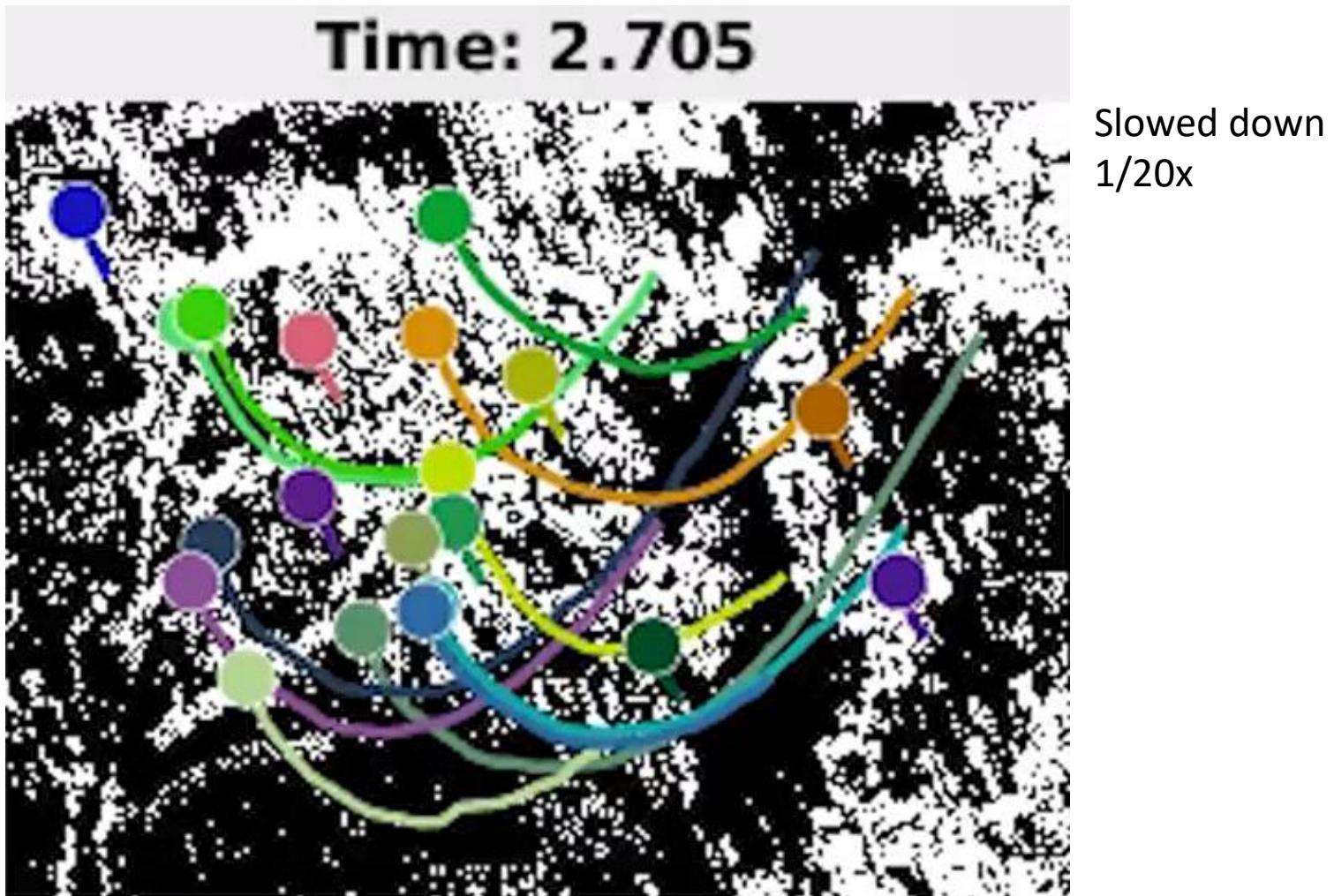
- Affine Feature Registration

$$\min_{A, b, r} \sum_{i=1}^n \sum_{j=1}^m r_{ij} \|A(x_i - t_i v) + b - p_j\|^2$$



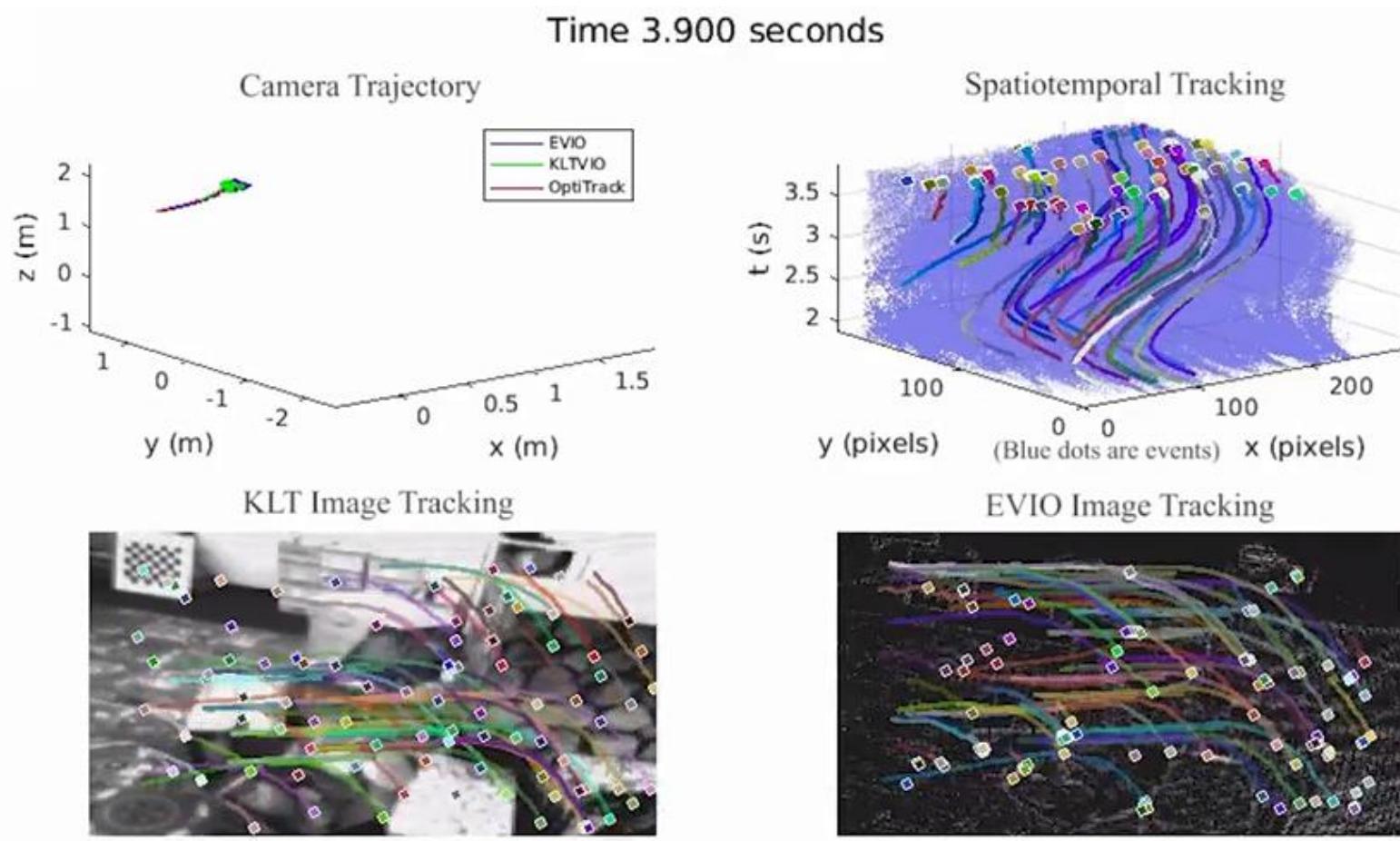
# Tracking fast motions

- In normal conditions, tracking accuracy  $\sim 1$  pix



# Application to Visual-Inertial Odometry

- **Feature tracks** and **IMU** data are fed to an **Extended Kalman Filter** that computes 6-DOF camera pose

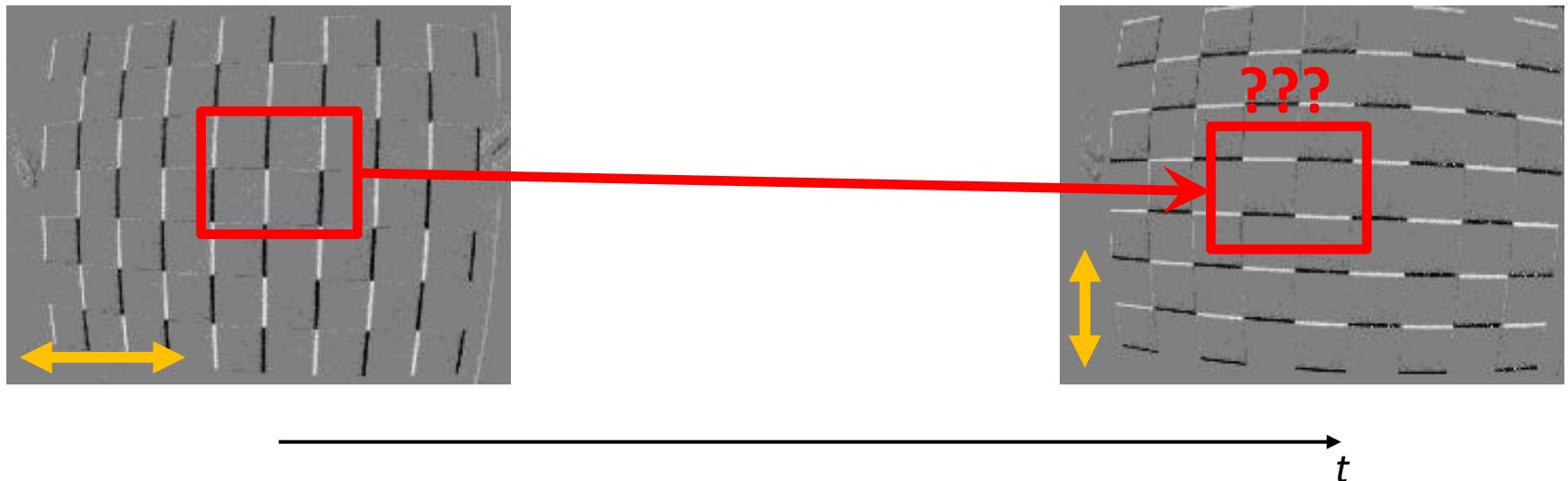


# More accurate?

(albeit more computationally expensive)

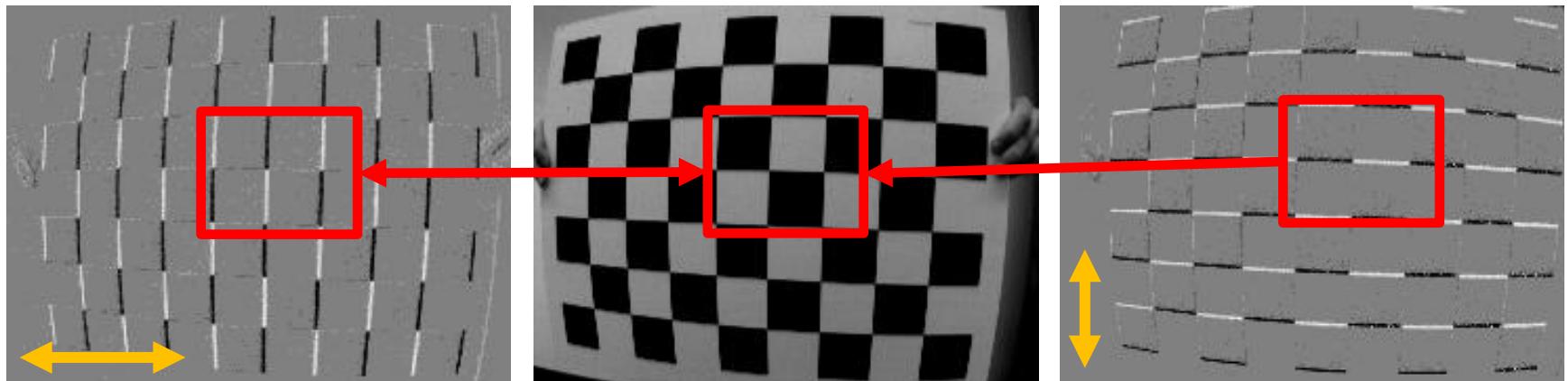
# The Problem of Data Association

- Remember one of the key ingredients?
- **Data-association** using events is difficult because:
  - Even if the model is invariant to **motion** (to some degree), events are not
  - The “**appearance**” of windows of events is **not preserved** across time
  - Event **noise** (specially in low light and for fast motions)
- What’s the **challenge**? To track well despite motion dependence



# The Problem of Data Association

- Remember one of the key ingredients?
- **Data-association** using events is difficult because:
  - Even if the model is invariant to **motion** (to some degree), events are not
  - The “**appearance**” of windows of events is **not preserved** across time
  - Event **noise** (specially in low light and for fast motions)
- What’s the **challenge**? To track well despite motion dependence
- A solution: match events via a more motion-invariant representation



Example of matching events across time (similar comment applies to event-to-model matching)

# Data Association – Solutions?

Event associated to a **model** (i.e., “template”) that is ...

- **User-defined** (star, kernels,...)
  - Papers: Ni NECO’15, Lagorce TNNLS’14
  - ✓ Pros: properly defined, can track well (shows potential)
  - Cons: limited applicability (not natural)
- Built from a **short time window of events**
  - Papers: Zhu ICRA’17, Rebecq BMVC 2017
  - ✓ Pros: it is not expensive and applies to natural scenes
  - Cons: the edge-like pattern depends on motion. Drift will arise
- More **motion-invariant**:
  - Built from a **frame** (Tedaldi EBCCSP’16, Kueng IROS’16, Gehrig IJCV’19)
  - Built from past events by **image reconstruction** (Gehrig IJCV’19)

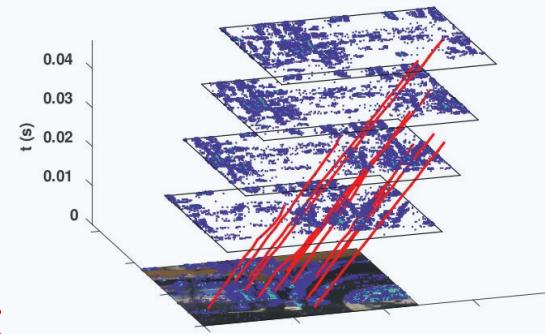
# Trackers we have studied

Zhu ICRA 2017

Event-Based Feature  
Tracking with Probabilistic  
Data Association

Events only

Changing feature  
appearance causes drift

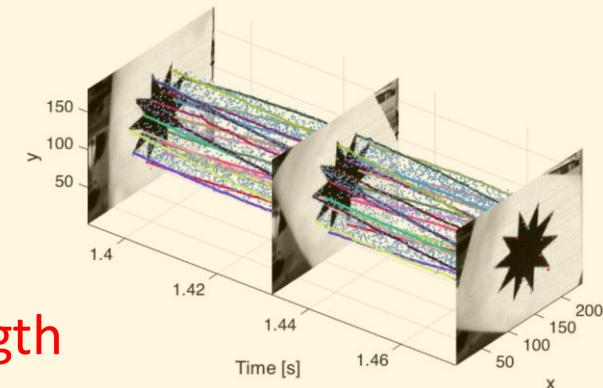


Tedaldi EBCCSP 2016  
Kueng IROS 2016

Low-Latency VO using  
Event-based Feature Tracks

Events & frames

No explicit edge strength

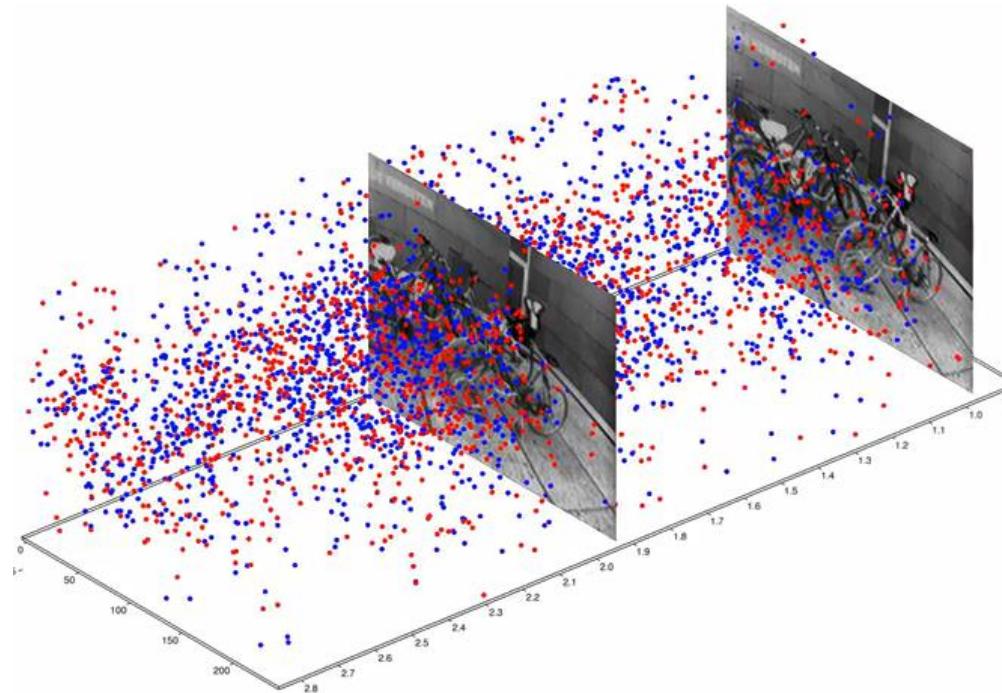


EKLT:

static appearance from frames  
considers edge strength

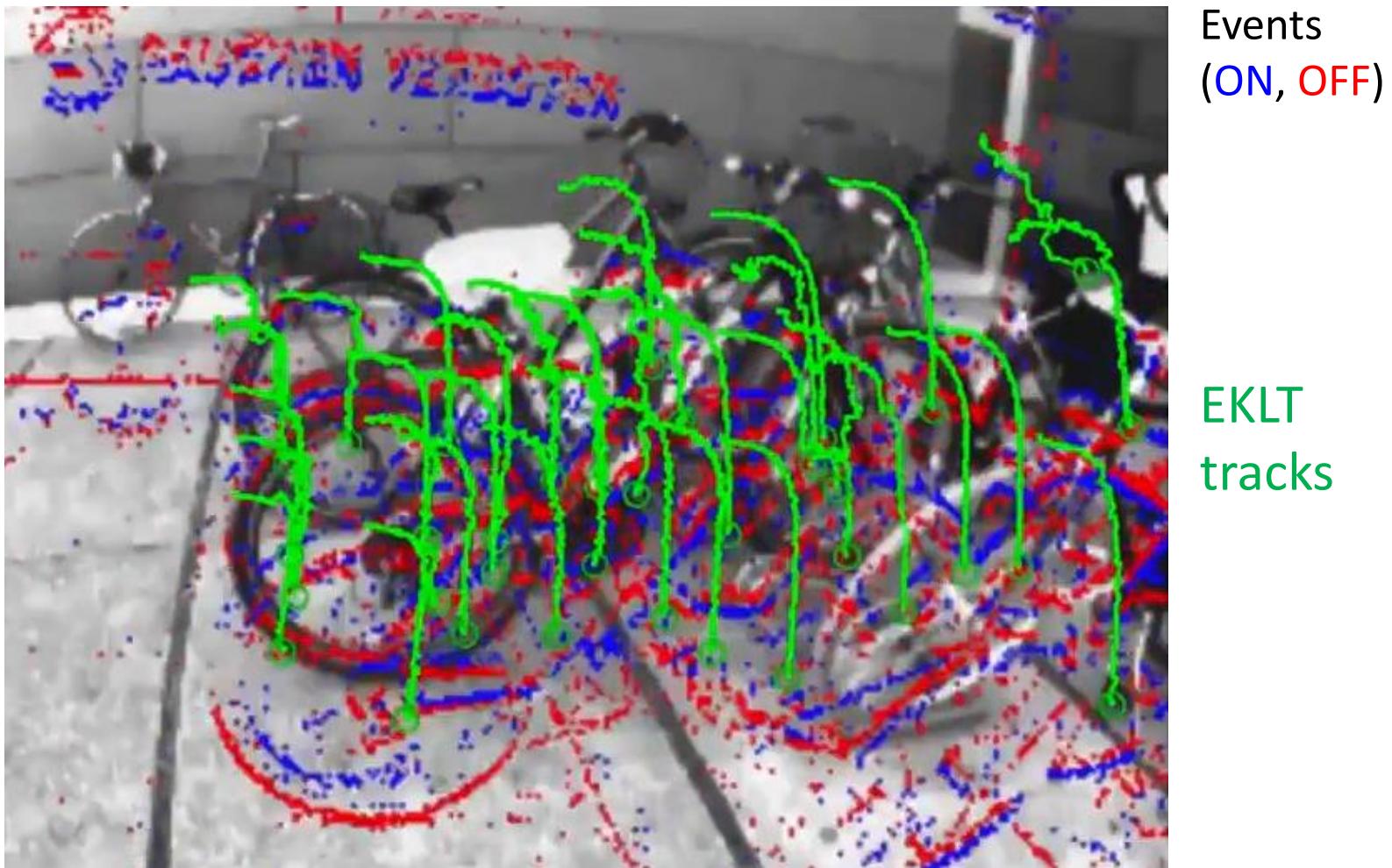
# EKLT: Event-based Kanade-Lucas-Tomasi Tracker

- **Goal:** Track features in the blind time between frames using events
- **Approach:** Extract features on frames and track them using events
  - **Asynchronous**, low-latency ( $\mu\text{s}$ ), tracking
  - **Probabilistic** (Max Likelihood) approach: compare events (**data**) to their prediction (**event generation model**) using frame gradients and optical flow
  - **Joint estimation** of feature warps and optical flow



# EKLT: Event-based Kanade-Lucas-Tomasi Tracker

- Track features with events and few frames (e.g., at 1Hz)



# EKLT Event-based Kanade-Lucas-Tomasi Tracker

**Brightness Increment**

$$\Delta L \approx \frac{\partial \hat{L}}{\partial t} \Delta t$$

Extract feature  
on frame

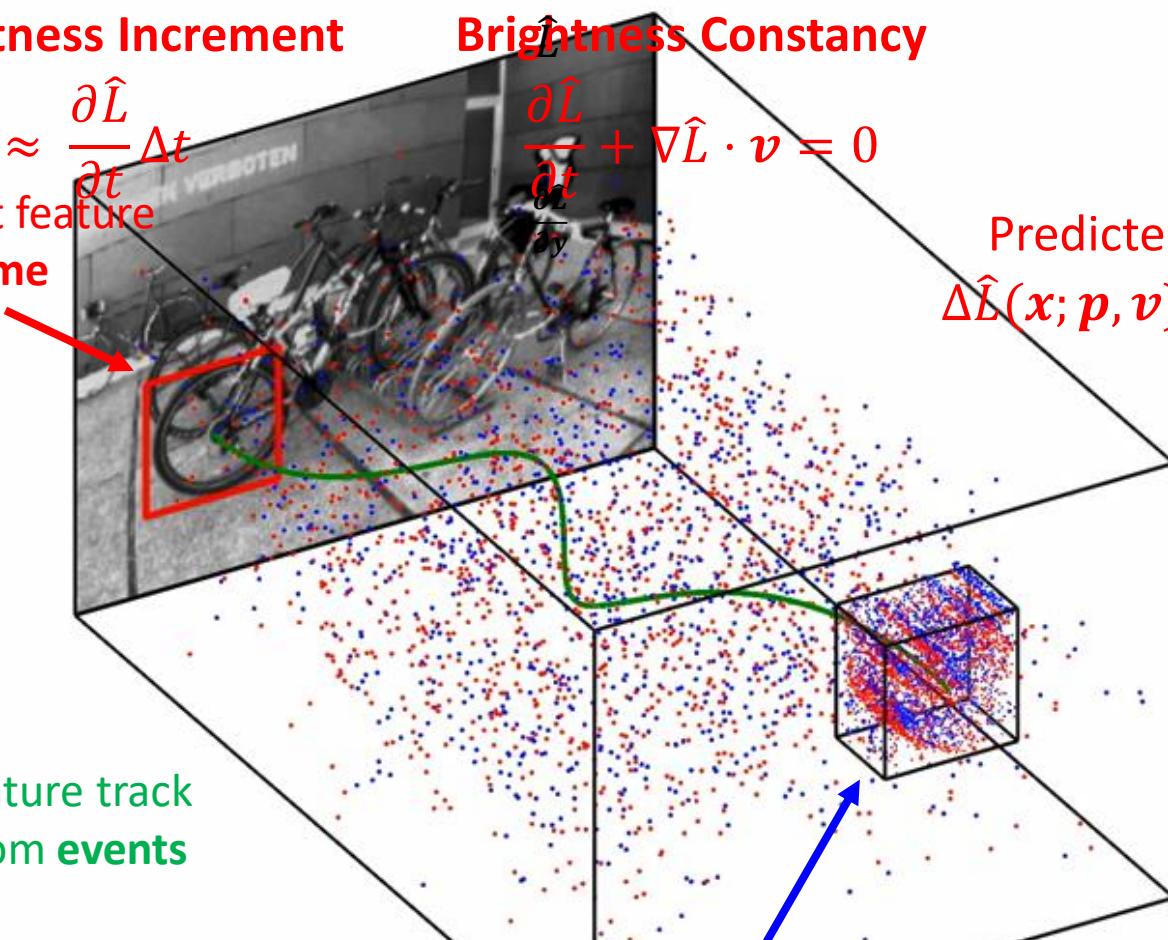
**Brightness Constancy**

$$\frac{\partial \hat{L}}{\partial t} + \nabla \hat{L} \cdot \mathbf{v} = 0$$

$$\Delta L \approx -\nabla \hat{L} \cdot \mathbf{v} \Delta t$$

Predicted Brightness Increment  
 $\hat{\Delta L}(x; p, v) = -\nabla \hat{L}(W(x; p)) \cdot v \Delta t$

