Technische Universität Berlin

EINSTEIN CENTER Digital Future

# Event-based Robot Vision

Prof. Dr. Guillermo Gallego
Chair: Robotic Interactive Perception

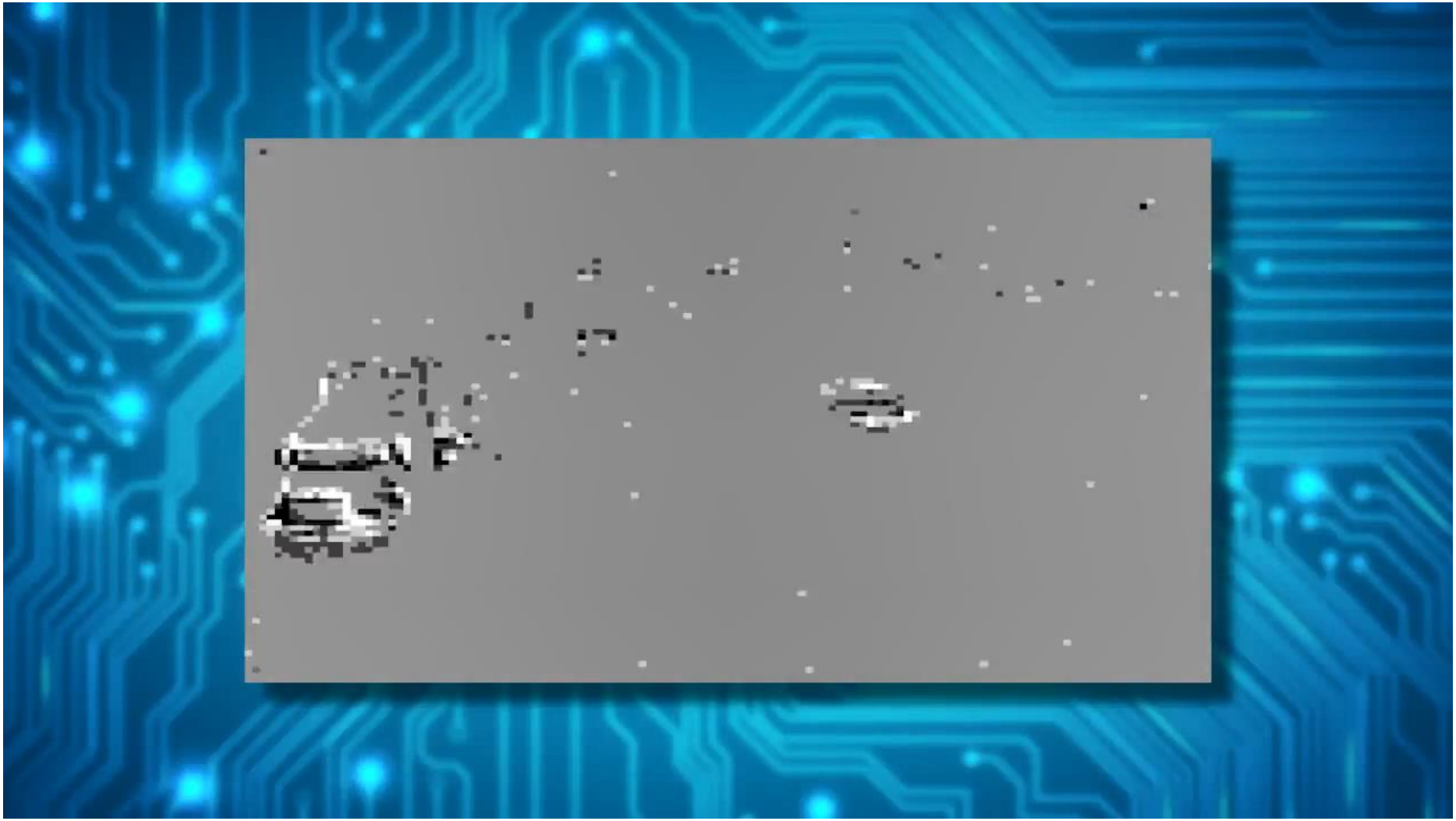
guillermo.gallego@tu-berlin.de

http://www.guillermogallego.es

# Blob / Cluster Tracking

# Blob Tracking for Surveillance

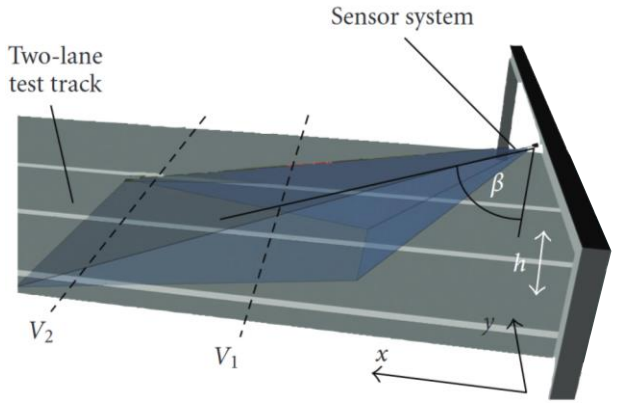
- Application: counting cars and measuring their speed