Feature track  
from events



Events in spatio-  
temporal window

Measured Brightness Increment

$$\Delta L(x) = \sum_{k \in \Delta t} \pm_k C \delta(x - x_k)$$

# EKLT: Event-based KLT tracker



**Predicted Brightness Increment**

$$\Delta\hat{L}(x; p, v) = -\nabla\hat{L}(W(x; p)) \cdot v\Delta t$$

**Measured Brightness Increment**

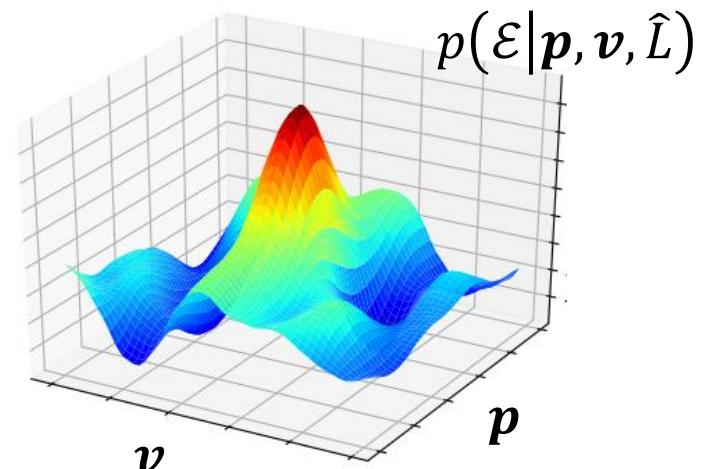
$$\Delta L(x) = \sum_{k \in \Delta t} \pm_k C \delta(x - x_k)$$

**Maximum Likelihood** approach:

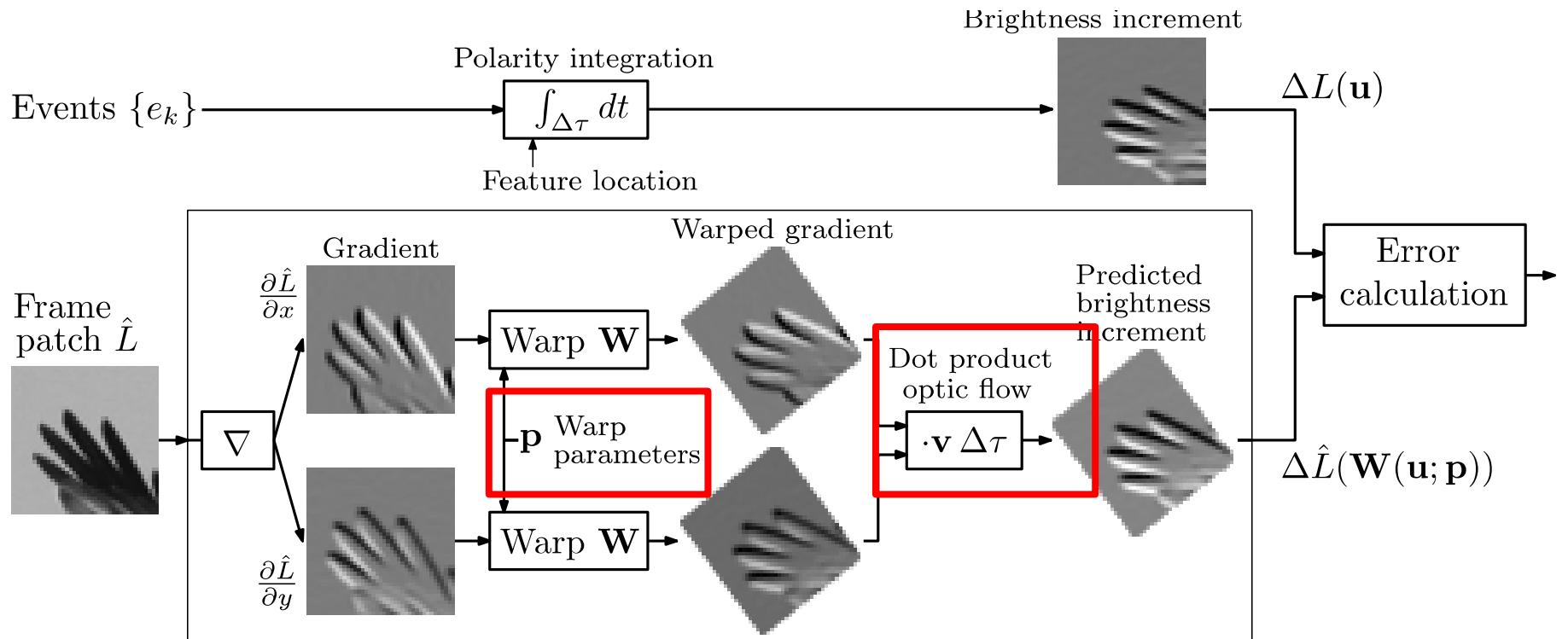
$$p(\mathcal{E} | \mathbf{p}, \mathbf{v}, \hat{L}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} \int_{\mathcal{D}} (\Delta L(x) - \Delta\hat{L}(x; \mathbf{p}, \mathbf{v}))^2 dx\right)$$

↑  
Events   ↑  
Intensity frame  
Optical flow  
Feature parameters (warp)

**Joint optimization**

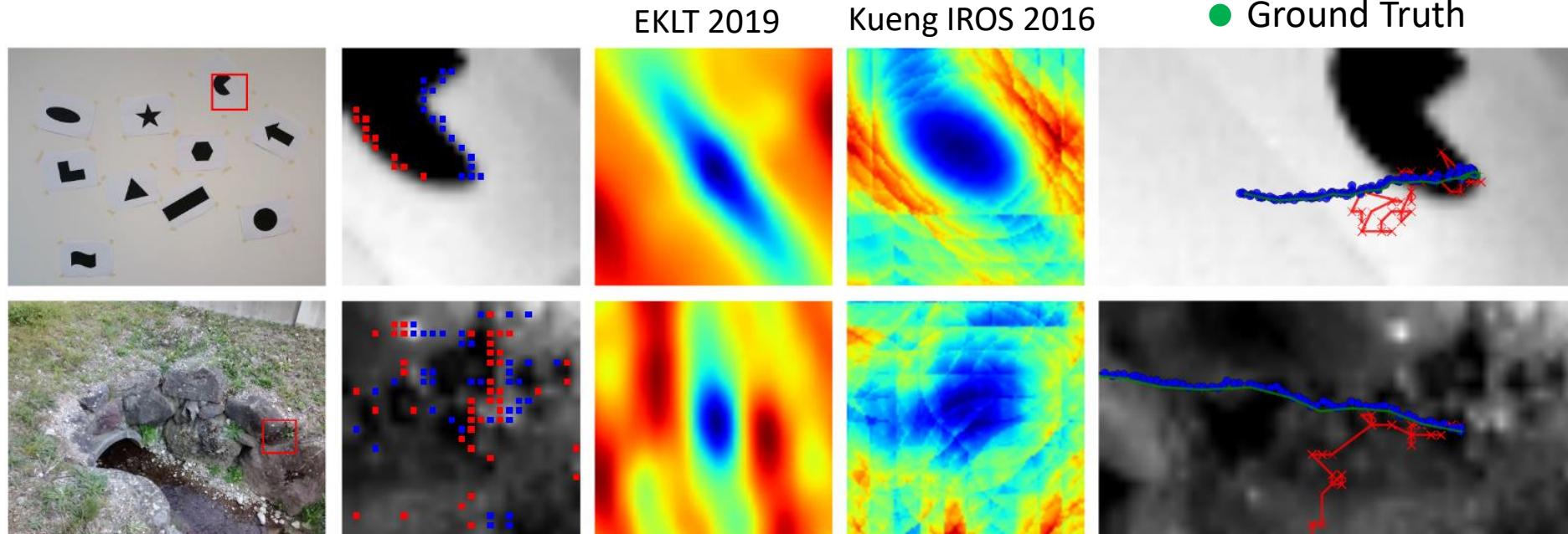


# Detail of Generation Model



# Tracking Comparison

- EKLT
- Kueng et al.
- Ground Truth

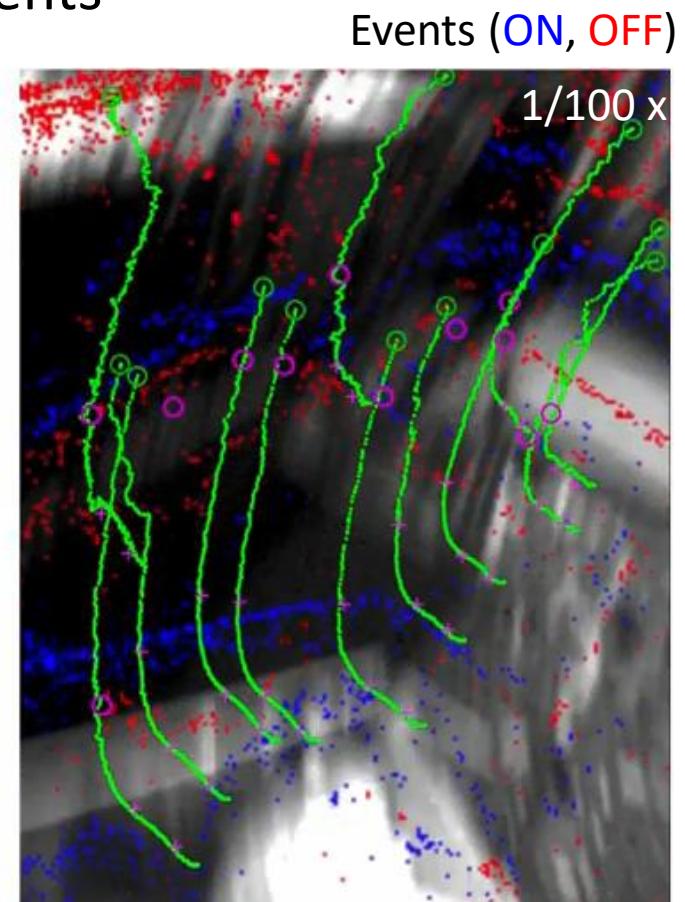
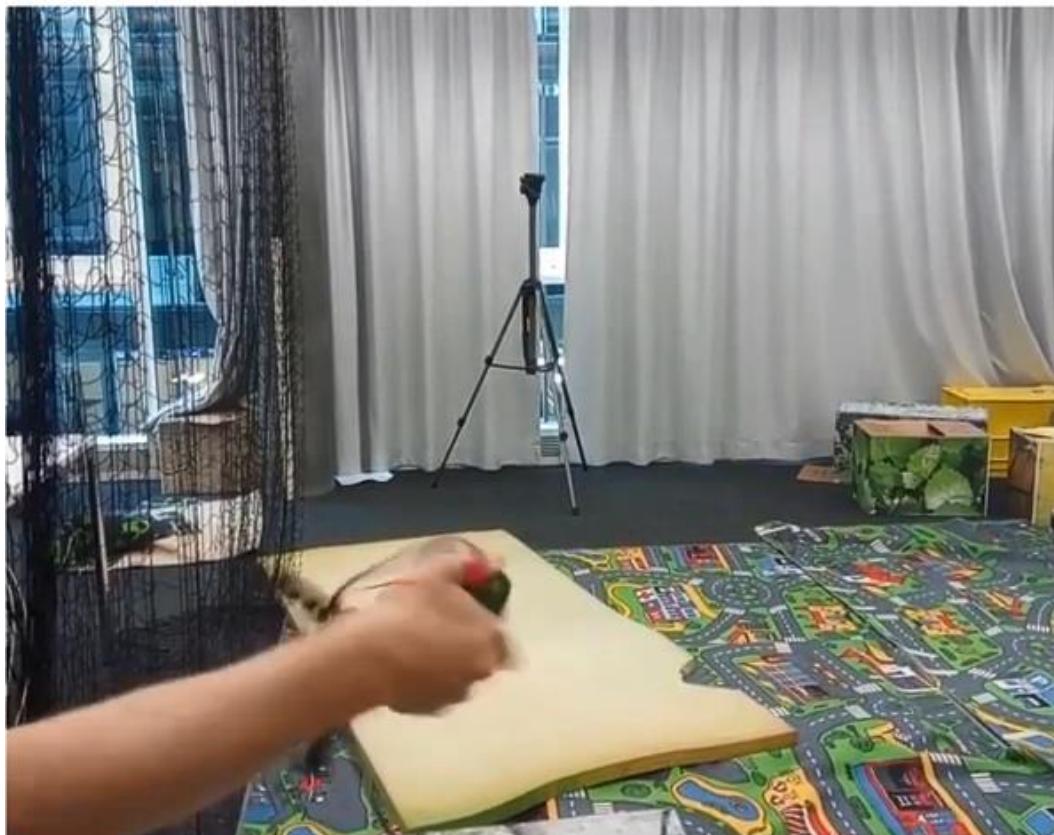


- ICP performs better on simpler scenes (sparse edges)
- The objective function in EKLT is better behaved → more robust in natural scenes

$$\left\| \frac{\Delta L(\mathbf{u})}{\|\Delta L(\mathbf{u})\|_{L^2(\mathcal{P})}} - \frac{\Delta \hat{L}(\mathbf{u}; \mathbf{p}, \mathbf{v})}{\|\Delta \hat{L}(\mathbf{u}; \mathbf{p}, \mathbf{v})\|_{L^2(\mathcal{P})}} \right\|_{L^2(\mathcal{P})}^2$$
$$\sum_{(\mathbf{p}_i, \mathbf{m}_i) \in \text{Matches}} b_i \| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{m}_i \|^2$$

# Tracking fast motions

- Tracking in the **blind-time** between frames, until leaving the FOV
- Frames blurred and with large displacements



KLT tracker vs. EKLT tracker

# Tracking in High Dynamic Range Scenes

- Features tracked in both bright and dark image (frame) regions

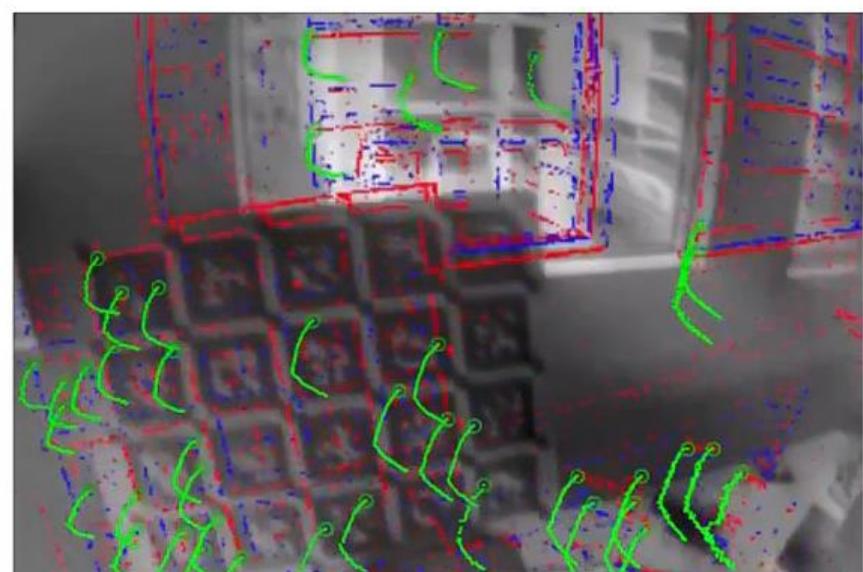
Overexposed  
frame



Underexposed  
frame



HDR Frame reconstructed  
from events (ON, OFF)

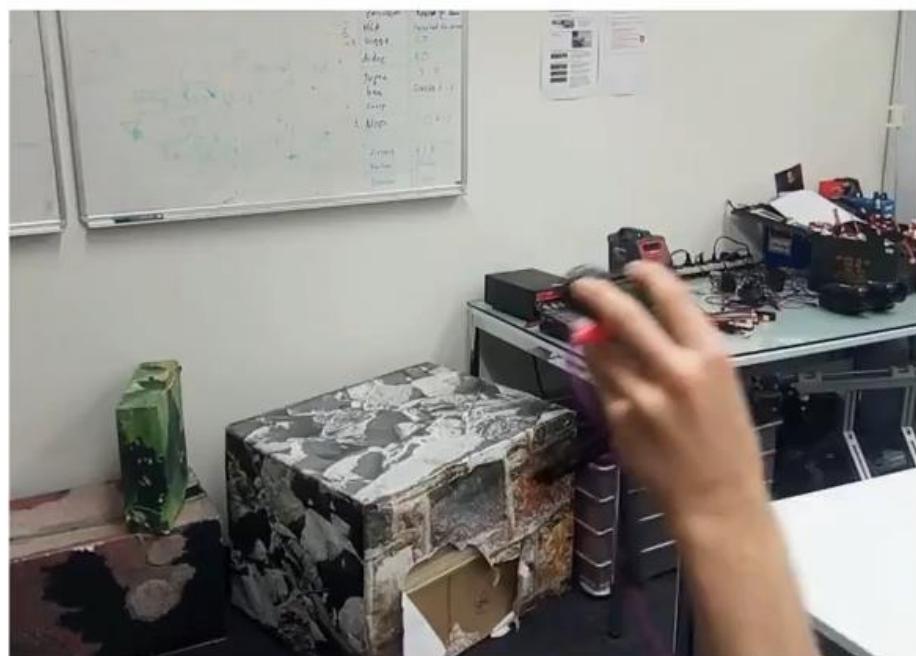


EKLT tracker

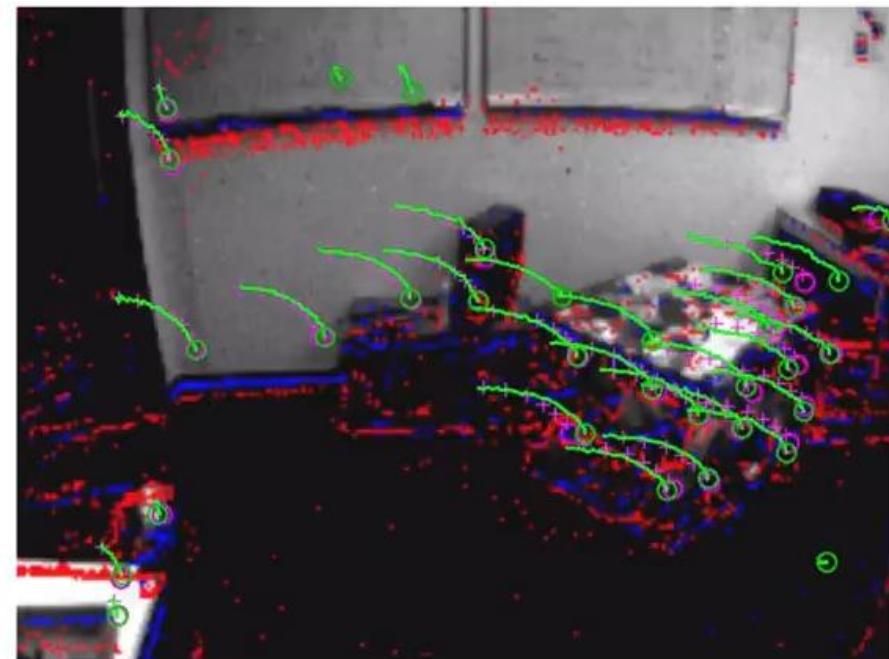
# Tracking in difficult illumination conditions

- Features tracked in low light and through abrupt illumination changes

Lights **ON** / **OFF**



Events (**ON**, **OFF**)



KLT tracker vs. EKLT tracker

# How do trackers compare?

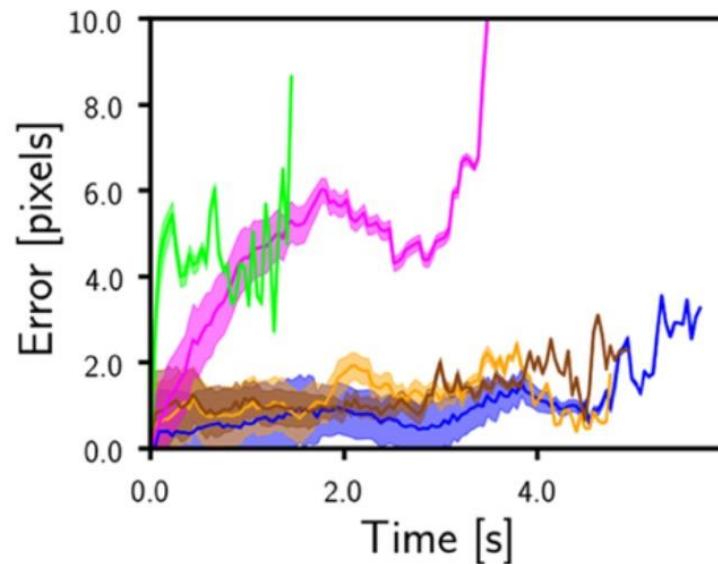
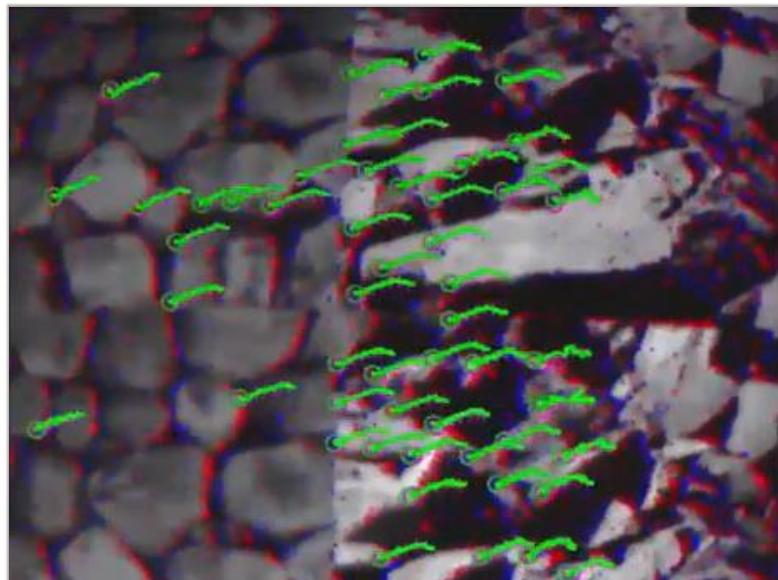
# Comparison of five trackers

- **ICP** (Kueng et al. IROS 2016, Tedaldi EBCCSP 2016)
  - Iterative Closest Point algorithm on two point sets
- **EM-ICP** (Zhu et al. ICRA 2017)
  - Tracking on motion-compensated points sets
- **KLT-MCEF** (Rebecq et al. BMVC 2017)
  - Classic KLT tracker on motion-compensated images
- **KLT-HF**
  - Classic KLT tracker on intensity images reconstructed by ACCV 2018 filter
- **EKLT** (Gehrig et al. IJCV 2019)
  - Joint tracking and flow estimation using event generation model

# Comparison of multiple (five) trackers

Performance metrics: **accuracy** and **feature age**

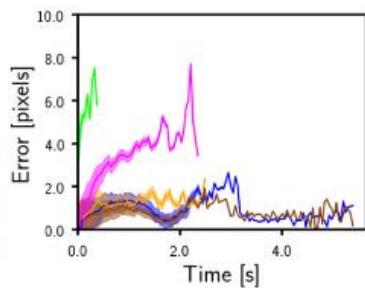
- How accurate is the tracker?
- For how long does it track?



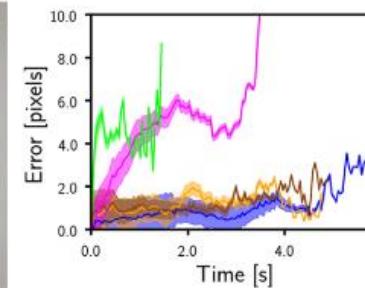
Trackers:    — ICP    — EM-ICP    — KLT-MCEF    — KLT-HF    — EKLT

# Comparison of multiple (five) trackers

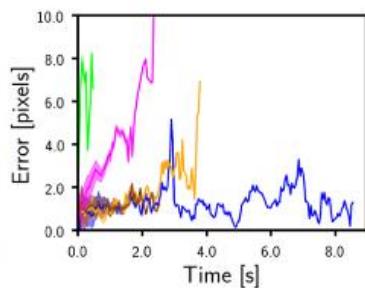
Trackers:  ICP  EM-ICP  KLT-MCEF  KLT-HF  EKLT



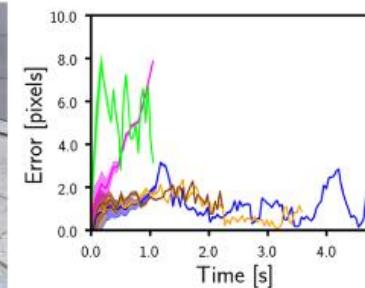
(c) boxes\_6dof



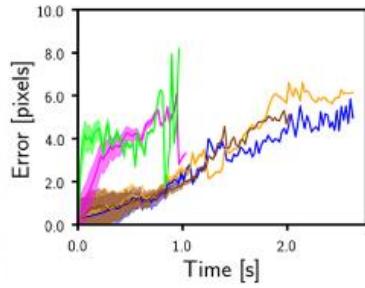
(d) poster\_6dof



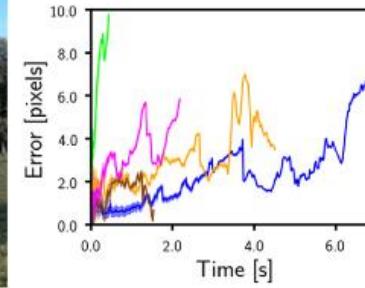
(e) pipe\_2



(f) bicycles



(g) outdoor\_day1



(h) outdoor\_forward

# Comparison of multiple (five) trackers

- Accuracy and feature age on 8 scenes and 5 trackers

**Table 2** Comparison of five different feature tracking methods on eight test sequences from real data. Average of the tracking error, normalized by the length of the tracks, for each combination of method and sequence

Scene	Datasets	Track-normalized error (px)				
		Ours	ICP	EM-ICP	KLT-MCEF	KLT-HF
Black and white	shapes_6dof	<b>0.80</b>	1.49	2.31	0.94	2.43
	checkerboard	<b>1.21</b>	1.92	2.30	2.30	1.75
High texture	poster_6dof	<b>0.64</b>	2.48	3.10	0.97	1.18
	boxes_6dof	<b>0.72</b>	4.59	1.60	0.80	1.24
Natural	bicycles	<b>0.76</b>	4.22	1.50	1.26	1.21
	pipe_2	<b>0.78</b>	4.90	1.63	1.04	1.06
	outdoor_day1	<b>0.71</b>	2.96	2.30	2.00	2.52
	outdoor_forward5	<b>0.80</b>	4.15	1.47	1.58	2.36

The best result per row is highlighted in bold

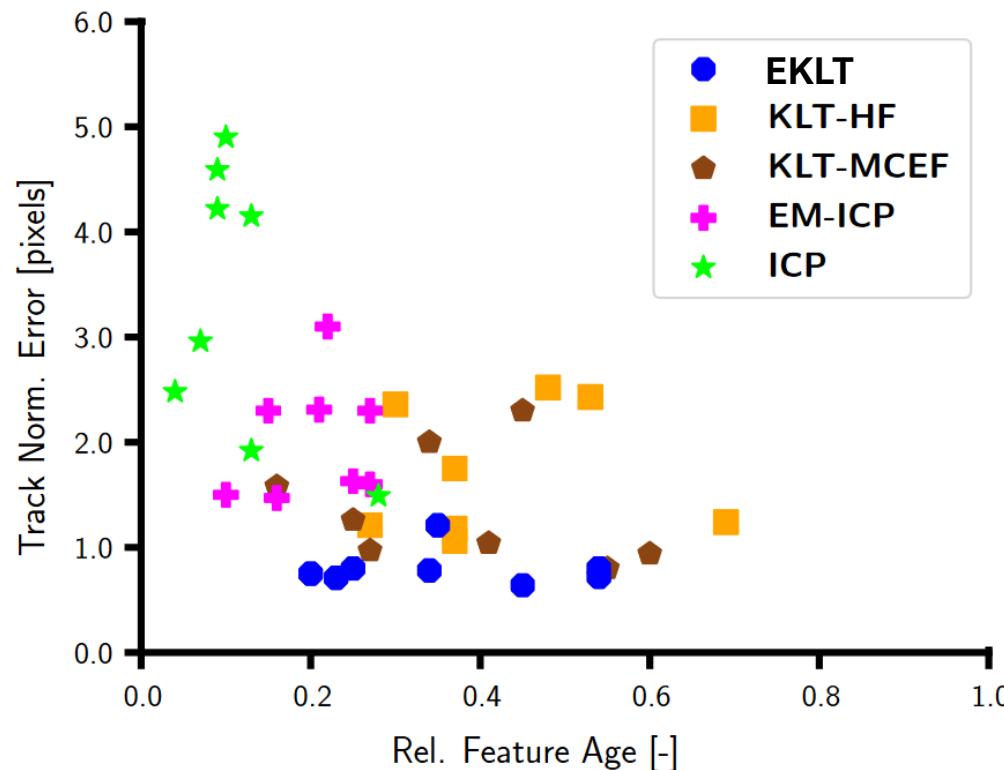
**Table 3** Comparison of five different feature tracking methods on eight test sequences from real data. Relative feature age, normalized by the length of KLT tracks, for each combination of method and sequence

Scene	Datasets	Relative feature age				
		Ours	ICP	EM-ICP	KLT-MCEF	KLT-HF
Black and white	shapes_6dof	0.54	0.28	0.21	<b>0.60</b>	0.53
	checkerboard	0.35	0.13	0.27	<b>0.45</b>	0.37
High texture	poster_6dof	<b>0.45</b>	0.04	0.22	0.27	0.37
	boxes_6dof	0.54	0.09	0.27	0.55	<b>0.69</b>
Natural	bicycles	0.20	0.09	0.10	0.25	<b>0.27</b>
	pipe_2	0.34	0.10	0.25	<b>0.41</b>	0.37
	outdoor_day1	0.23	0.07	0.15	0.34	<b>0.48</b>
	outdoor_forward5	0.25	0.13	0.16	0.16	<b>0.30</b>

The best result per row is highlighted in bold

# Comparison of multiple (five) trackers

- Try to show two performance metrics simultaneously



- Missing: computational performance  
EKLT (python): process 8000 events/s, “real-time” ~200k ev/s

# Discussion

## Pros:

- **Low latency** (>100 updates per second per feature)
- Sub-pixel precision and tracking for several seconds
- Managed to achieve subpixel accuracy on many sequences
- Preliminary benchmark (five trackers, eight diverse sequences)

## Cons:

- There is a **lack of established benchmark** to advance the state of the art. Getting ground truth at high speed and HDR is difficult.
- **Performance highly depends** on the scene, texture, motion, etc.
- Some trackers are still expensive.  
Research does not always optimize for speed.  
Accuracy – Computational performance trade-off not quantified yet

# References

- Section 4.1 of [Event-based Vision: A Survey](#), arXiv 2019
- Papers referenced at the bottom of each slide.
- Classical computer vision books for the topic of feature detection and tracking.

# Feature Tracking

## Discussion

# Questions

- What can be tracked?
  - Brightness changes → Need temporal **contrast** (DVS has a contrast sensitivity of ~ 15%)
  - What causes brightness changes?
    - Blinking lights (LEDs) - seldom considered.
    - **Moving edges**, found in textures, silhouettes, etc.  
Regions with low contrast or texture are challenging to track.
- What's the scenario?
  - Static or moving camera?
  - How many objects are there in the scene?
  - How big are the objects on the image plane?
- How to express what we want to track in terms of events?
  - Need a “template” or “representation” (**Detection**)

# Questions

- How to track?
  - Can we track with just one event?
  - Better if multiple events (or additional knowledge) to build a **representation** to track.
- Exploiting high temporal resolution and spatial vicinity: (nearest neighbor) **data association**.
- How to handle **nuisances**, such as perspective distortion, noise, etc.?
  - We would like the tracker to have some “**invariance**” and “**robustness**”.
- Tracking may be posed as a **registration** problem
  - and solved via ICP, gradient descent, etc.

# Tracking Methods

- Blob / Cluster tracking
  - A filter updates the parameters of the “blob” distribution
  - Example: vehicles or people modeled as “blobs”
- Shape tracking (evolving point shapes on image plane 2D)
  - Simple shapes (squares, triangles, stars)
  - More complex shapes
- Tracking posed as a registration problem
  - What is registration?
    - Matching events to a model
    - What is being optimized?
    - What type of distortions can be handled?
  - Iterative Closest Point (ICP)
  - Point set registration by gradient descent
  - Extension of the KLT tracker to events: EKLT

# More Questions

- How are **outliers** handled by ICP?
- How to handle events due to **occlusions** or disocclusions?
- Is **polarity** used in the different methods?
  - Typically not. Only in EKLT (in the event generation model).
  - Example: an Off event may be caused by a dark-to-bright (DL) edge moving right or by a bright-to-dark (LD) edge moving left, thus polarity can be confusing; need to be able to **disambiguate**.
- Classical computer vision:
  - What is Canny edge detection?
  - What are Harris corners?
  - What are the colors in the plot? (Tedaldi et al. EBCCSP 2016)

# Evolution Analysis

- We can identify several **axes of progression** (evolution):
  - Increasing **complexity of shapes** to be tracked: from hand-crafted to natural shapes, obtained from the data itself.
  - **Techniques** used:
    - **Data association**: from hard to soft (explicit or implicit).
    - **Registration / alignment methods**: from point-based methods (shapes represented as 2D point sets) to grid-based ones (shapes represented by histograms of events, to account for edge strength, using lattice-based implicit data association and reutilizing registration techniques from images): more **accurate and longer tracks** but higher **computational cost**.
- Improve this progression by reading **Section 4.1** of [Event-based Vision: A Survey](#), TPAMI 2020

# Event-based Robot Vision

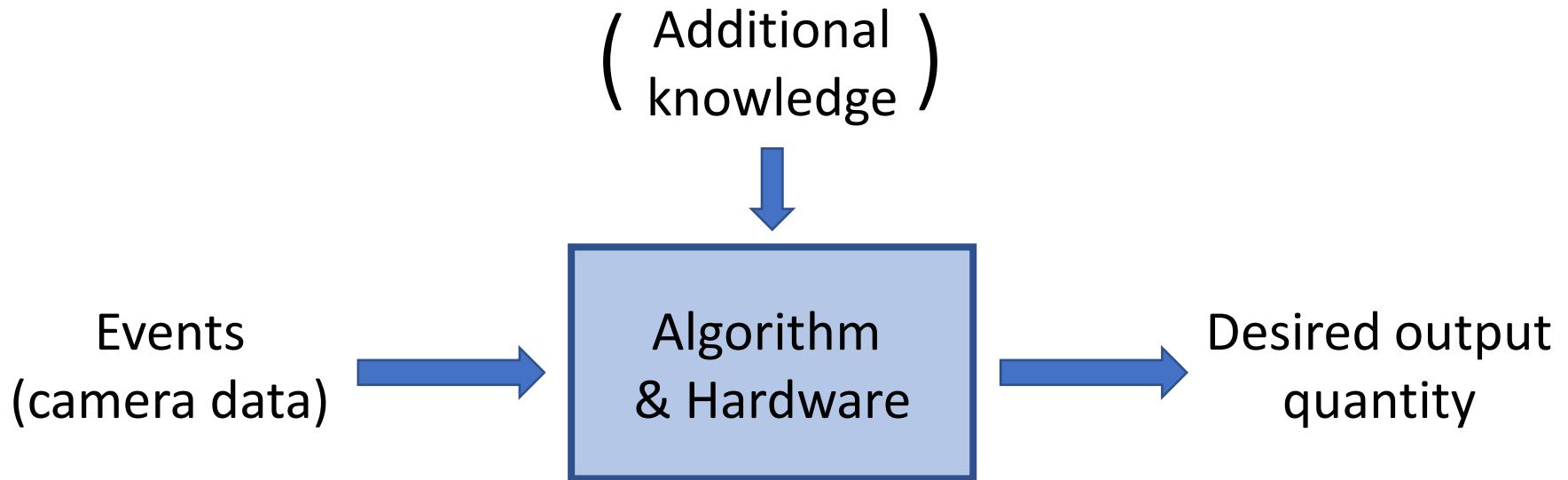
Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

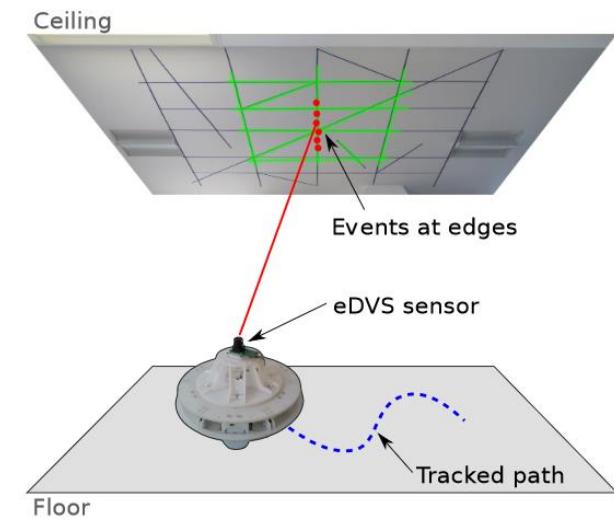
# How to Process the Events?

# Overview



# The New Paradigm

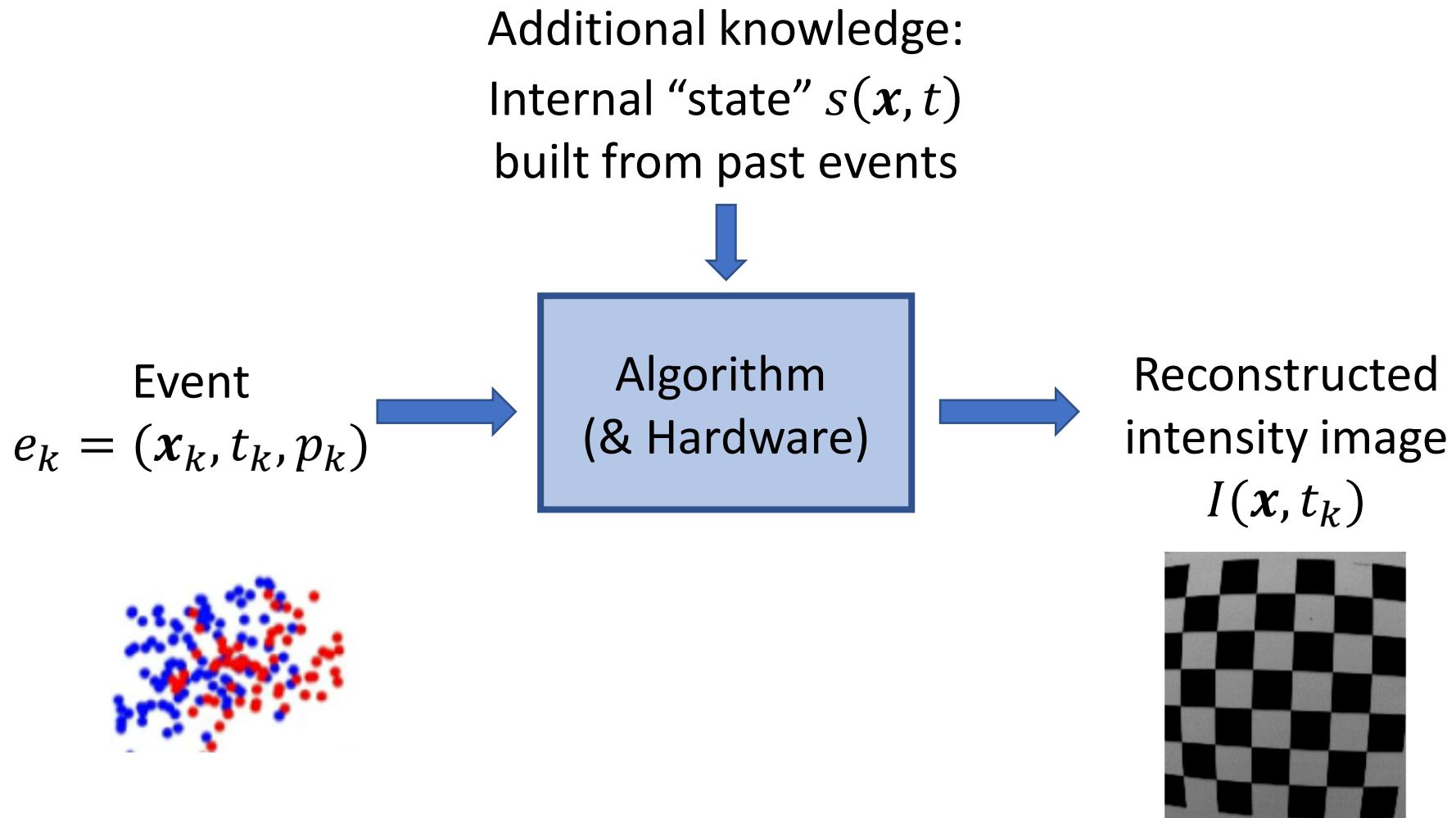
- Process **one event at a time**
  - How much information does an event carry?
  - The vision system / algorithm needs to have **additional knowledge** (either from external information or built from past events) to assimilate (fuse with) each event.
- 
- **Advantages:**
    - **Minimum latency**
    - Asynchronous & sparse
  - **Disadvantages:**
    - Updating the system on a per-event basis may be expensive, depending on the task



Weikersdorfer et al. ICVS 2013

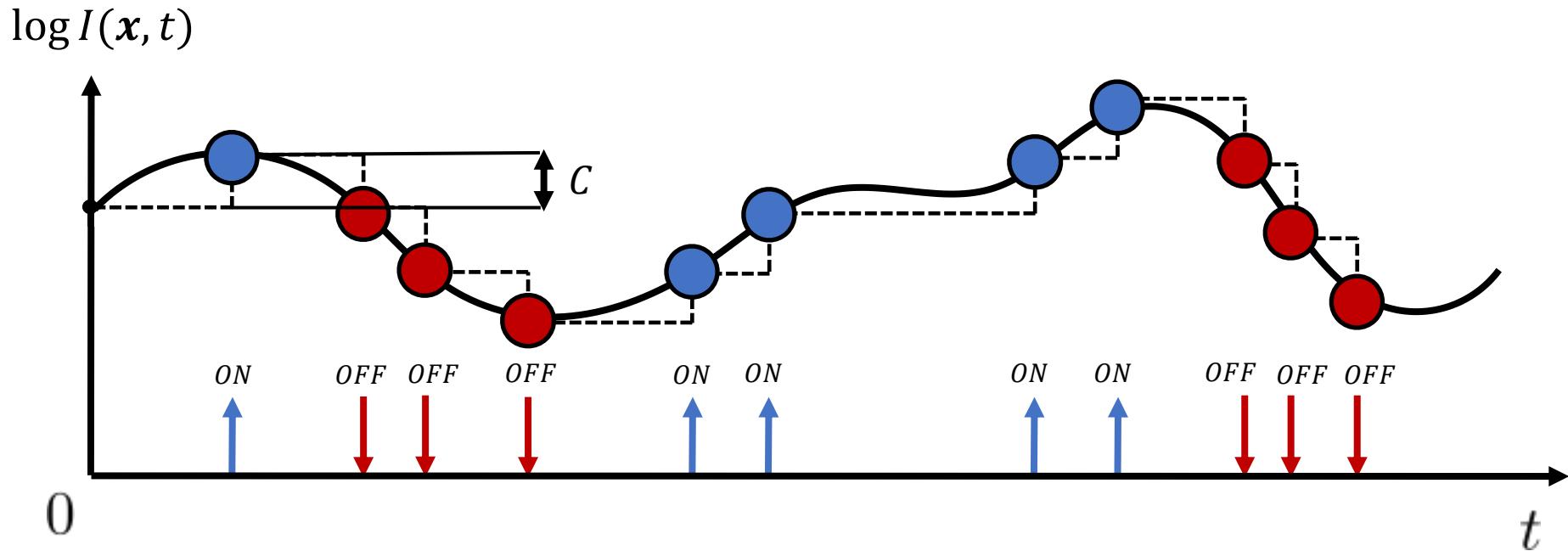
# The Paradigm: Event-by-event Processing

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



# Recall the Event Generation Model

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

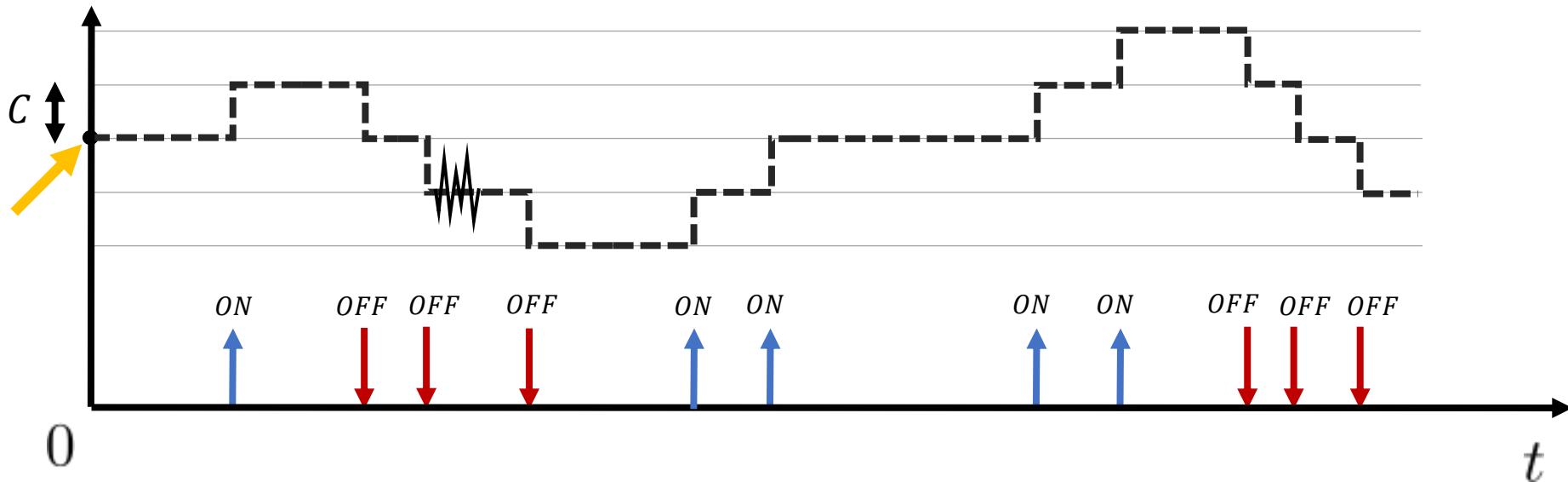


Light ( $\log I$ ) has been transduced into asynchronous events...

**Given the events, can we recover the absolute intensity?**

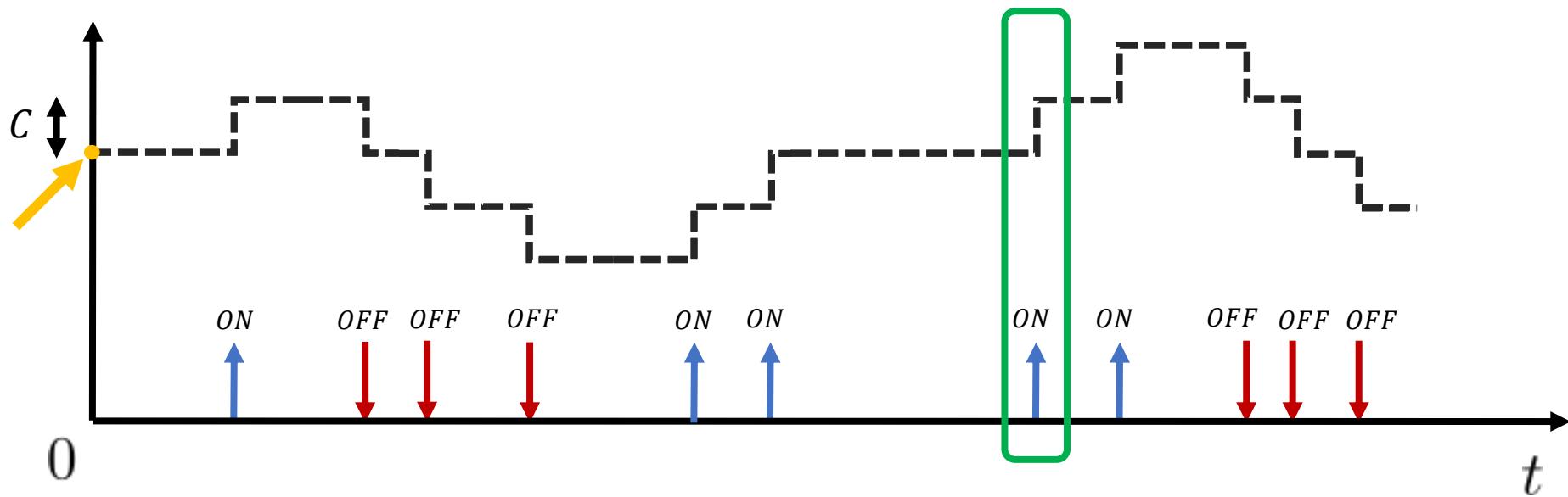
# Let us try to recover the pixel's intensity

- Events represent **intensity changes**  $\Rightarrow$  **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...