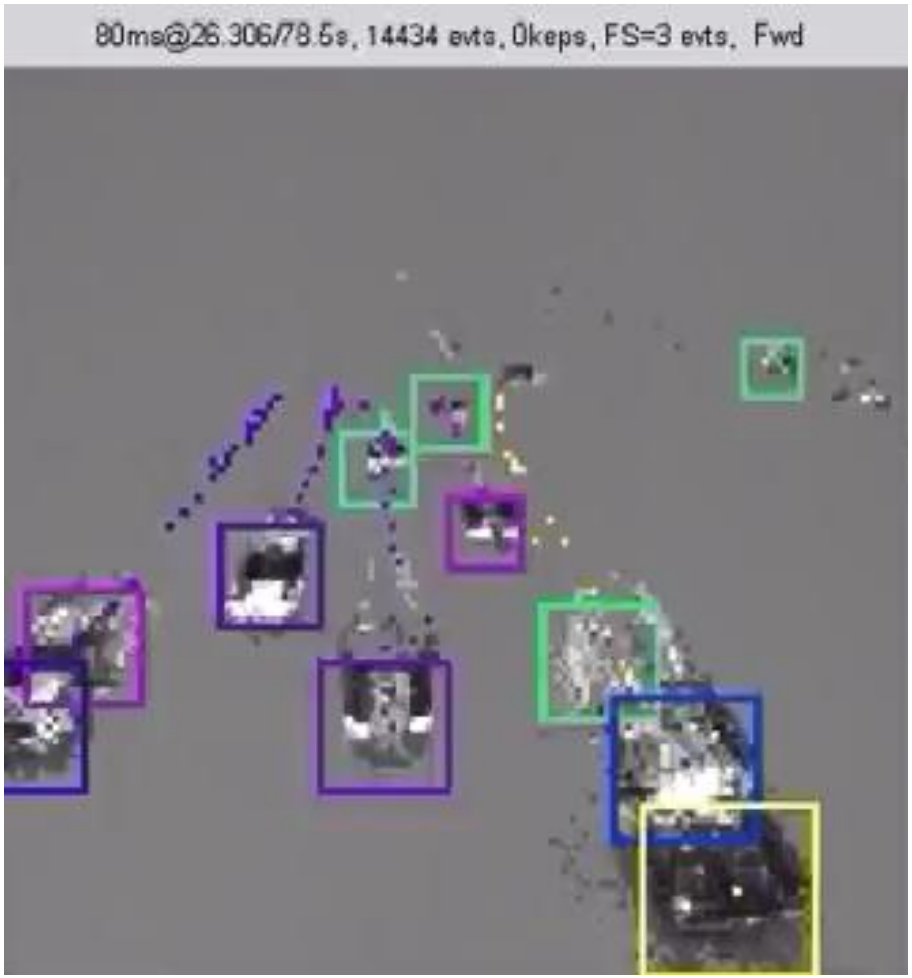
https://youtu.be/Y3izMuXsJMM BrainChip

# Blob Tracking for Surveillance

- Application: counting cars and measuring their speed








Video: T. Delbruck https://youtu.be/4mKlVUz7EIY

Litzenberger et al. ITSC 2006

# Blob / Cluster Tracking

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

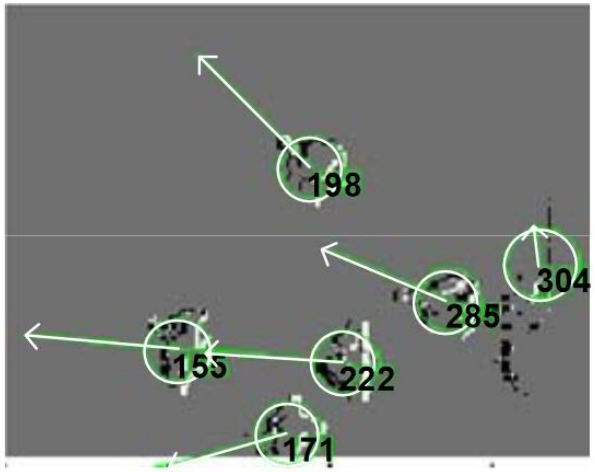

Tracking clusters of events (red and green dots)

https://youtu.be/j3aX3wj72cU

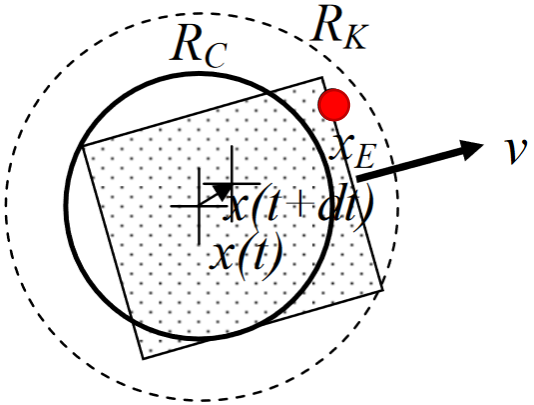
# Blob / Cluster Tracking

- 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?

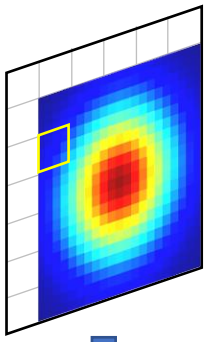
There are **two key ingredients** that are also present in other trackers:

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



Internal state:

$$s(t_{k-1}) = \begin{bmatrix} \boldsymbol{c}(t_{k-1}) \\ \sigma(t_{k-1}) \end{bmatrix}$$

centroid

size

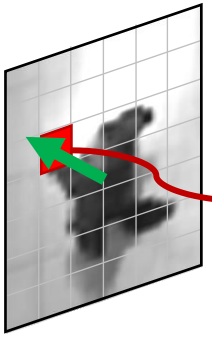