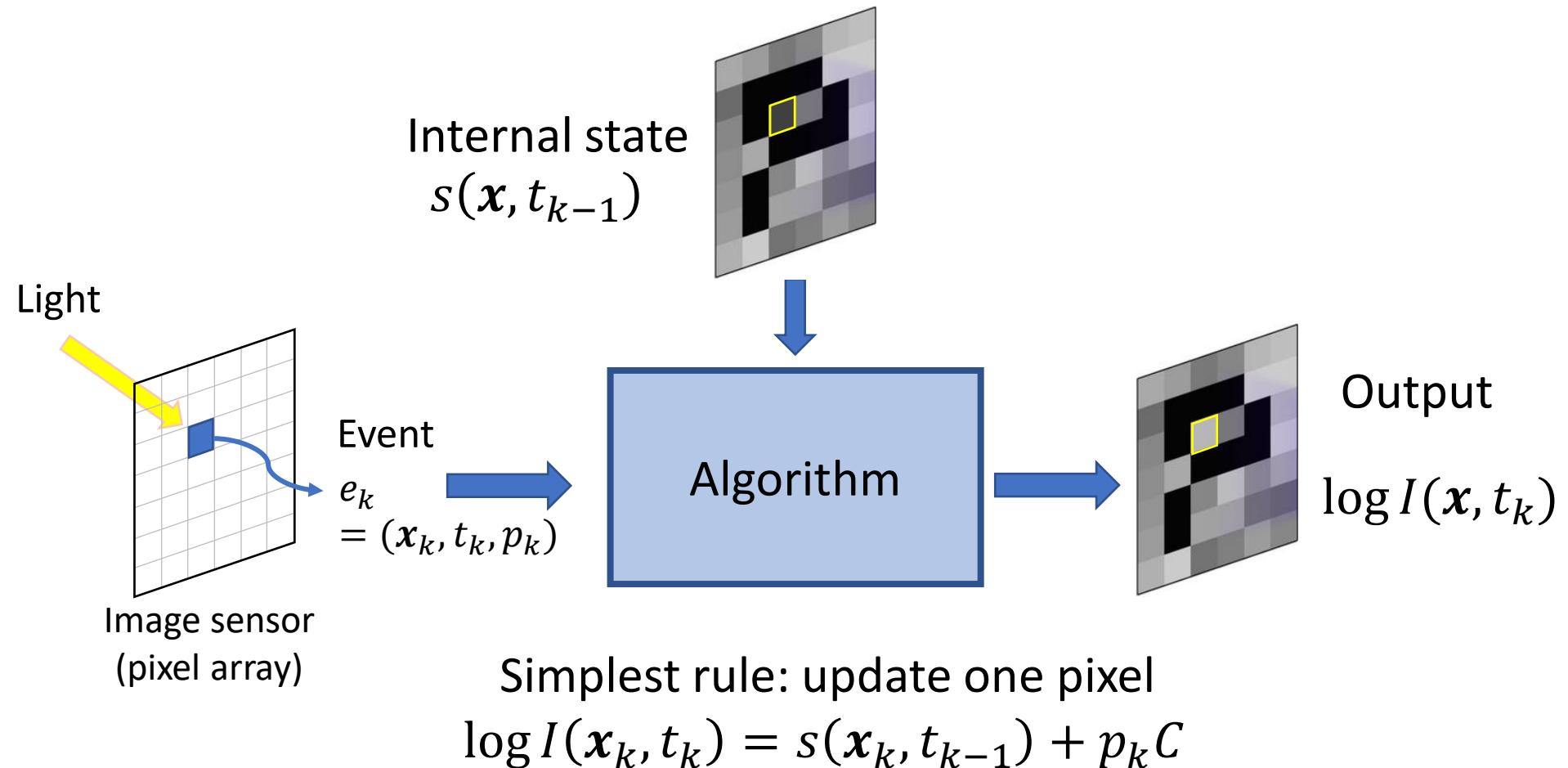
# Let us try to recover the pixel's intensity



- 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**



# Event Integration

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

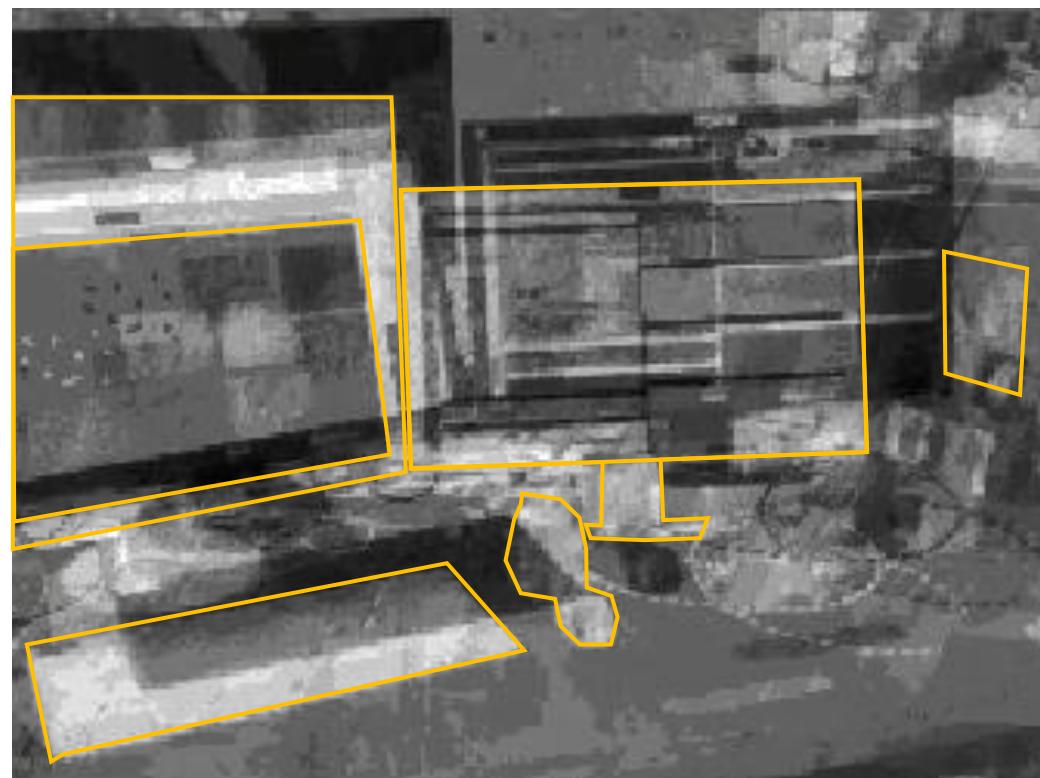
Log-intensity  
at time  $t$

Log-intensity  
at time  $t=0$

Increment  
intensity in  $[0, t]$  is

$$\int_0^t \frac{dL(v)}{dt} dv \approx \sum_k \Delta L_k$$

$$\tilde{L}(x_k, t_k) = s(x_k, t_{k-1}) + p_k \quad , \text{starting from } s=0$$



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

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

# Event Integration

- Estimated intensity. Intuition:

$$L(t) = L(0) + \underbrace{\int_0^t \frac{dL}{dt}(\tau) d\tau}_{\text{Increment } \Delta L := L(t) - L(0)}$$
$$\log \hat{I}(x, t) = \log I(x, 0) + \sum_{0 < t_k \leq t} p_k C \delta(x - x_k) \delta(t - t_k)$$

↑  
pixel

↑  
Intensity at  $t = 0$   
(offset)

↑  
0 <  $t_k \leq t$   
polarity

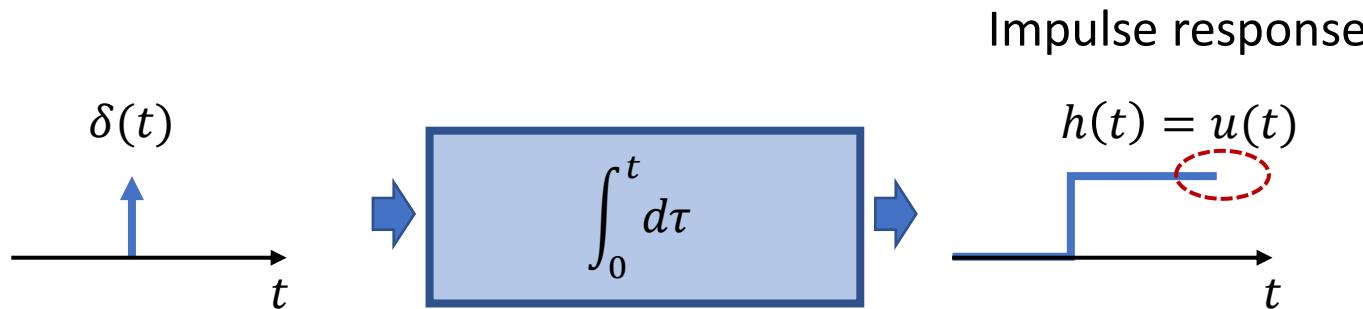
↑  
Kronecker delta (discrete variable)

↑  
Dirac delta (continuous variable)

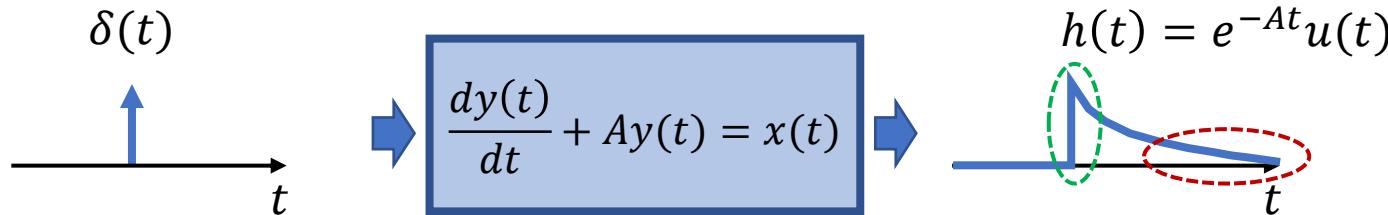
↑  
Intensity increment  $\Delta \log I$

# 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  $y(t)$ :

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

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

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  $\omega$  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  $\omega \rightarrow 0$ :

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

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

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

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:

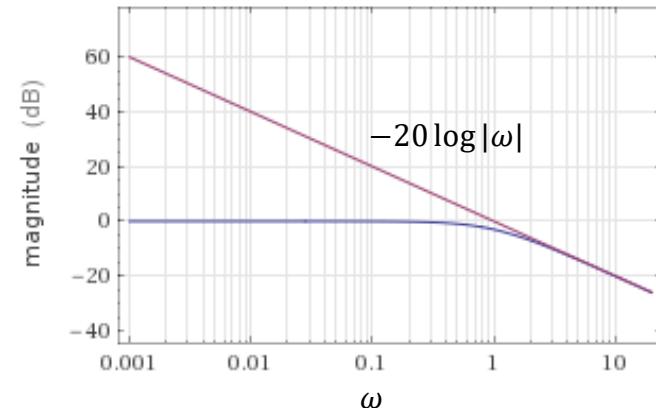
$$H(s) = \frac{1}{s + A} \Leftrightarrow e^{-At} u(t) = h(t)$$

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:

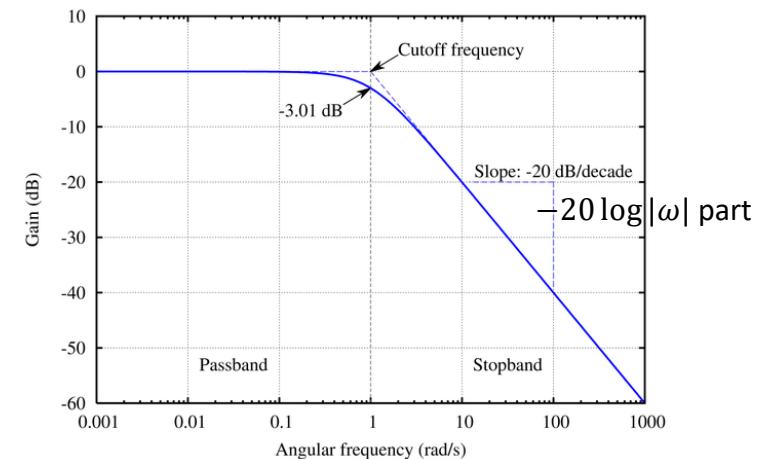
$$h(t) = e^{-\frac{t}{\tau}} u(t)$$

where  $\tau$  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**  $\int_0^t f(v)dv \xrightarrow{\text{Laplace}} \frac{F(s)}{s}$  has a Fourier response  $\frac{1}{s} = \frac{1}{j\omega}$  that blows up at  $\omega = 0$ .

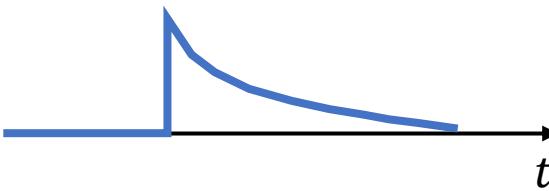


**Leaky integration** smooths off the infinite value at  $\omega = 0$ . Thus the amplitude spectrum  $20 \log \frac{1}{|\omega|} = -20 \log |\omega|$ , except that it is not  $\infty$  at  $\omega = 0$ .

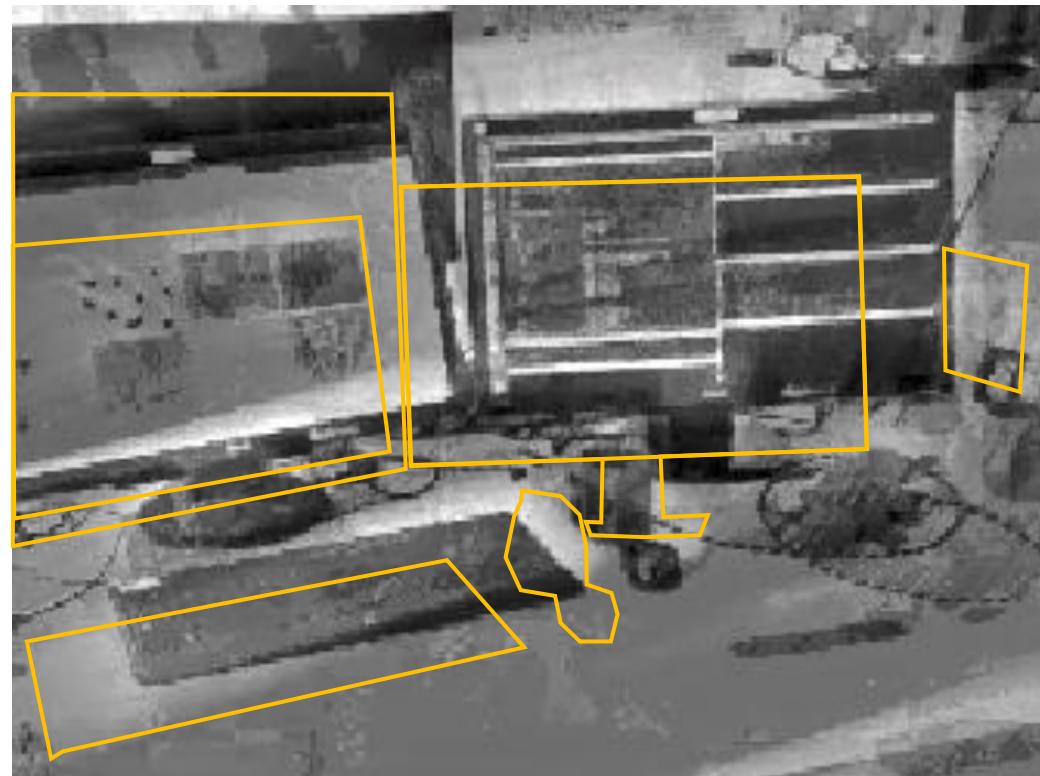


$$\alpha = 0.3$$

# Leaky integration



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

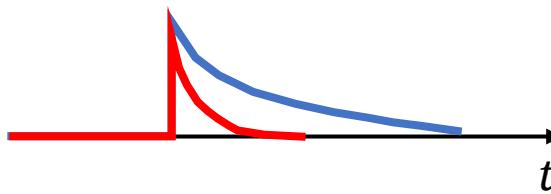


Leaky integration of events can recover  $\approx$  absolute intensity, **even without knowing the offset  $L(x, 0)$ !**

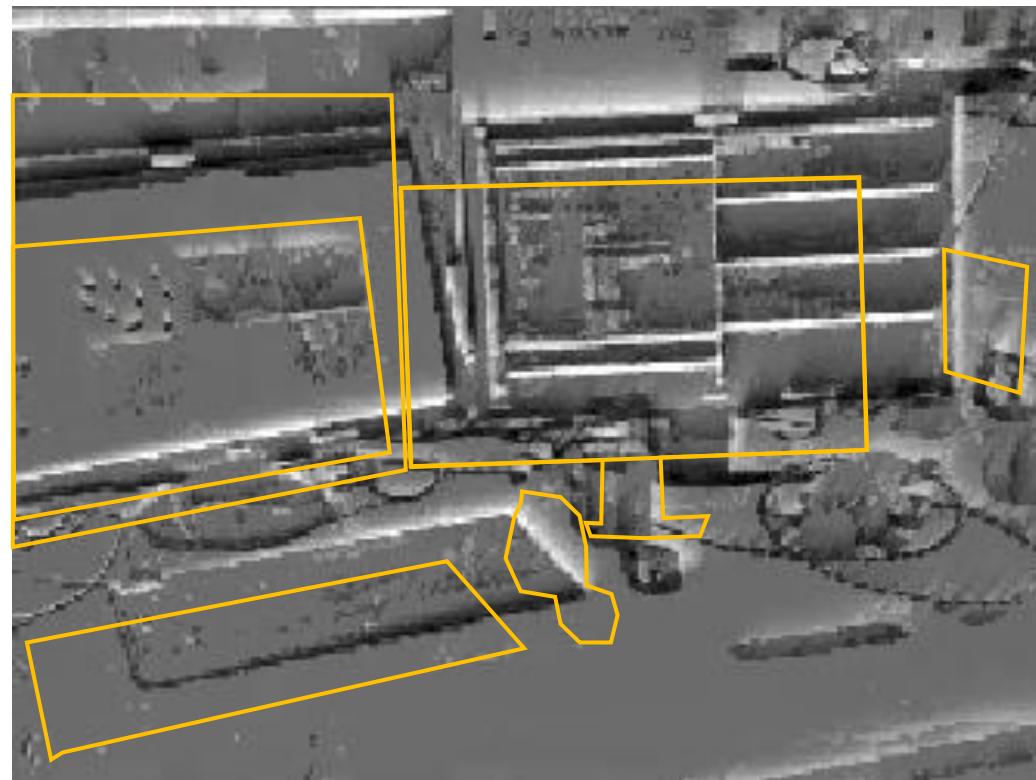
Isn't it magic?  
Temporal leakage "forgets" the initial intensity.  
Spatial filtering is **not** needed.

$$\alpha = 2$$

# Leaky integration. Short decay



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

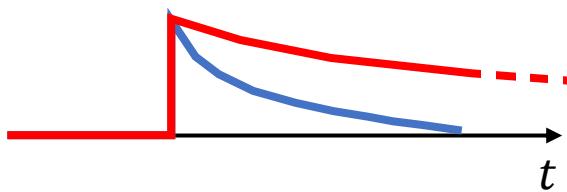


**Too much leakage** (forgetting quickly)  
Faster decay  $\rightarrow$  not integrating well

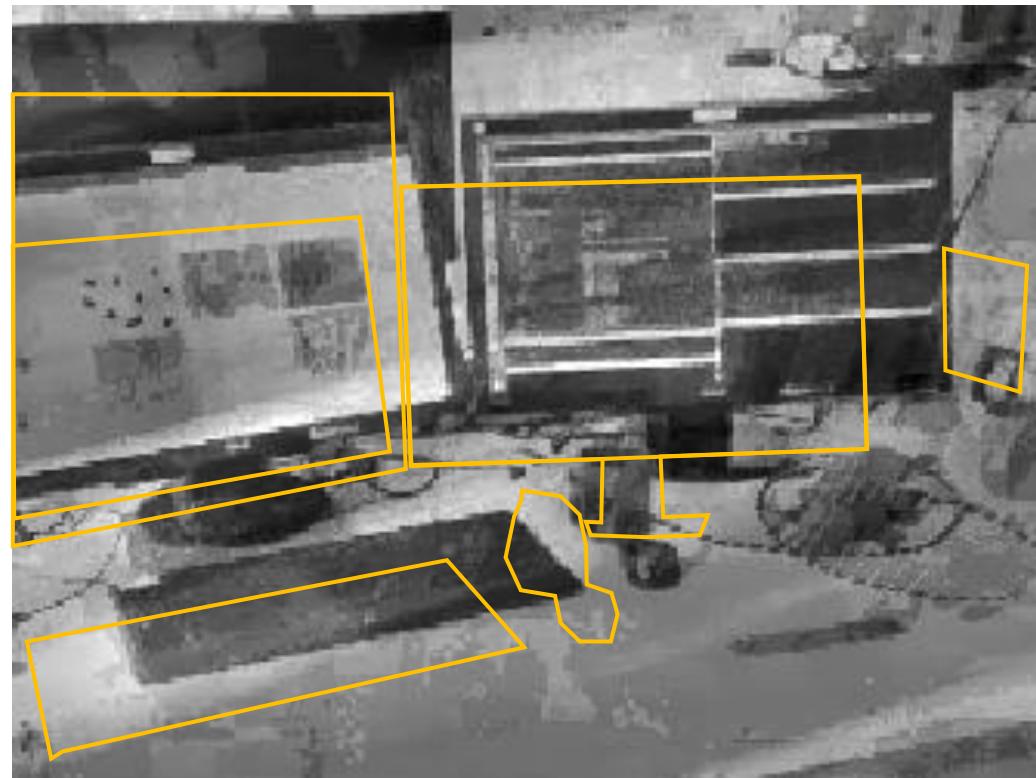
(even if offset  $L(x, 0)$  is washed out)

$$\alpha = 0.1$$

# Leaky integration. Long decay



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



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

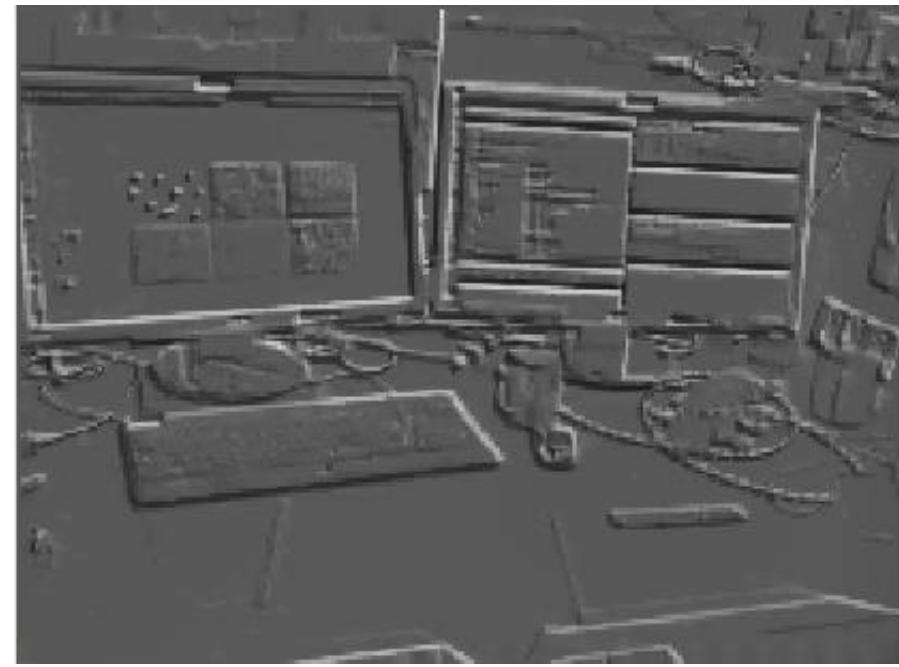
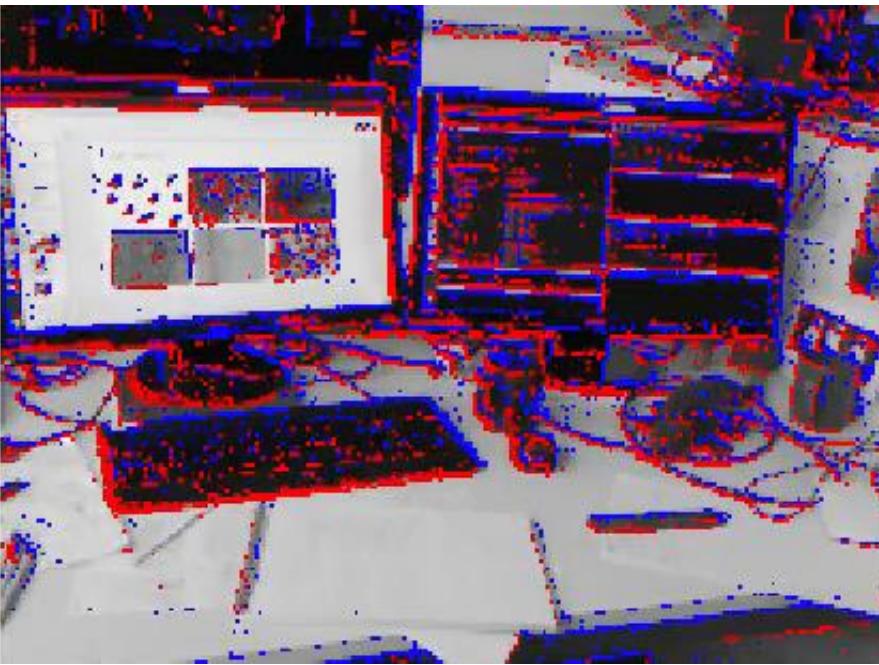
(even if offset  $L(x, 0)$  is washed out)

# Noise-Free, Simulated Events

Desktop Scene

$$\alpha = 0$$

# Direct (ordinary) Integration



Input: **events** (ON, OFF)

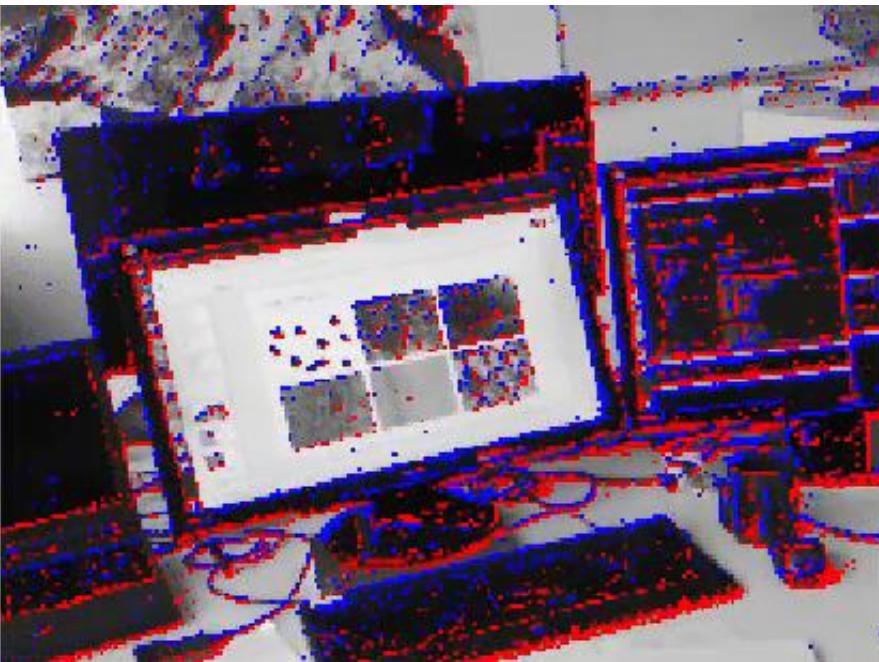
Grayscale frames are just used for visualization.

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

**starting from  $s=0$**

$$\alpha = 0$$

# Direct (ordinary) Integration



Input: **events (ON, OFF)**

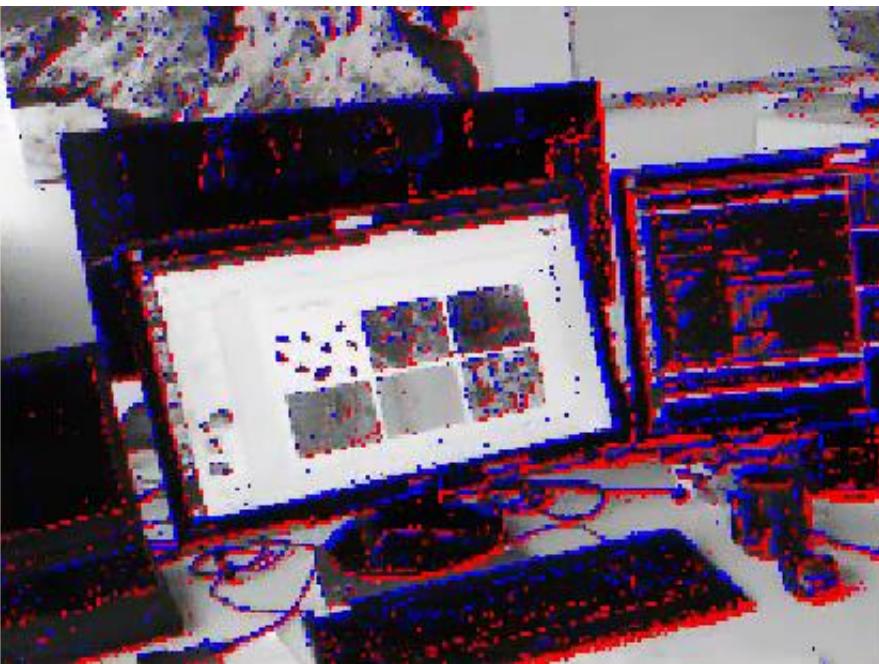
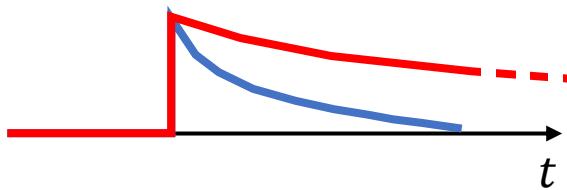
Grayscale frames are just used for visualization.

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

starting from  $s=0$

$$\alpha = 0.1$$

# Leaky integration. Long decay



Input: **events** (ON, OFF)

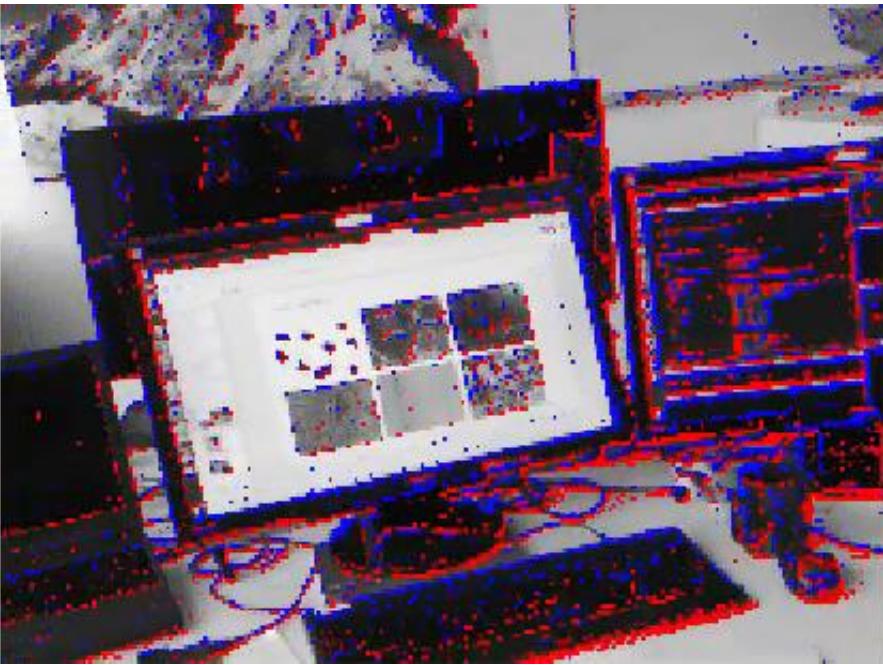
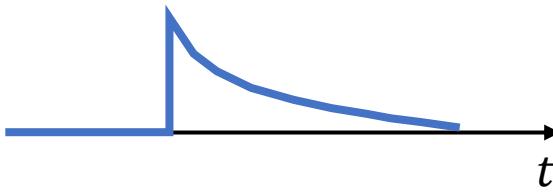
Grayscale frames are just used for visualization.

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

starting from  $s=0$

$$\alpha = 0.3$$

# Leaky integration



Input: **events (ON, OFF)**

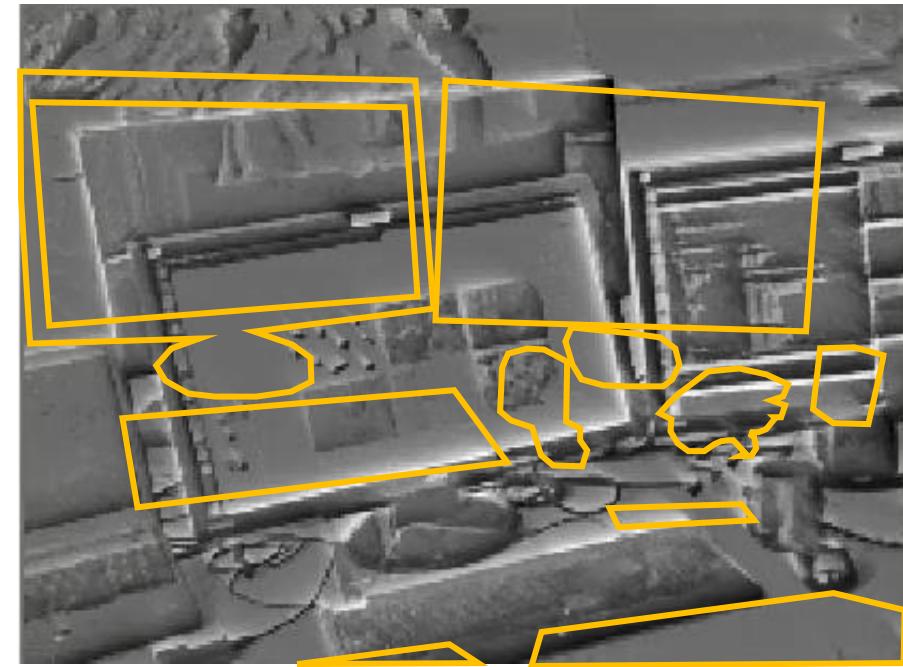
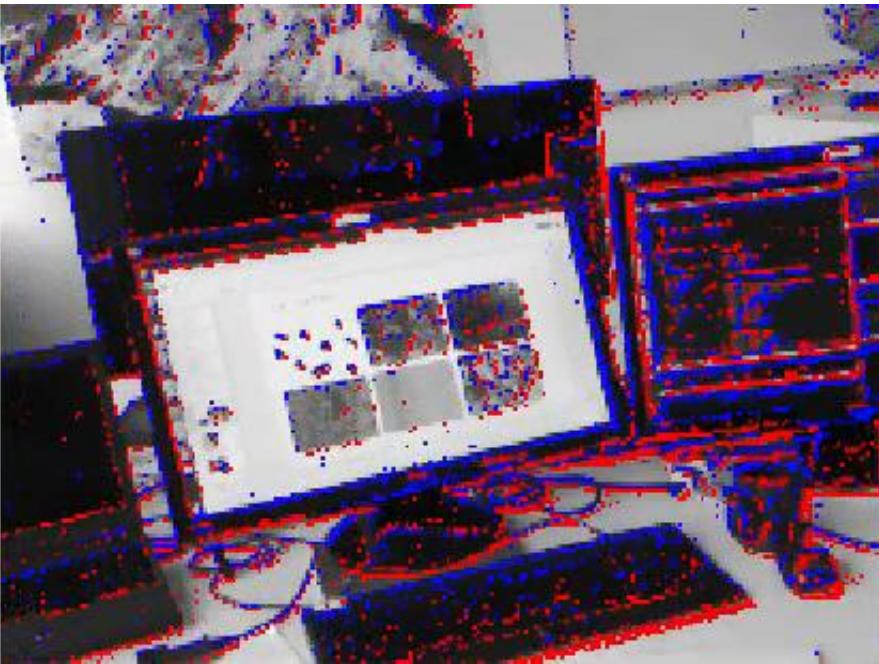
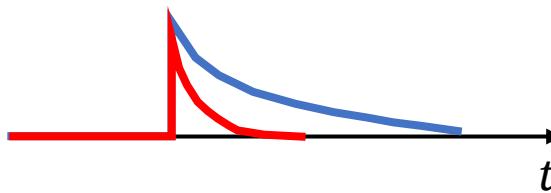
Grayscale frames are just used for visualization.

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

starting from  $s=0$

$$\alpha = 2$$

# Leaky integration. Short decay



Input: **events** (ON, OFF)

Grayscale frames are just used for visualization.

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

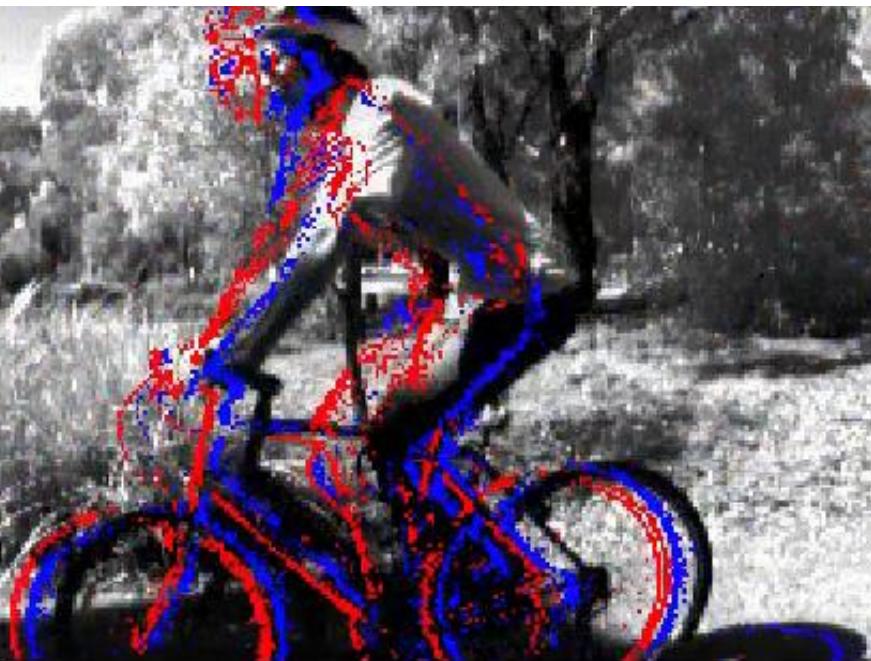
starting from  $s=0$

# Real data (noisy)

Bicycle scene

$$\alpha = 0$$

# Direct (ordinary) Integration



Input: **events** (ON, OFF)

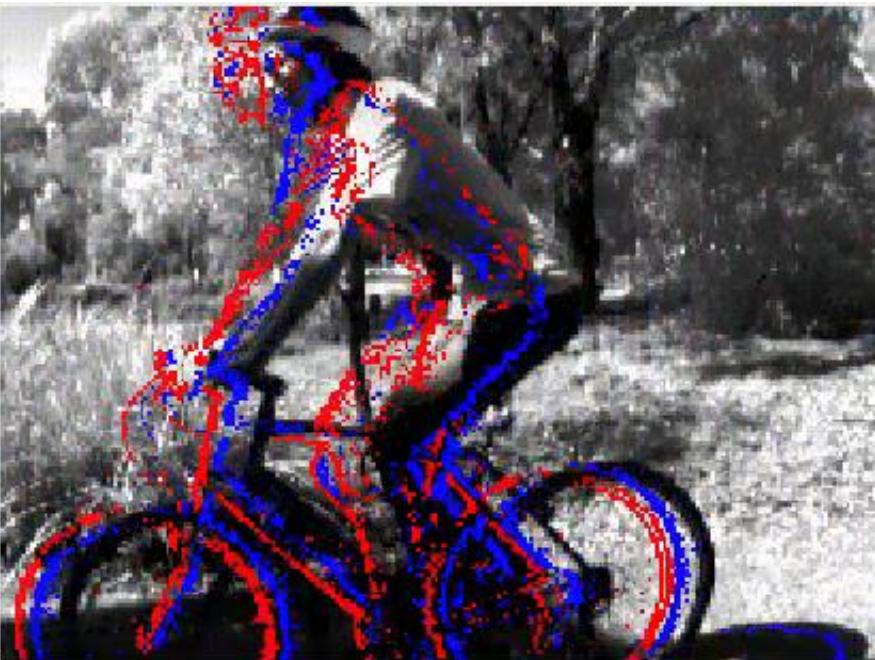
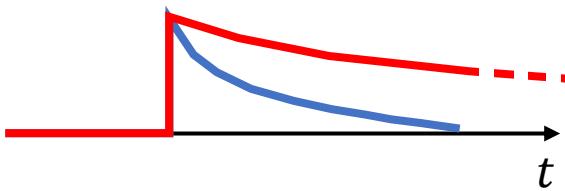
Grayscale frames are just used for visualization.

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

starting from  $s=0$

$$\alpha = 0.1$$

# Leaky Integration. Long decay



Input: **events** (ON, OFF)

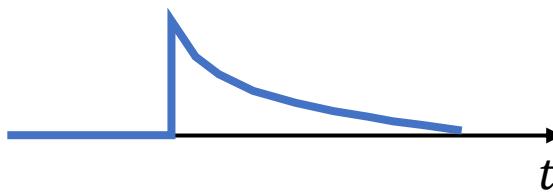
Grayscale frames are just used for visualization.

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

starting from  $s=0$

$$\alpha = 0.3$$

# Leaky Integration



Input: **events** (ON, OFF)

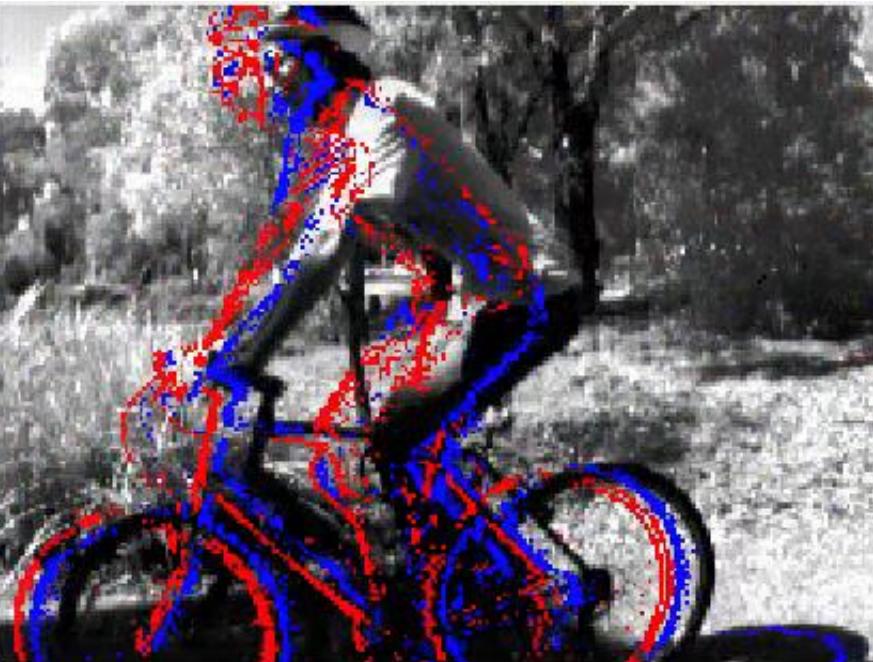
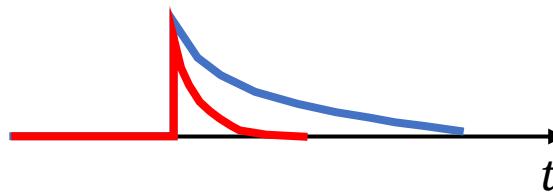
Grayscale frames are just used for visualization.

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

starting from  $s=0$

$$\alpha = 2$$

# Leaky Integration. Short decay



Input: **events** (ON, OFF)

Grayscale frames are just used for visualization.

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

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



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



Output of the filter

# 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 Using Event Cameras](#), ACCV 2018.
- Scheerlinck et al. [Asynchronous Spatial Image Convolutions for Event Cameras](#), RA-L 2019
- Jupyter notebook [demo](#)

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego  
Chair: Robotic Interactive Perception

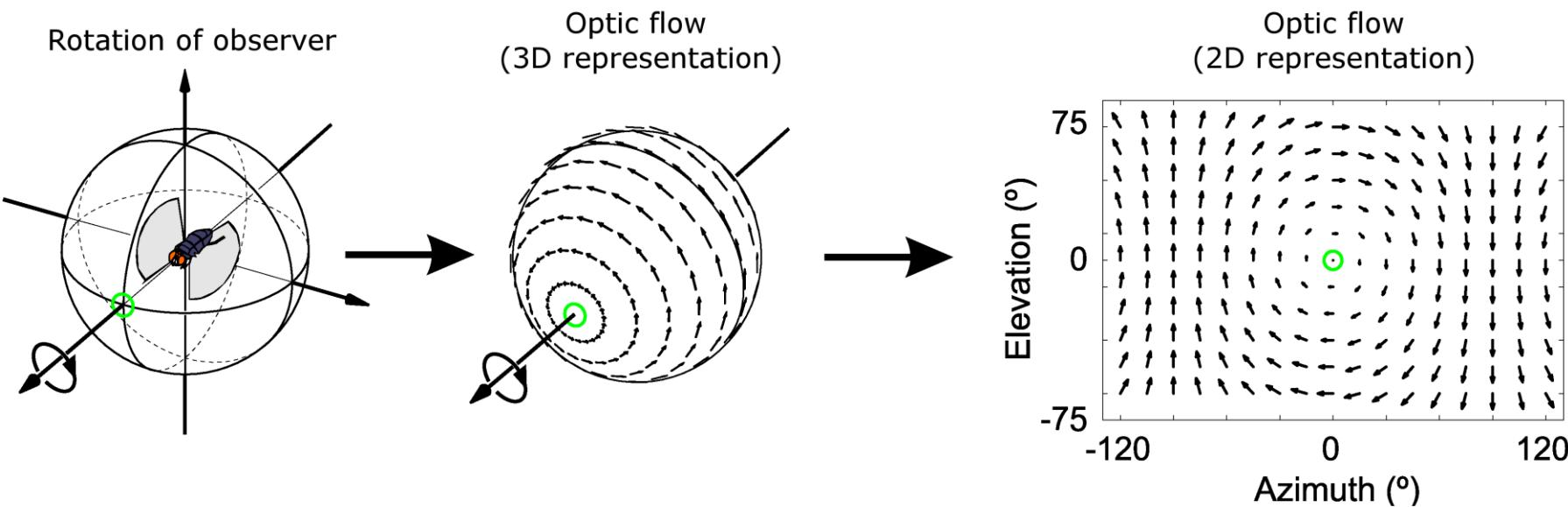
[guillermo.gallego@tu-berlin.de](mailto:guillermo.gallego@tu-berlin.de)

<http://www.guillermogallego.es>

# Optical Flow

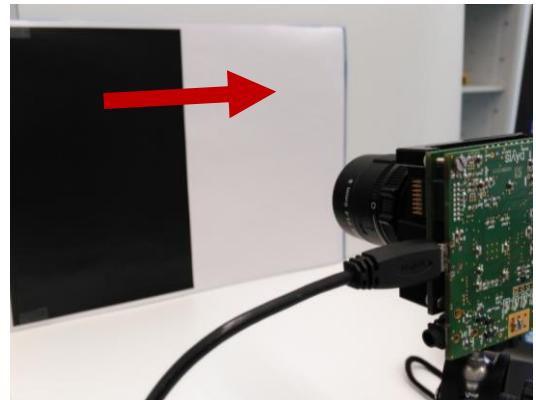
# What is Optical Flow?

- A technical topic with > 40 years of research
- It is the **apparent motion** of pixels on the image plane.  
Find it without knowledge about scene geometry or motion
- What is a “flow field”?

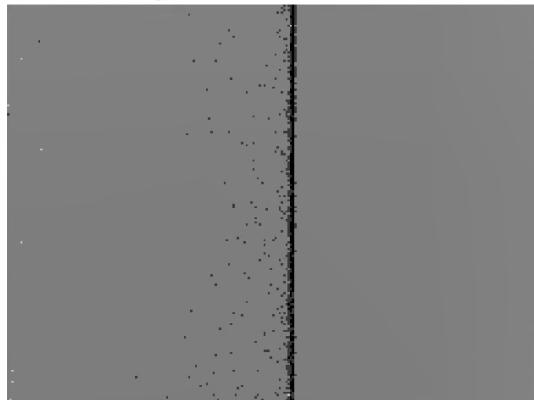


# A moving edge

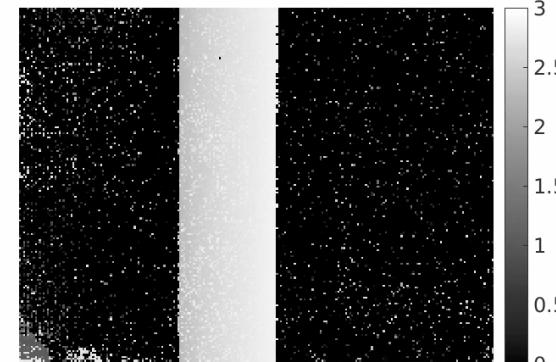
Consider the motion of an edge, viewed by an event-based camera:



Event image (1000 events).  $t = 2.856$



Time of the last event

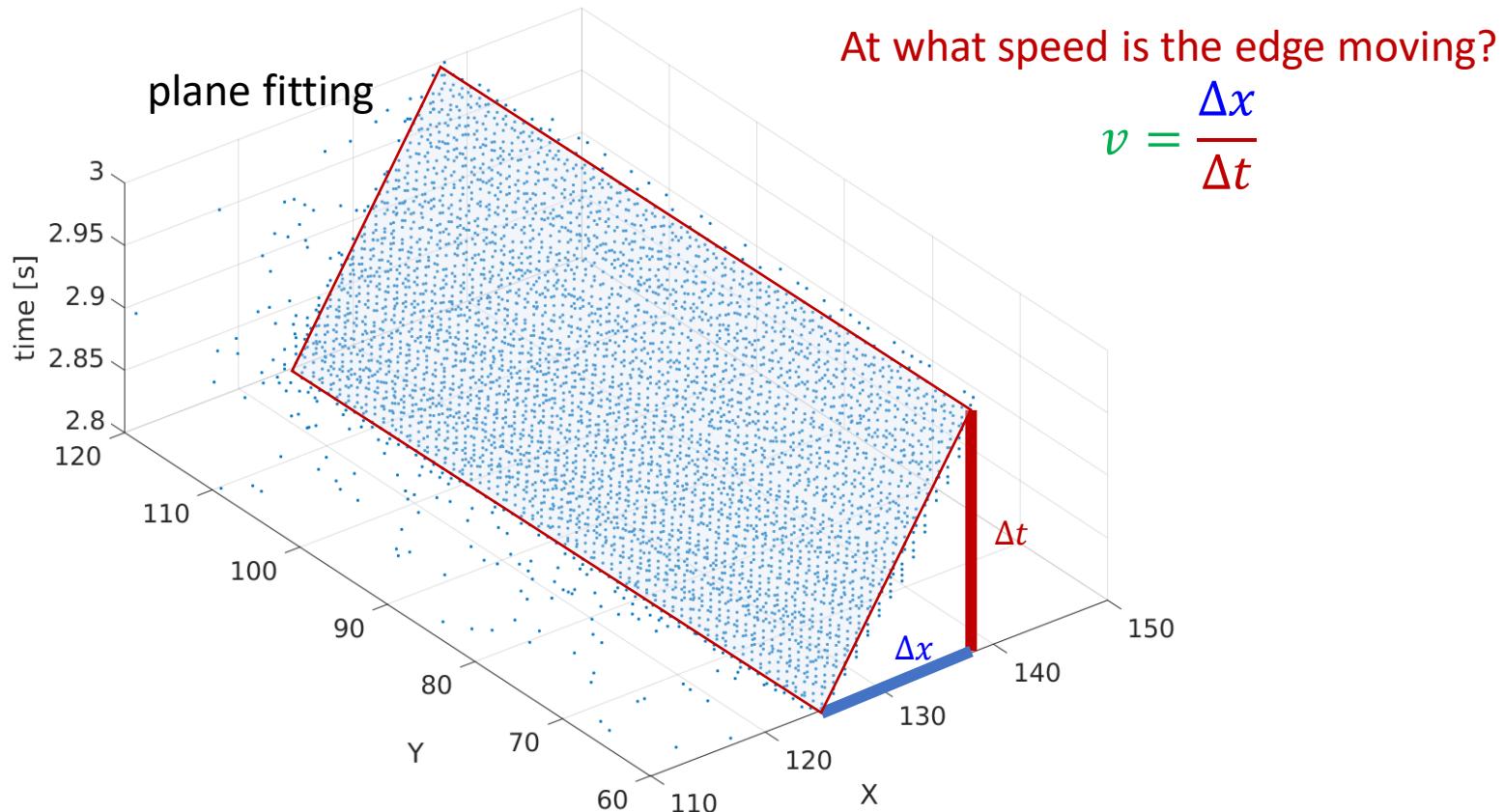


White pixels become black

- brightness decrease
- negative events (in black color)