Image sensor
(pixel array)

Event

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

Tracking Algorithm

Output

$\boldsymbol{c}(t_k)$

Update the state of the blob:

$$\boldsymbol{c}(t_k) = \boldsymbol{c}(t_{k-1}) + (\Delta x, \Delta y)$$

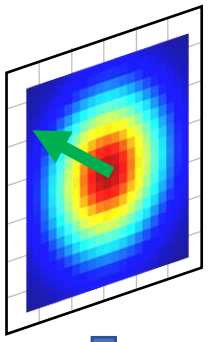
$$\sigma(t_k) = \sigma(t_{k-1}) + \Delta\sigma$$

From current event, "innovation"

# The paradigm: Event-by-event Processing

- Case Study: Blob/Cluster Tracking

Ready for next event:

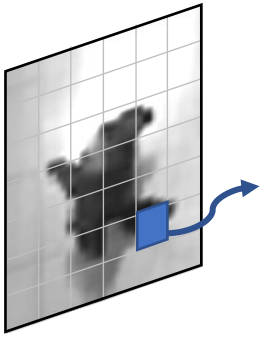


Internal state:

$$s(t_k) = \begin{bmatrix} \boldsymbol{c}(t_k) \\ \sigma(t_k) \end{bmatrix}$$

centroid

size



$e_{k+1}$

Tracking Algorithm

# Video of car-slot racing tracker

- Tracking can be tailored to the problem and very efficient

Delbruck et. al, *Human vs. computer slot car racing using an event and frame-based DAVIS vision sensor*, ISCAS 2015. YouTube