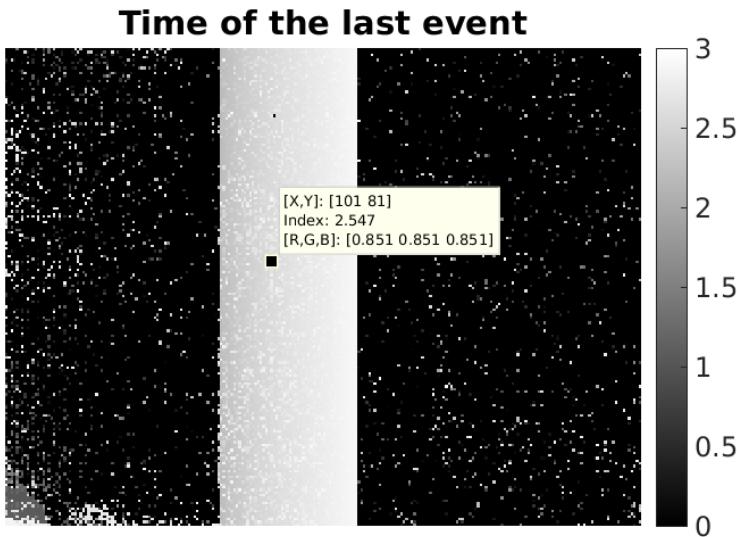
# A moving edge

The same edge, visualized in space-time  
Events are represented by dots (point "cloud")

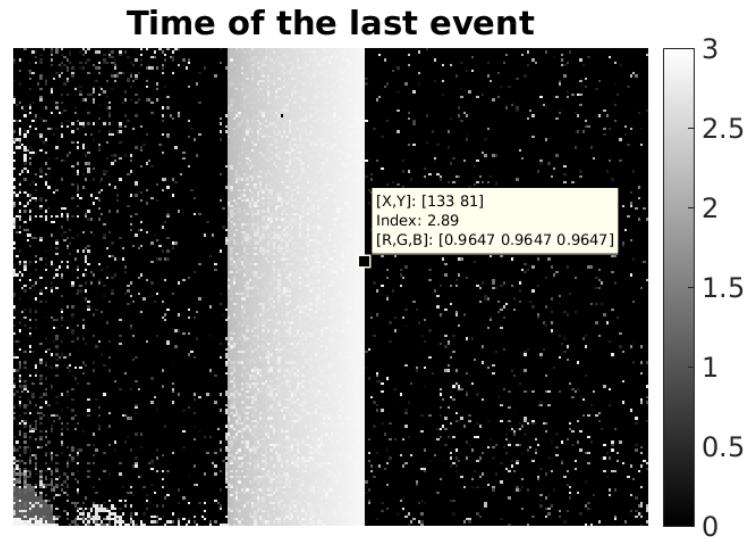


# A moving edge

At  $t = 2.547$ ,  $x = 101$



At  $t = 2.89$ ,  $x = 133$

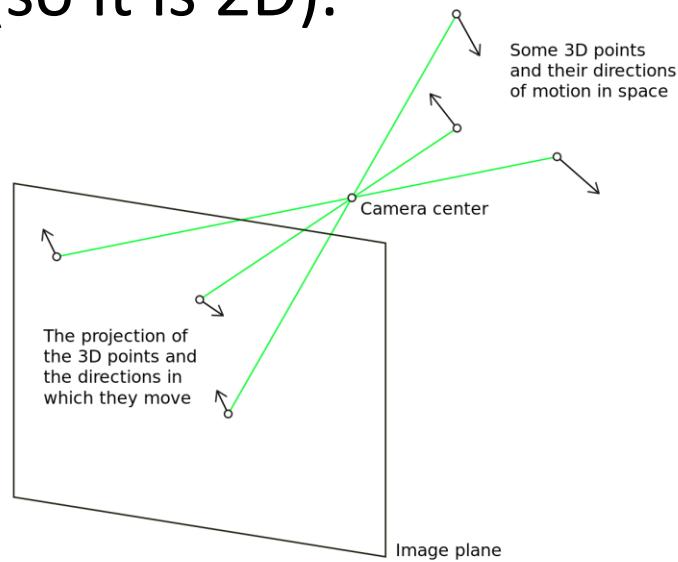


Speed of the edge:

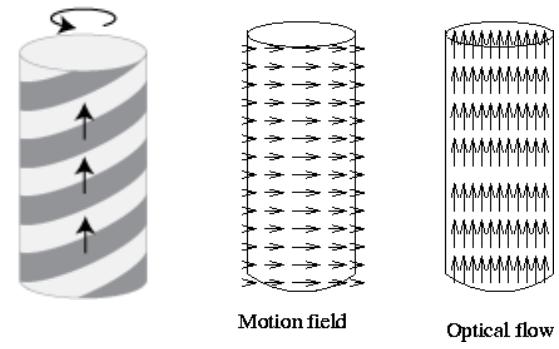
$$v = \frac{(133 - 101)}{(2.89 - 2.547)} = 93.3 \text{ pix/sec}$$

# What is Optical Flow?

- **Optical Flow vs. Motion Field**
- **Motion field** is the physical motion (3D) projected onto the image plane (so it is 2D).

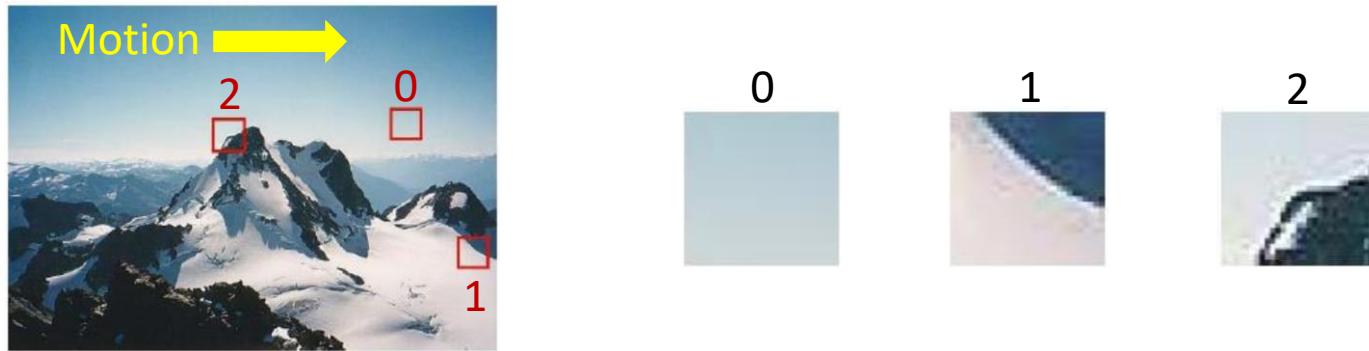


- OF and MF may differ: barber-pole illusion.



# Where is optical flow $\approx$ motion field?

- Three types of neighborhoods of image points:



- **Region of constant brightness:** “impossible” to determine optical flow. Algorithms provide a solution based on priors (regularizer).
- **Single edge (1D):** can only determine the optical flow perpendicular to the edge. (Normal component of motion field, called normal flow)
- **Corner or complex edge pattern:** brightness gradients in more than one direction. Unambiguous to determine motion
- In all three cases there is a motion field vector, but only in one case the optical flow is a good approximation of it (case 2).
- We are omitting noise, occlusions, etc. in this analysis.

# Motion field

- The motion field has two components:
  - Rotational** component (independent of depth Z)
  - Translational** component (depends on depth Z). Parallax

3D velocity

2D velocity

linear

angular

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} -1 & 0 & x \\ 0 & -1 & y \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} xy & -1-x^2 & y \\ 1+y^2 & -xy & -x \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

Due to translation and depth

Due to rotation

The diagram illustrates the decomposition of 3D velocity into linear and angular components. The 3D velocity vector is shown branching into a red arrow labeled "linear" and a blue arrow labeled "angular". These components are then mapped through a depth-dependent transformation (indicated by a green circle with a dashed line) to produce the 2D velocity vector. A red brace below the first matrix indicates the "Due to translation and depth" component, and a blue brace below the second matrix indicates the "Due to rotation" component.

$x, y$ : pixel coordinates

# Brightness Constancy

- It assumes that brightness on the image plane does not change for projections of the same object (3D) point:

$$L(\mathbf{x}(t), t) = \text{const}$$

for all points  $\mathbf{x}(t)$  corresponding to an object point.

- In practice, it holds for short time (differential viewpoint):

$$0 = \frac{dL(\mathbf{x}(t), t)}{dt} = \nabla L \cdot \frac{d\mathbf{x}}{dt} + \frac{\partial L}{\partial t}$$

where  $\dot{\mathbf{x}} = d\mathbf{x}/dt$  is the point's velocity  $\mathbf{v}$ , i.e., the **optical flow**.

- If  $L$  is known, it gives 1 equation in 2 unknowns → underdetermined
- **Problem:** event cameras do not provide  $L$ . How do we get around?

# Frame-based vs. event-based Optical Flow

- **Similarities?**

- Some try to exploit common **principles** (e.g., brightness constancy), even if that produces different methods.
- Suffer from similar fundamental **problems**:
  - The **aperture problem** does not vanish by having events instead of frames
  - (But frame-based acquisition suffers from motion blur. Events do not)
- Computing optical flow on every pixel and time (i.e., each point in space-time) is **expensive**.

# Frame-based vs. event-based Optical Flow

- **Differences?** Different characteristics of signals:
  - **Geometry:** synchronous and dense vs. asynchronous and sparse
  - **Photometry:** absolute intensity vs. temporal contrast
  - **Noise:** event noise vs. intensity noise
- **Scenarios:** flow in high-speed & HDR scenarios
- **Technology:** maturity

# Why is event-based optical flow attractive?

- Events allow to obtain flow in **high speed** and **HDR** scenarios, and at **low power**
- Potentially **more efficient** computations by focusing on edges (the informative regions of the image plane)
- Potentially implement in neuromorphic hardware (**biologically plausible** computational model).  
Trying to understand Nature

# Opportunities. What's missing?

- **Datasets and benchmarks** (metrics and protocols) for comparison that foster progress in the field, to advance the state of the art.
  - Ground truth optical flow is difficult to obtain.
  - We may use as ground truth the motion field from **simulated** 3D scenes and camera motions. However, event noise is not modeled well yet. Real-world data is needed.
  - Metrics: Quantify **trade-offs** in accuracy, efficiency, latency, etc.
- A **thorough comparison** of multiple methods in a variety of scenes (texture, illumination) and motions (speed, occlusions, parallax, etc.) that will allow us to **identify key ideas and best practices**.

(Discussion):

- This is an emerging field of research (> 2008).  
We are still in an **exploratory phase** of methods.
- Presumably, data-driven methods will soon dominate, as it has happened in conventional computer vision.

# Literature Review

# Questions for each method?

- **Key principle** (main idea of the method):
  - **Assumptions.** What is the **additional knowledge**?
  - What is **being optimized**?
  - Differential method, correlation-based, phase-based, based on event generation model, data-driven (CNN), ...
  - What event **representations** are used (at input or intermediate)?
  - What is the **inference mechanism** to compute flow given events?
- **Characteristics:**
  - Output: full flow vs normal flow?
  - Output: dense vs. sparse?
  - Method: Event-by-event vs. packet of events?
  - Method: Bio-inspired vs not?
  - Method: Does it exploit polarity? yes/no
- **Pros / Cons** (limitations)
  - Supported by **experiments**?

# Taxonomy of some event-based OF methods

- Multiple dimensions (criteria) can be considered

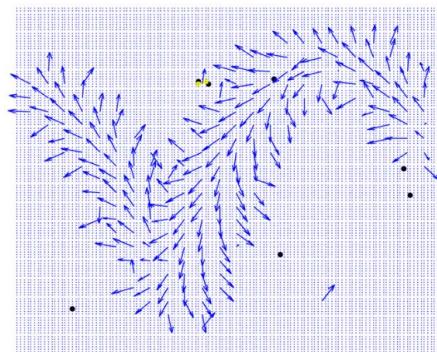
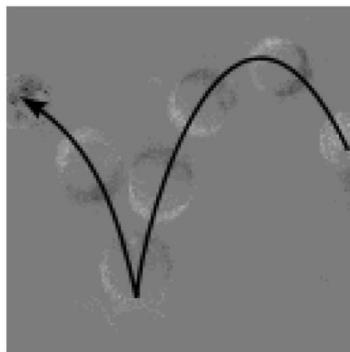
Table 2

Classification of several optical flow methods according to their output and design. Some methods provide full motion flow (F) whereas others only its component normal to the local brightness edge (N). The output may be a dense (D) flow field (i.e., optical flow for every pixel at some time) or sparse (S) (i.e., flow computed at selected pixels). According to their design, methods may be model-based or model-free (Artificial Neural Network - ANN), and neuro-biologically inspired or not.

Reference	N/F?	S/D?	Model?	Bio?
Delbruck [80], [190]	Normal	Sparse	Model	Yes
Benosman et al. [190], [191]	Full	Sparse	Model	No
Orchard et al. [157]	Full	Sparse	ANN	Yes
Benosman et al. [21], [190]	Normal	Sparse	Model	No
Barranco et al. [192]	Normal	Sparse	Model	No
Barranco et al. [193]	Normal	Sparse	Model	No
Conradt et al. [194]	Normal	Sparse	Model	No
Brosch et al. [134], [195]	Normal	Sparse	Model	Yes
Bardow et al. [117]	Full	Dense	Model	No
Liu et al. [105]	Full	Sparse	Model	No
Gallego [128], Stoffregen [154]	Full	Sparse	Model	No
Haessig et al. [196]	Normal	Sparse	ANN	Yes
Zhu et al. [22], [119]	Full	Dense	ANN	No
Ye et al. [153]	Full	Dense	ANN	No
Paredes-Vallés [102]	Full	Sparse	ANN	Yes

# Adaptation of Frame-based methods

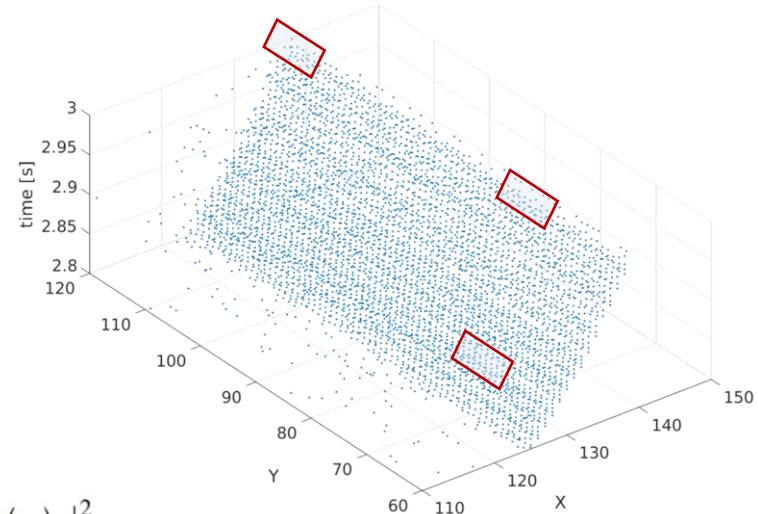
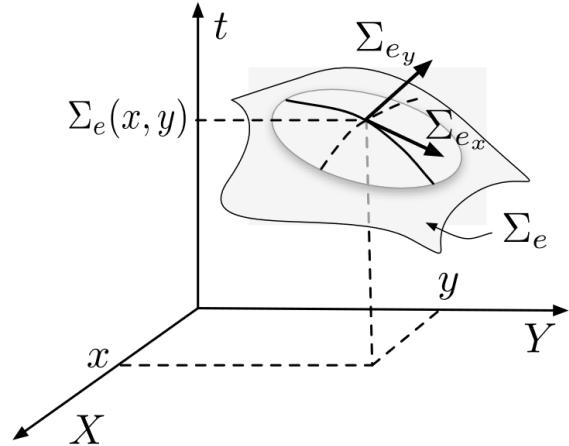
- Recall **brightness constancy**:  $\nabla L \cdot \frac{dx}{dt} + \frac{\partial L}{\partial t} = 0$
- At each pixel: 1 eq., 2 unknowns → Lucas-Kanade (LK) assumed **flow is locally constant** → more equations in the same 2 unknowns.
- Event camera does not provide  $L$  ...
  - Adapted LK (2012) proposes to **use increment image of events** in place for  $L$ .



- Critique by Brosch et al (2015):
  - Differential methods do not work so well because computing derivatives with only few events per pixel is not reliable.
  - **Methods working on the event cloud directly (without derivat.) are preferred**

# Optical Flow by Local Plane Fitting

- **Idea:** fit planes locally (in space-time) to the point cloud of events
- Why? A moving edge produces a trail of events that resembles a surface. Fit a plane, get the velocity from the plane coefficients.

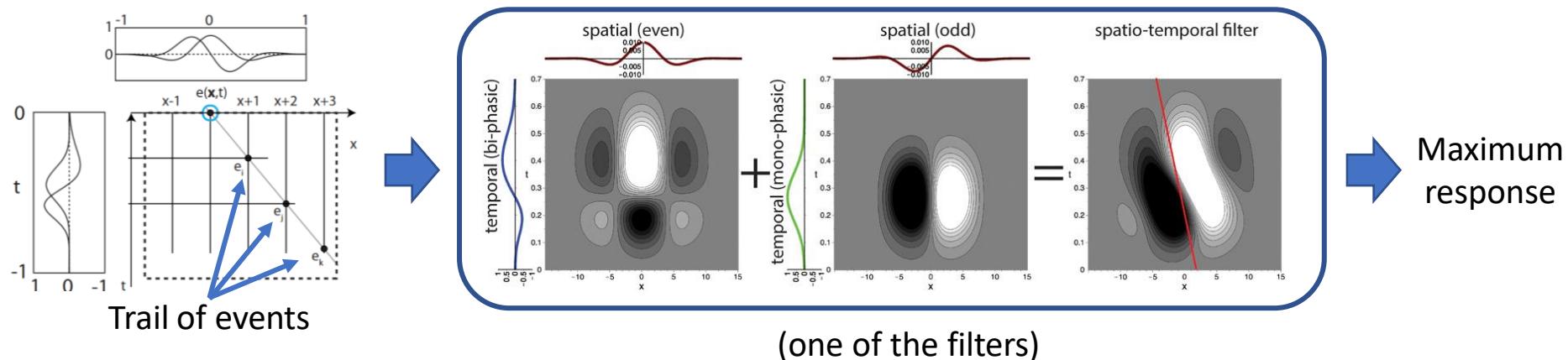


$$\nabla \Sigma_e = \left( \frac{1}{v_x}, \frac{1}{v_y} \right) \quad \tilde{\Pi}_0 = \underset{\Pi \in \mathbb{R}^4}{\operatorname{argmin}} \sum_i \left| \Pi^T \begin{pmatrix} \mathbf{p}_i \\ t_i \\ 1 \end{pmatrix} \right|^2$$

- Can only determine the **normal component** of the flow
- Sparked the concept of event “lifetime”

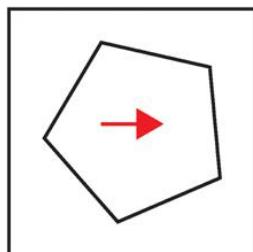
# Space-time Direction-Selective Filter Bank

- **Idea:** Convolve events with **space-time filters** of selected response
  - Biologically / physiologically inspired
  - Build **motion-direction sensitive filters** from **separable components**

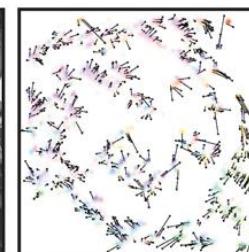


- **Results:**

Results on translation

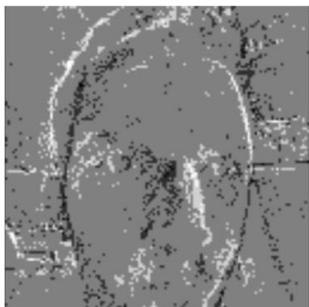


Results on rotation

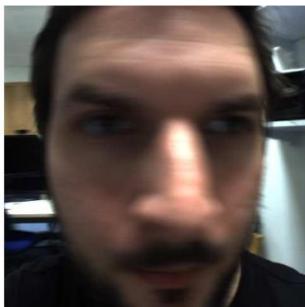


# SOFIE: Simultaneous Optical Flow & IE

- **Joint optimization** over **image brightness** and **optical flow** to explain a volume of events (voxel grid)
- **Idea:** Penalize **deviations** for all equations (brightness constancy and event generation model) and assume **smooth** solution ( $\mathbf{u}, L$ ).
- Solve a variational optimization problem:



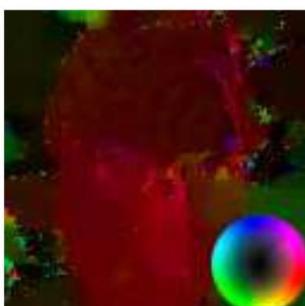
(a) Raw event camera output



(b) Standard camera image



(c) Intensity estimate from events



(d) Optical flow from events

$$\begin{aligned} \min_{\mathbf{u}, L} \int_{\Omega} \int_T & \left( \lambda_1 \|\mathbf{u}_x\|_1 + \lambda_2 \|\mathbf{u}_t\|_1 + \lambda_3 \|L_x\|_1 + \right. \\ & \left. \lambda_4 \|\langle L_x, \delta_t \mathbf{u} \rangle + L_t\|_1 + \lambda_5 h_\theta(L - L(t_p)) \right) dt dx \\ & + \int_{\Omega} \sum_{i=2}^{|P(\mathbf{x})|} \|L(t_i) - L(t_{i-1}) - \theta \rho_i\|_1 dx, \end{aligned}$$

**Smoothness terms**

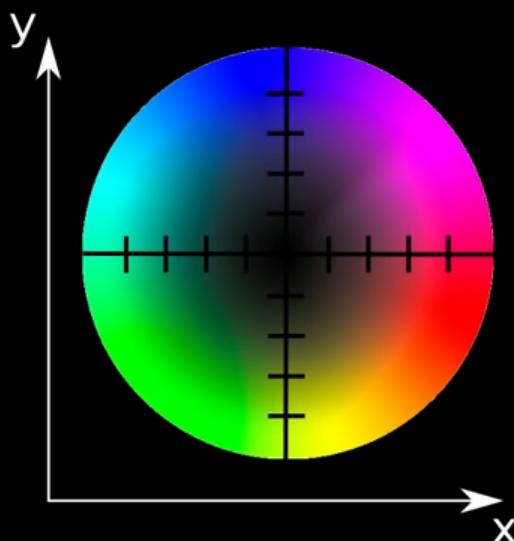
**Optical flow term (brightness constancy)**

**No-event term**

**Event term**

# SOFIE: Simultaneous Optical Flow & IE

**Conference on Computer Vision and  
Pattern Recognition (CVPR 2016)**

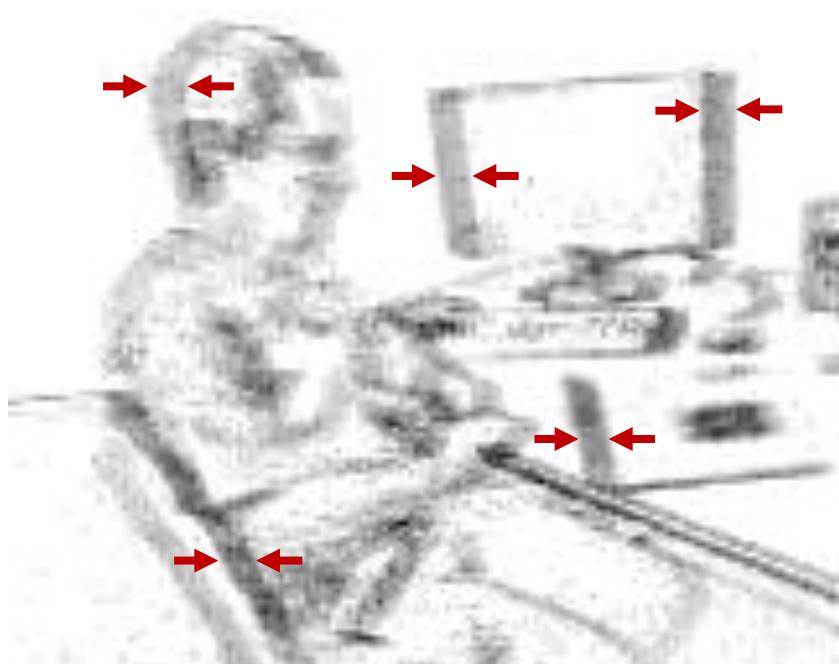


Optical Flow Visualization:

- Each Pixel's
- direction is shown as color and
- velocity as brightness
- (as shown in the color wheel)

# Optical Flow from Motion Compensation

- Assume **constant flow** in a small space-time volume (i.e., events in the volume share the same flow vector)
- **Idea:** find the flow  $\nu$  that best *aligns* corresponding events.
- Why? as an edge moves, it triggers events at pixels it crosses. Try to “**undo**” (compensate) this motion, to **recover the thin edge**.



# Optical Flow from Motion Compensation

- Assume **constant flow** in a small space-time volume (i.e., events in the volume share the same flow vector)
- **Idea:** find the flow  $\nu$  that best *aligns* corresponding events.
- Why? as an edge moves, it triggers events at pixels it crosses. Try to “**undo**” (compensate) this motion, to **recover the thin edge**.



# Optical Flow from Motion Compensation

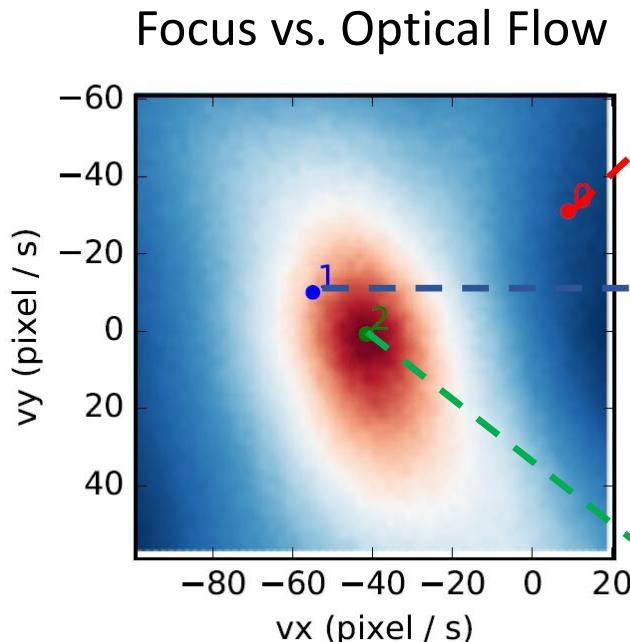
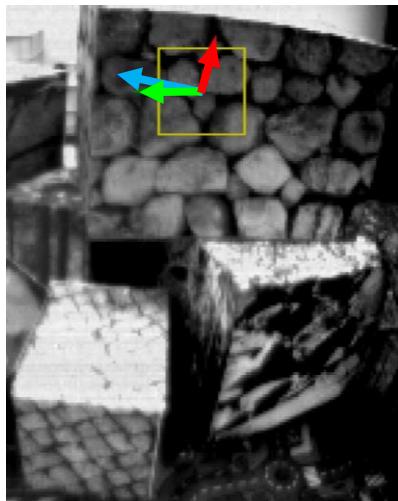
Iteration steps:

1. **Transform** events:  $e_k = (\mathbf{x}_k, t_k, p_k) \mapsto e'_k = (\mathbf{x}'_k, t_{\text{ref}}, p_k)$ 
  - Move events using candidate flow:
$$\mathbf{x}'_k \doteq \mathbf{x}_k - (t_k - t_{\text{ref}})\mathbf{v}$$
  - $t_{\text{ref}}$ : reference time (e.g., first event in the volume)
  - Easy: space = velocity \* time
2. Measure how well the warped events **align**. How?
  - Euclidean distance between events  $e'_k$  (point sets) or
  - Contrast (focus) of histograms / image of events or
  - Average time per pixel if using time surface-like representation

# Optical Flow from Motion Compensation

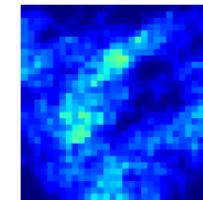
## Using histograms of events

Events generated  
by a patch

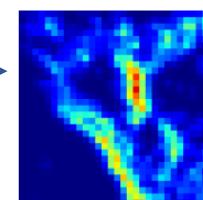


Warped Events

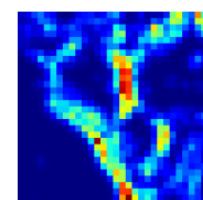
$$\mathbf{x}'_k = \mathbf{x}_k - \mathbf{v} t_k \quad \text{straight trajectories}$$



**Blurred**  
(without  
motion correction)



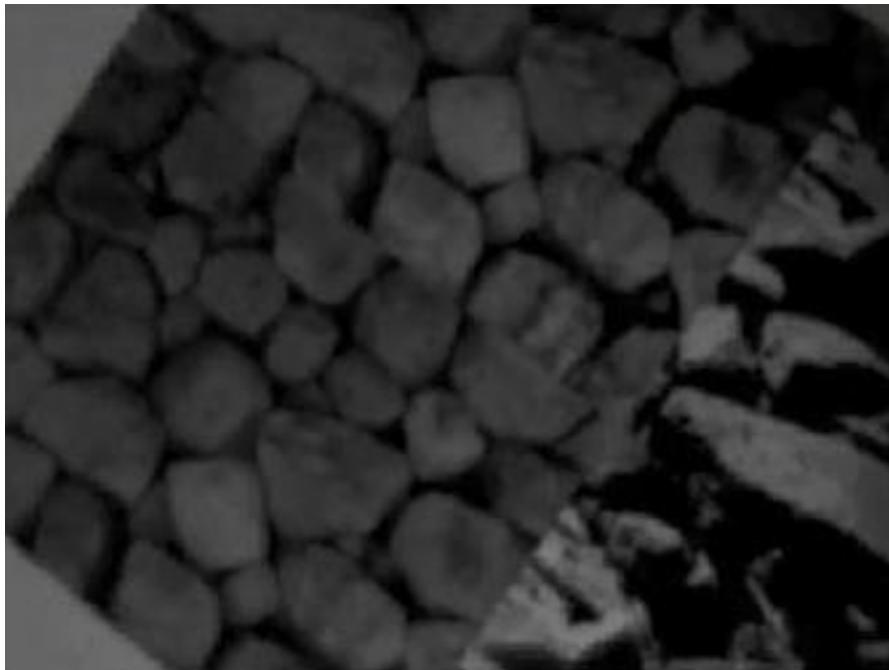
**Sharper**



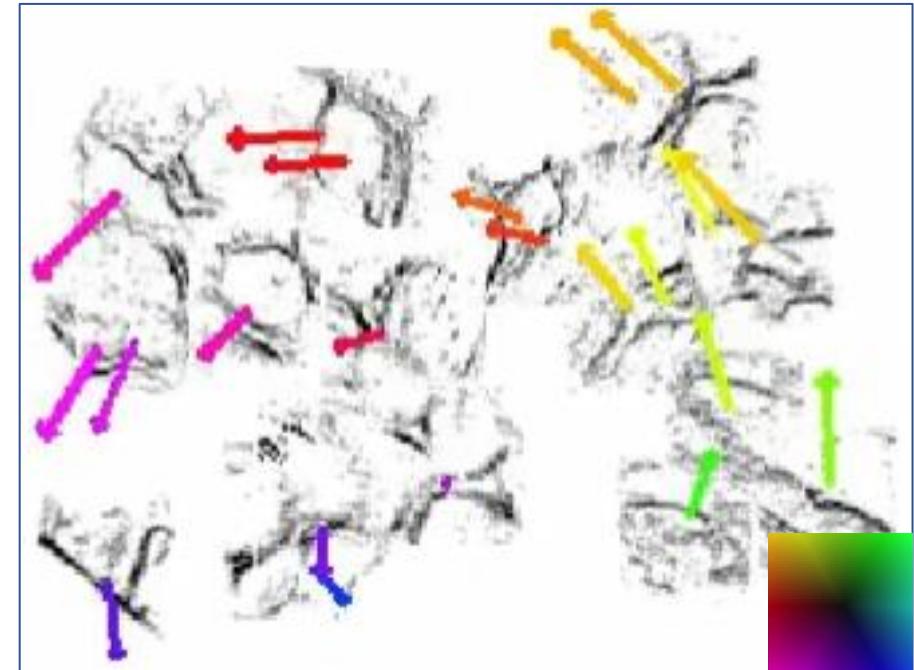
**Sharpest** (with  
motion correction)

# Optical Flow from Motion Compensation

Frames (not used)



Events & Optical Flow

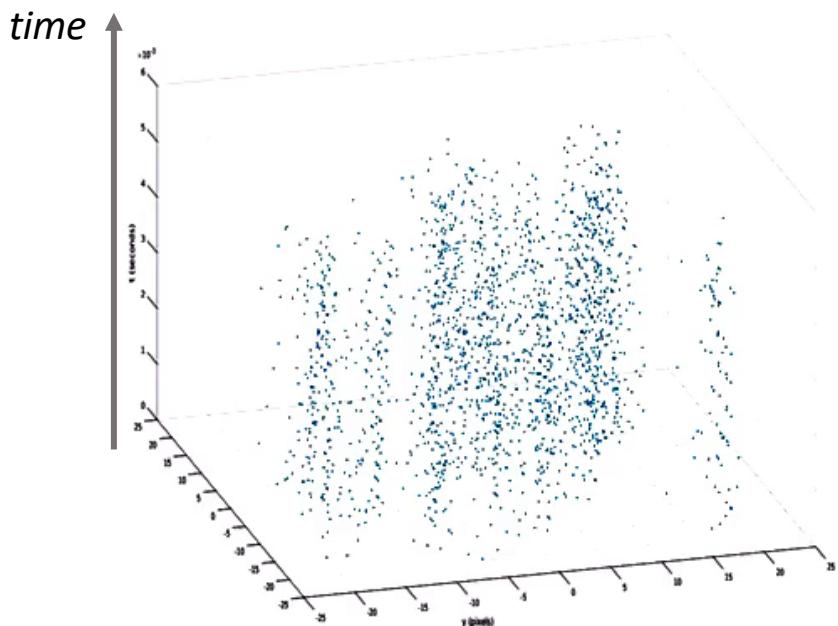


Full flow, at sparse locations (feature-based)

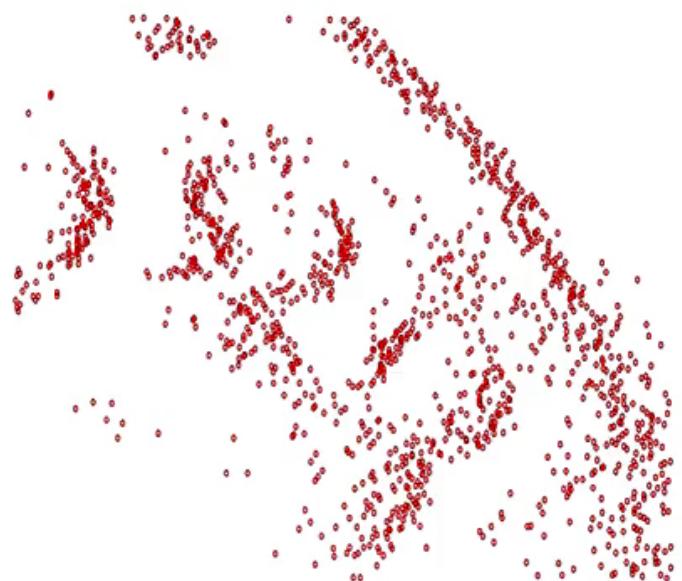
# Optical Flow from Motion Compensation

Interpreting events as **point sets**

Space-time (polarity not used)



1. Warped events  $\{x'_k\}_{k=1}^{N_e}$  (2D point set)  
(like a top-view of space-time)

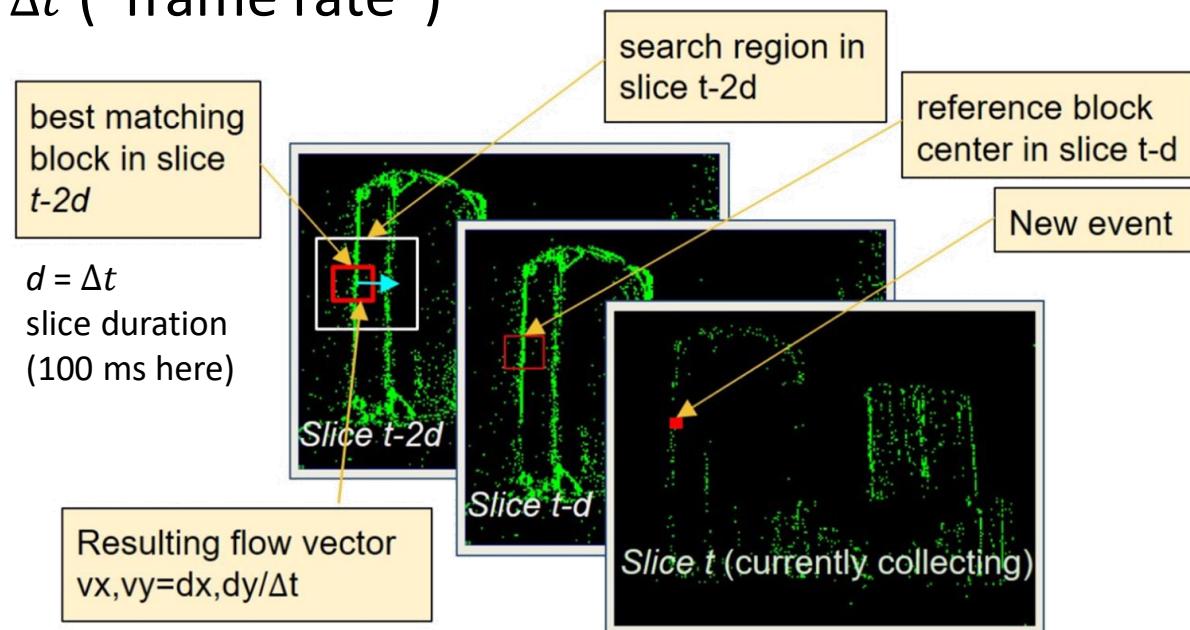


2. Measure the **alignment** of warped events using Euclidean distance between pairs

$$\min_v \sum_{i=1}^n \sum_{k=1}^n \left[ \sum_{j=1}^m r_{ij} r_{kj} \right] \|(x_i - t_i v) - (x_k - t_k v)\|^2$$

# Optical Flow from Block Matching

- **Idea:** reuse video processing technique to estimate motion vectors (“flow”): matching patches / “blocks” (e.g.,  $21 \times 21$  pix)
- Representation: event frames, i.e., **method compares histograms of events** (uncompensated) and finds the best match.
- **Efficient search** for motion vector
- Adaptive slice duration  $\Delta t$  (“frame rate”)
- FPGA implementation



# Optical Flow from Block Matching



Adaptive Time-Slice Block-Matching Optical Flow  
Algorithm for Dynamic Vision Sensors

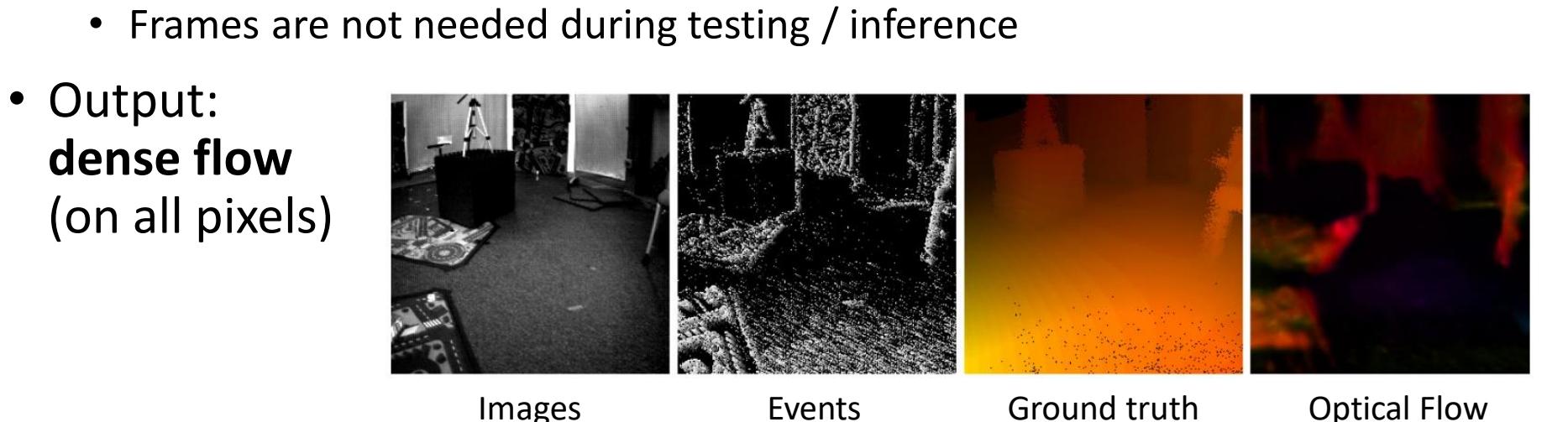
**Min Liu and Tobi Delbrück**  
*Institute of Neuroinformatics*  
*University of Zurich & ETH Zurich*



# Ev-FlowNet

- **Idea:** learn flow from (lots of) data
- **Architecture:**
  - **CNN** (conventional computer vision). U-Net with skip connections...
  - Input (**4-channel**): event frames & time surfaces, split by polarity
- **Self-supervised training using intensity frames**
  - **Loss:** Photometric error (using flow to warp frames) + smoothness (regularizer)

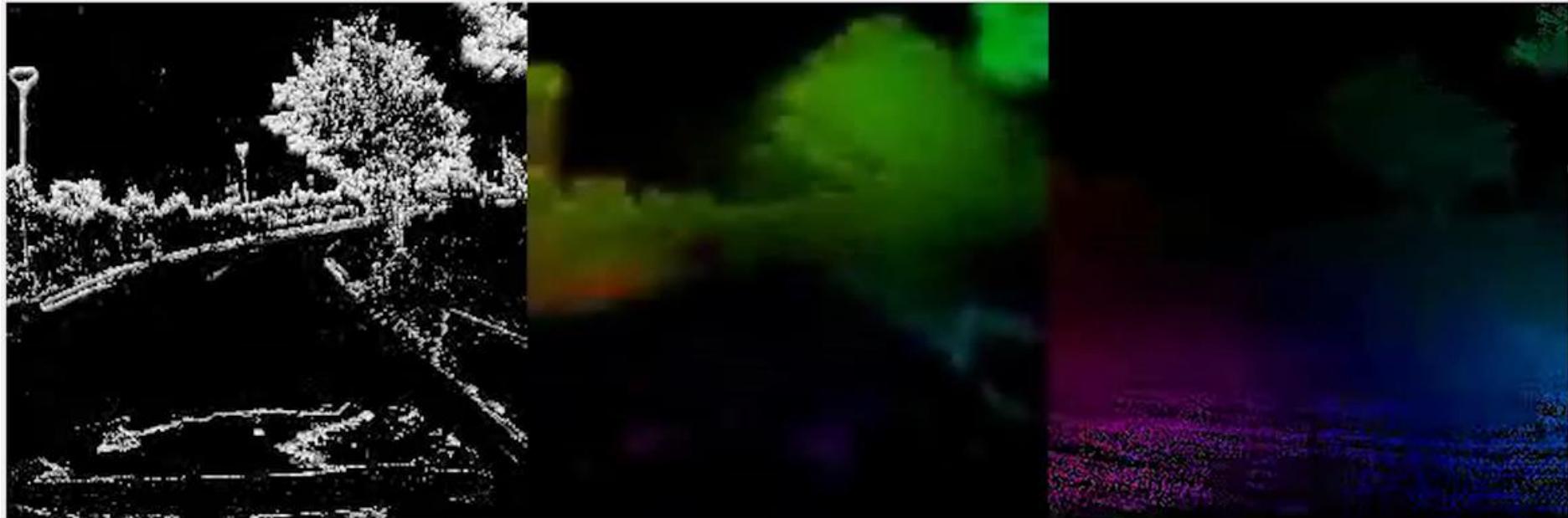
$$\ell_{\text{photometric}}(u, v; I_t, I_{t+1}) = \sum_{x,y} \rho(I_t(x, y) - I_{t+1}(x + u(x, y), y + v(x, y)))$$



# Ev-FlowNet

- Trained & tested on MVSEC dataset (on different sequences)
- **Ground truth** from motion field (depth provided by LIDAR)

Results: outdoor\_day1



Events  
(4-channel input)

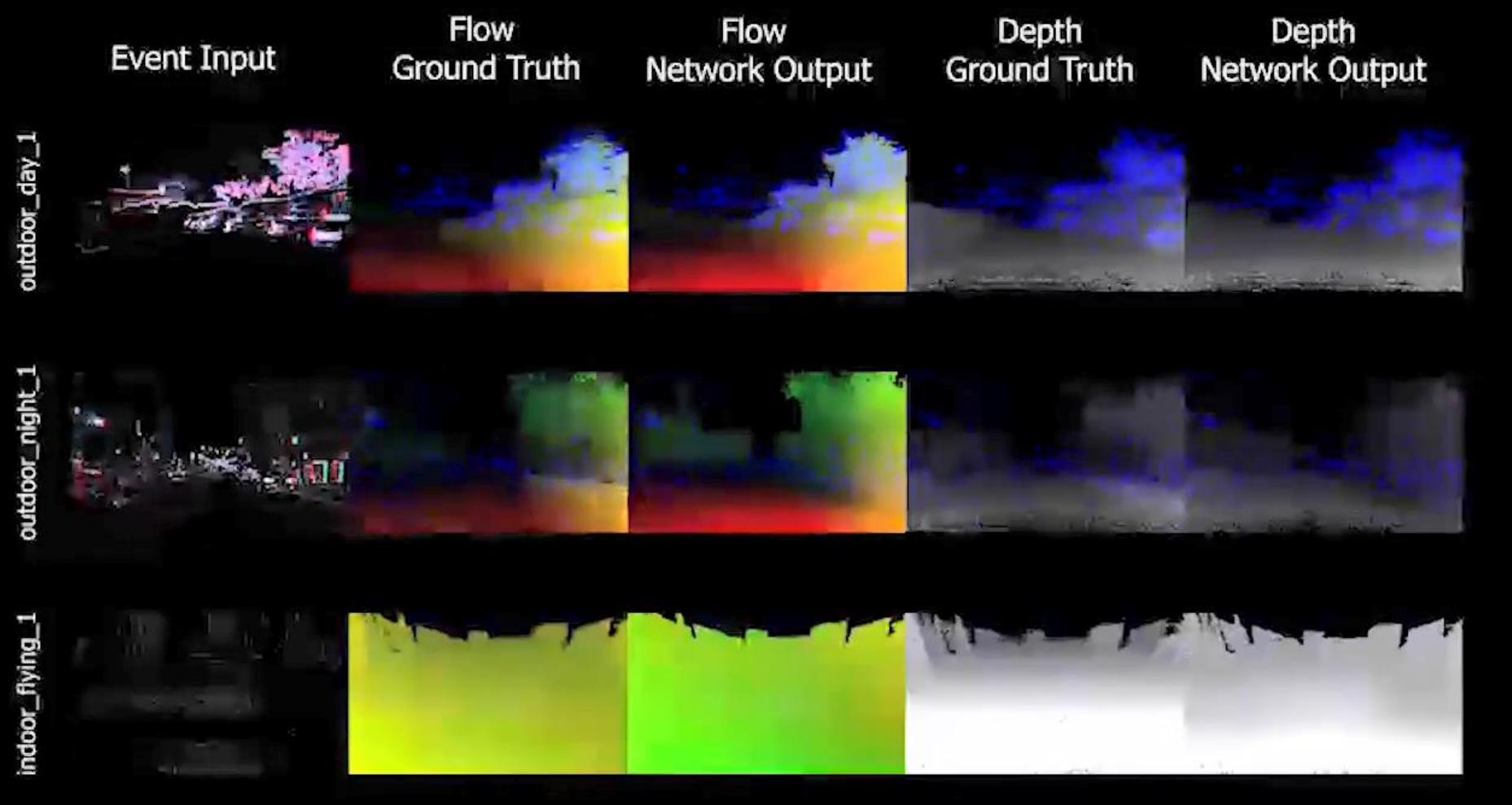
Estimated, dense flow

Ground truth flow

# Learning Structure from Motion

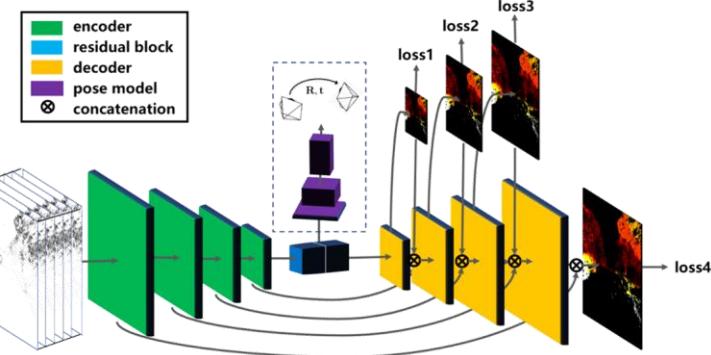
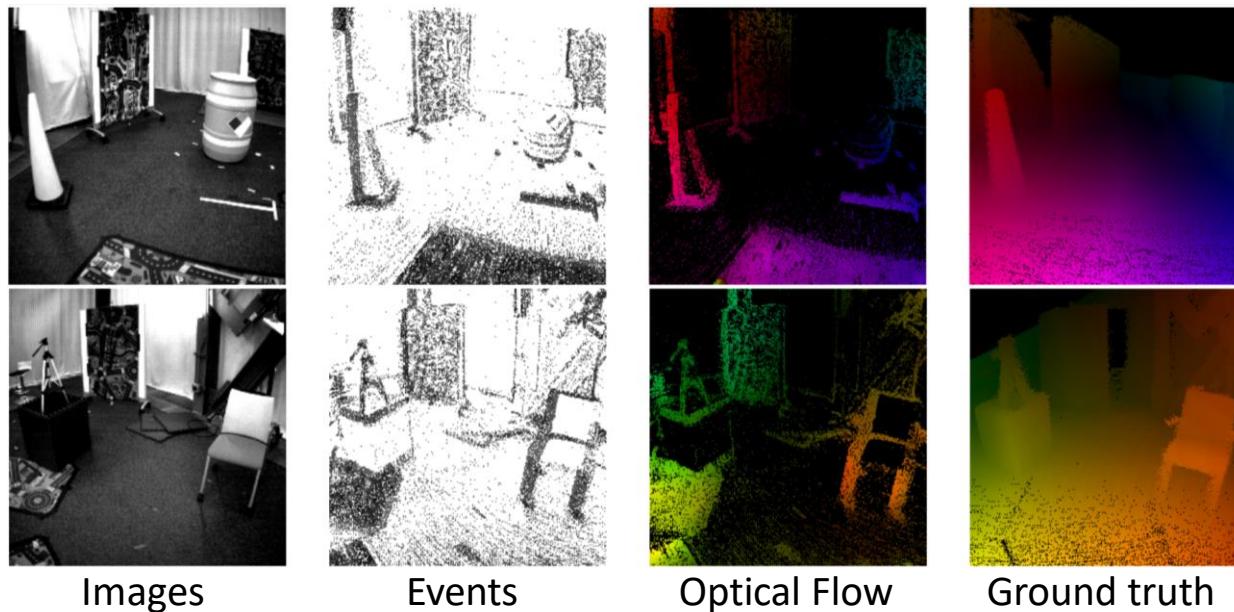
- Input (3-channel): events & average timestamps
- Architecture:
  - Evenly-Cascaded Network (**ECN**) design (small network, lightweight)
  - **Two ECNs**: to predict depth and ego-motion (pose)
- **Unsupervised** training: L1 loss between warped slices using flow
- Output: dense flow (**motion field**) and depth

# Learning Structure from Motion



# Unsupervised CNN-based

- Optical flow NN or Depth + ego-motion (SFM) NN
- Network architecture: U-Net with skip connections and multi-level loss
- Input: voxel grid
- Loss: motion-compensation
- Results:

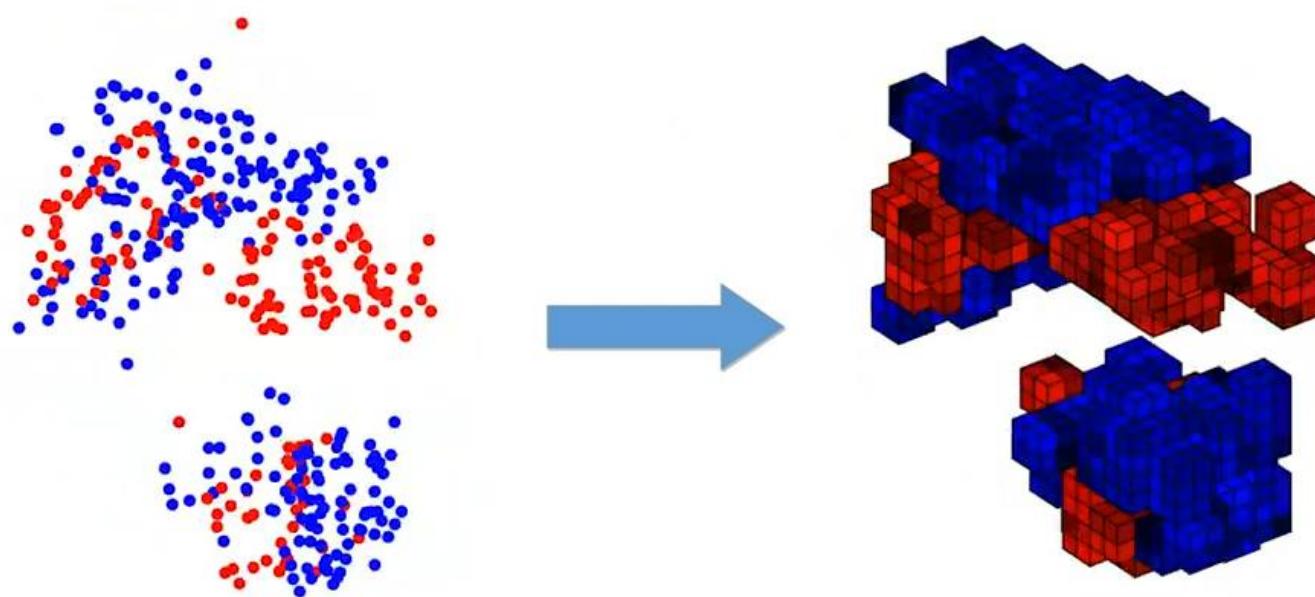


# Unsupervised CNN-based



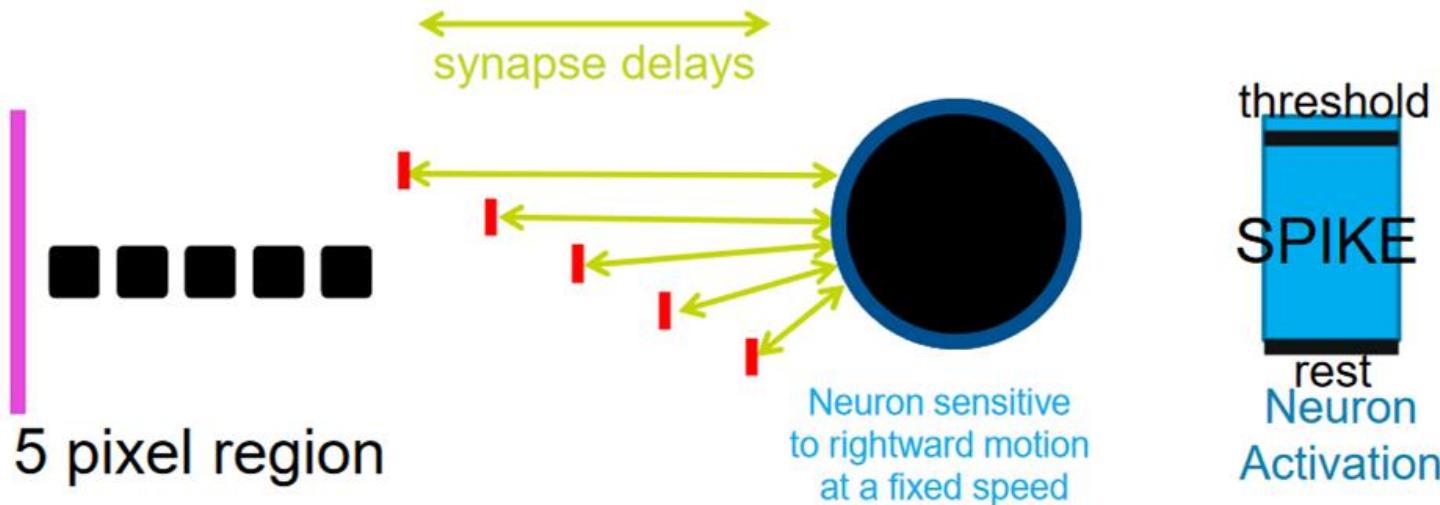
## Input Representation

We propose the **discretized event volume**. Events are inserted into the volume with **trilinear interpolation**, resulting in minimal loss in resolution.



# Optical Flow by SNN-coincidence detection

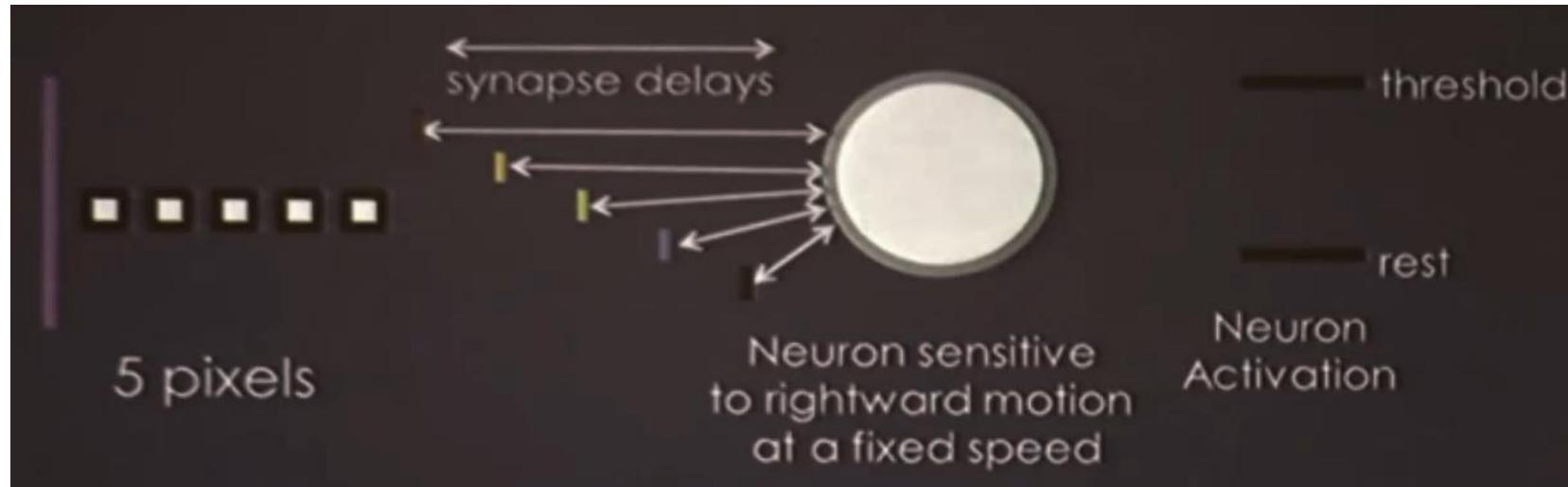
- What is a Spiking Neural Network (**SNN**)?
- Idea: detect a particular flow (direction and speed) by **coincidence detection** of events at a neuron.



- **Delay** each event by a different amount so that events in the receptive field arrive at the same time to the neuron.

# Optical Flow by SNN-coincidence detection

- What is a Spiking Neural Network (**SNN**)?
- Idea: detect a particular flow (direction and speed) by coincidence detection of events at a neuron.

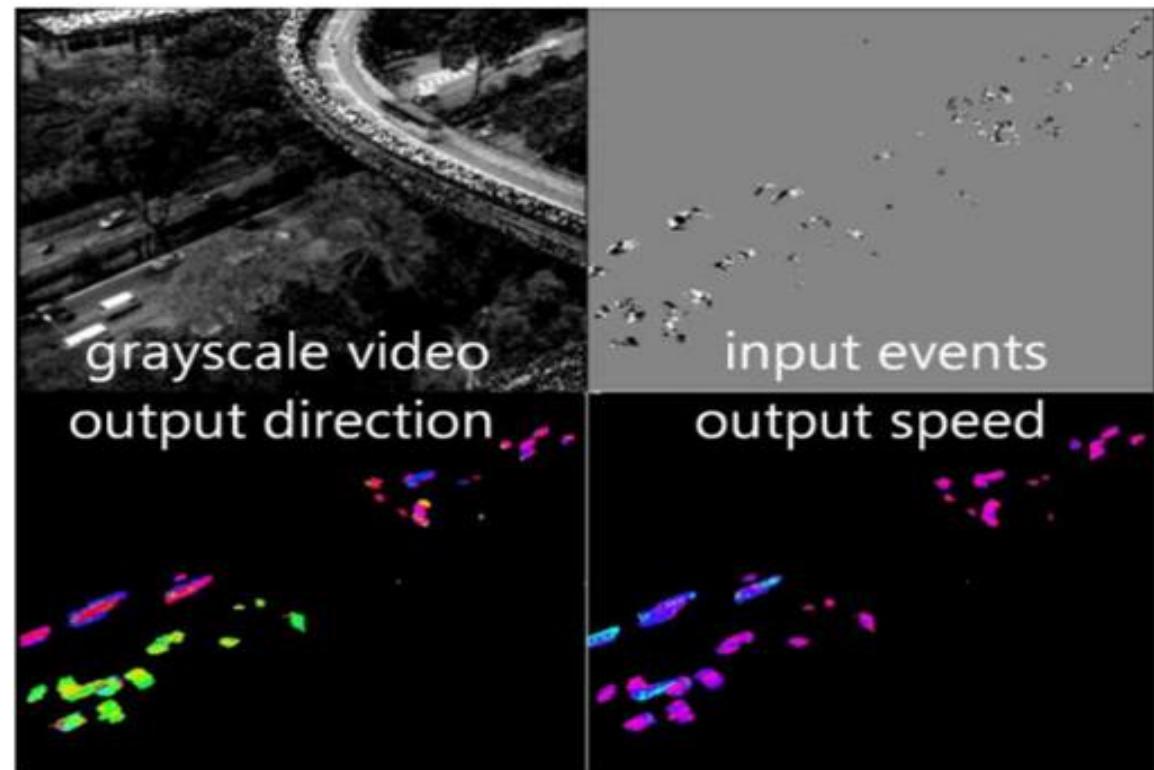


- **Delay** each event by a different amount so that events in the receptive field arrive at the same time to the neuron.

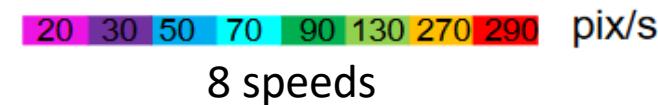
# Optical Flow by SNN-coincidence detection

## Experiments:

- ATIS camera
- Estimate flow in 8 directions and 8 speeds  
= 64 possible flow vectors (64 neurons)
- Manually set the SNN synapses



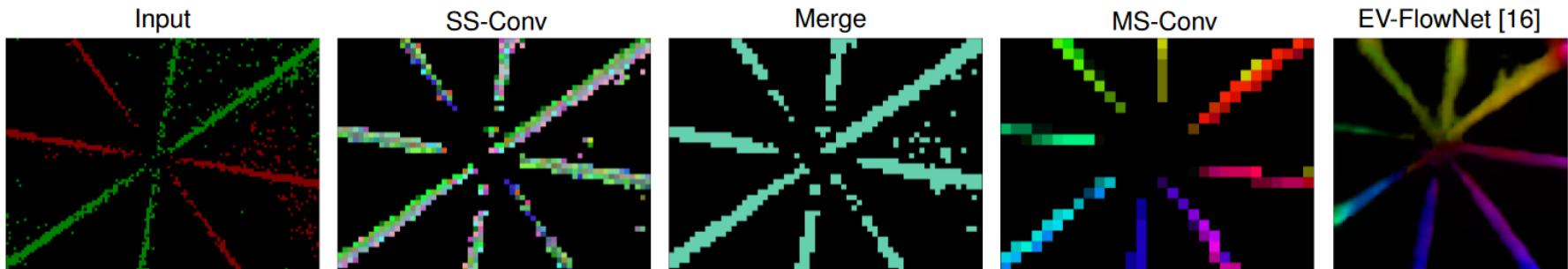
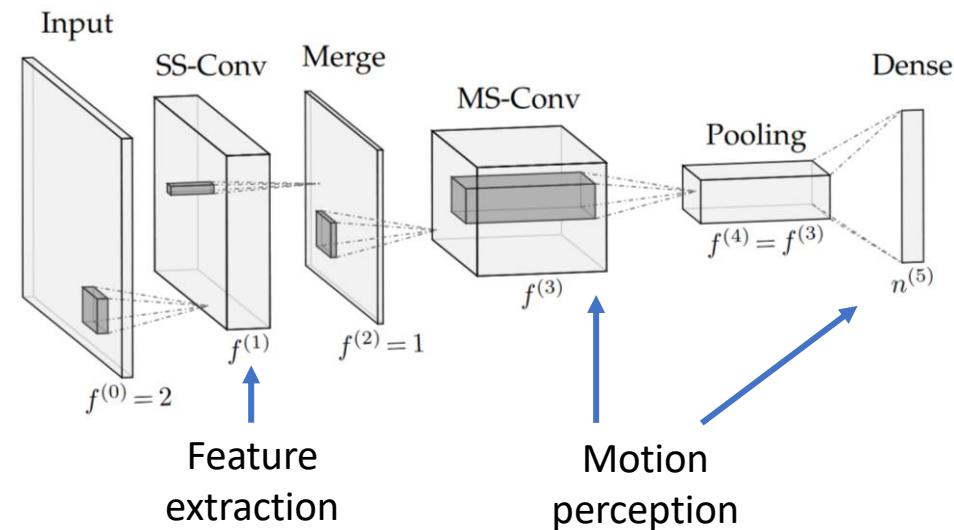
8 directions



8 speeds

# Optical Flow using a Hierarchical SNN

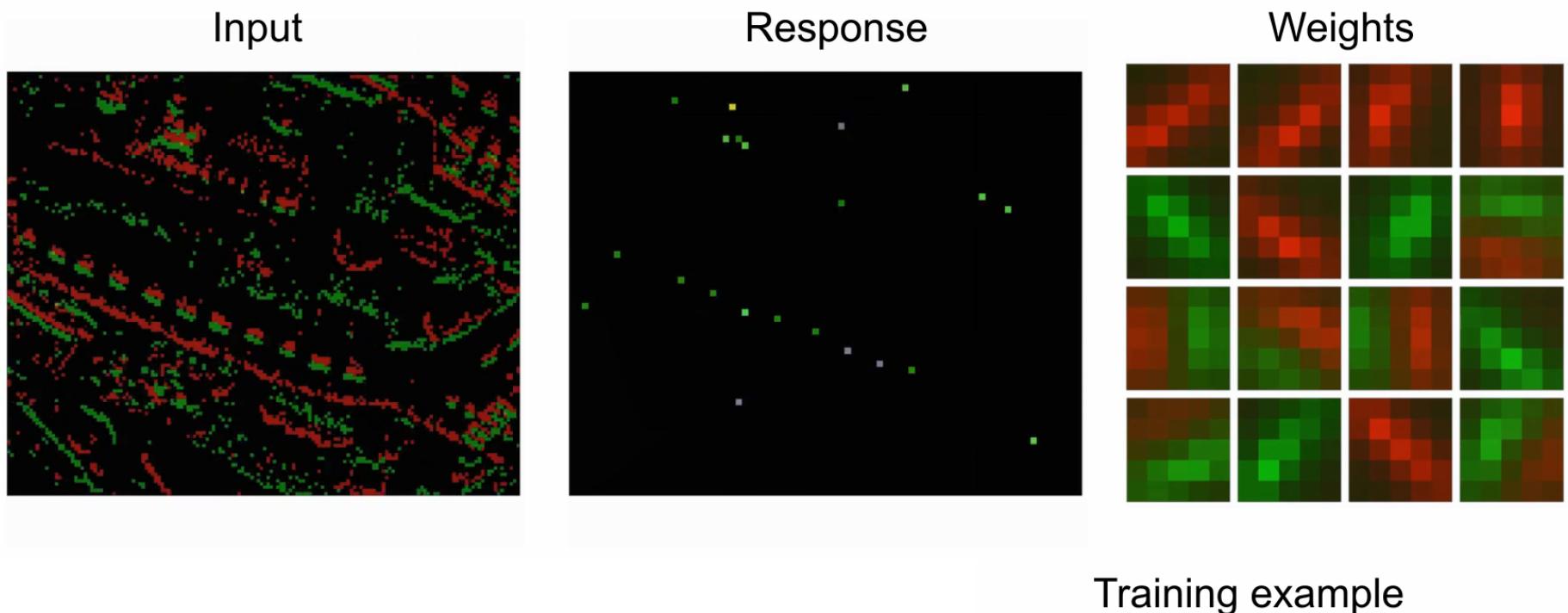
- Architecture: feedforward SNN
- **Connections are learned**
  - Synapses with delays
- Biological properties:  
**feature extraction**, and local  
and global **motion perception**
- All software, not implemented  
in hardware.



# Optical Flow using a Hierarchical SNN

- Result of the SS-Conv and MS-Conv layers

## SS-Conv Layer: Feature Extraction



# How do the methods compare?

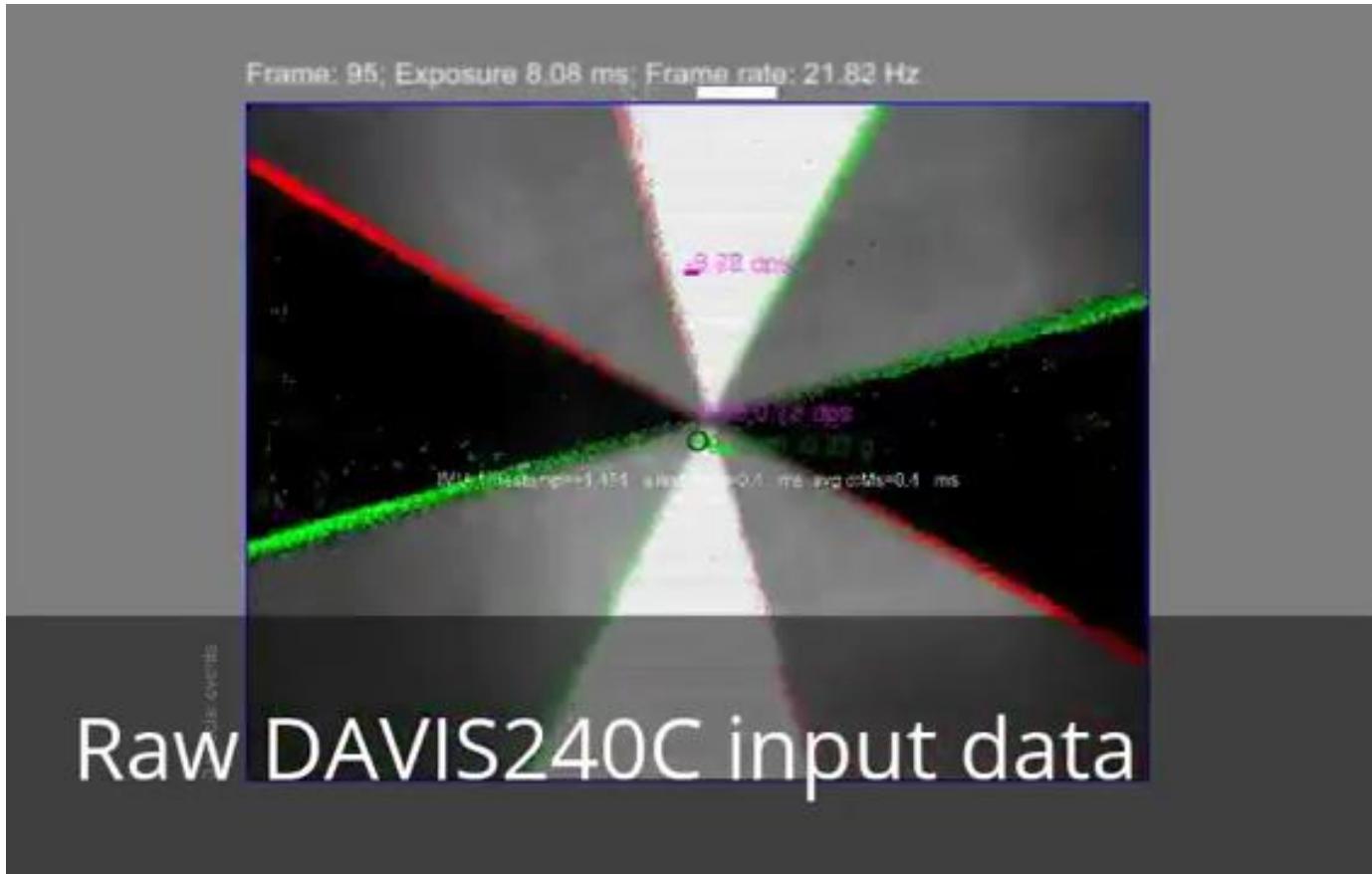
- Hard to tell... there is a lack of comparison benchmark
  - 2016 Comparison by Rueckauer et al. Front. Neuroscience
  - But... there have been many developments since 2016
- De facto standard: use MVSEC dataset (Zhu et al., 2018)
  - “Ground truth” flow from motion field obtained by SLAM-based sensor fusion (LIDAR + motion capture) + manual registration.
  - Take the numbers with a grain of salt.

dt=1 frame	outdoor day1		indoor flying1		indoor flying2		indoor flying3	
	AEE	% Outlier	AEE	% Outlier	AEE	% Outlier	AEE	% Outlier
Ours	<b>0.32</b>	<b>0.0</b>	0.58	<b>0.0</b>	1.02	4.0	0.87	3.0
EV-FlowNet	0.49	0.2	1.03	2.2	1.72	15.1	1.53	11.9
UnFlow	0.97	1.6	0.50	0.1	0.70	1.0	0.55	<b>0.0</b>
ECN <sub>masked</sub>	0.36	0.2	<b>0.20*</b>	<b>0.0*</b>	<b>0.24*</b>	<b>0.0*</b>	<b>0.21*</b>	<b>0.0*</b>

dt=4 frames	outdoor day1		indoor flying1		indoor flying2		indoor flying3	
	AEE	% Outlier	AEE	% Outlier	AEE	% Outlier	AEE	% Outlier
Ours	1.30	9.7	<b>2.18</b>	<b>24.2</b>	<b>3.85</b>	46.8	3.18	47.8
EV-FlowNet	<b>1.23</b>	<b>7.3</b>	2.25	24.7	4.05	<b>45.3</b>	3.45	39.7
UnFlow	2.95	40.0	3.81	56.1	6.22	79.5	<b>1.96</b>	<b>18.2</b>
ECN <sub>masked</sub>	-	-	-	-	-	-	-	-

# 2016 comparison of previous methods

- Nicely compare 4 + 4 + 1 algorithms vs ground truth (from IMU)
  - 4 variations of LK-method, 4 variations of plane-fitting, 1 direction selective
- But it lacks parallax



# References

- Section 4.2 of [Event-based Vision: A Survey](#), TPAMI 2020
- Papers referenced at the bottom of each slide.
- Classical computer vision books for the topic of feature detection and tracking.
  - Optic Flow – Scholarpedia:  
[http://www.scholarpedia.org/article/Optic flow](http://www.scholarpedia.org/article/Optic_flow)
  - D. J. Fleet & Y. Weiss, *Optical Flow Estimation*,  
Handbook of Mathematical Models in Computer Vision, 2005  
<http://www.cs.toronto.edu/~fleet/research/Papers/flowChapter05.pdf>
  - “What Is Optical Flow for?”: Workshop Results and Summary,  
ECCVW 2018.