# Blob / Cluster Tracking

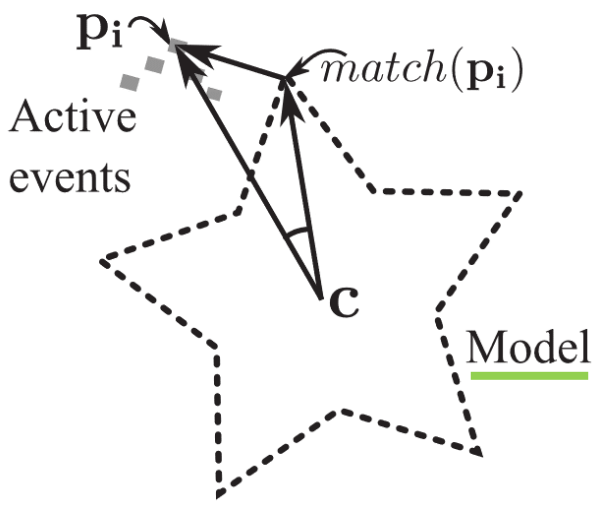
- **Pros**:
  - Very efficient. Do not waste effort on pixels with no events
  - Easy to implement
  - Low latency (event-by-event)

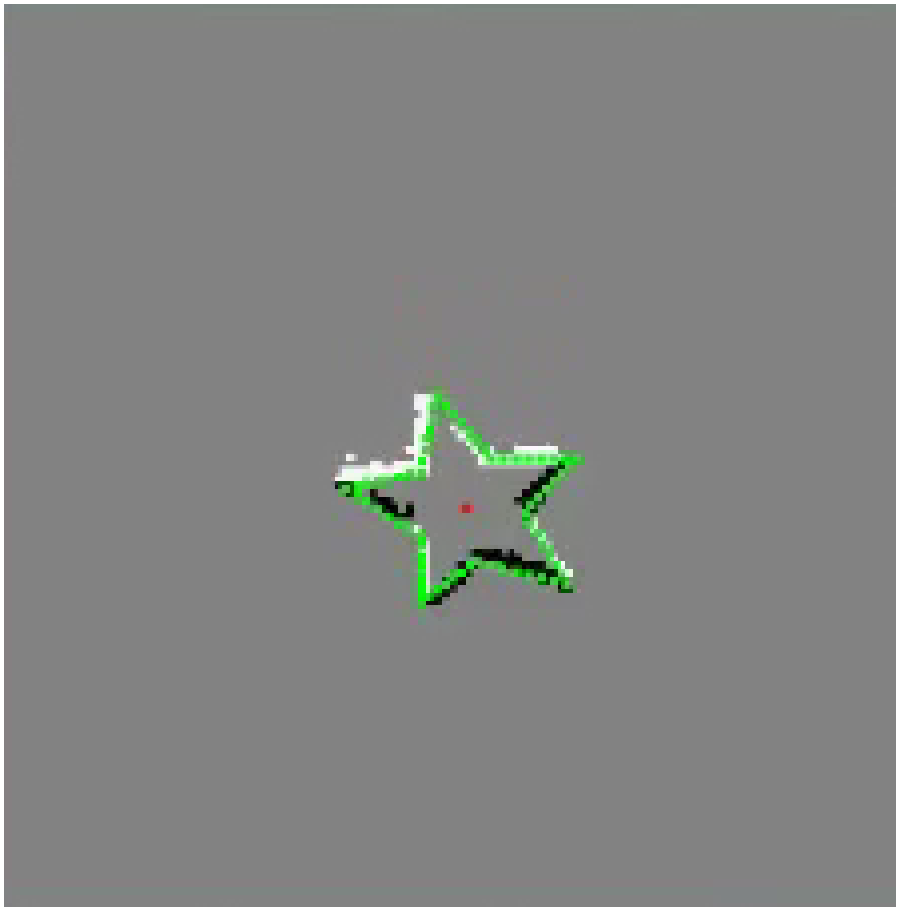
- **Cons**:
  - Rough approximation / simplification to shape.
    There is no concept of object; no clear “boundaries” unless enforced by a constrained scenario
  - Jitter or wiggling hinders accuracy. It should be damped
  - Problems arise if objects cross each other (occlusions…)

# 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)

Ni et al., *Visual Tracking Using Neuromorphic Asynchronous Event-Based Cameras*, Neural Computation, 2015. PDF, YouTube

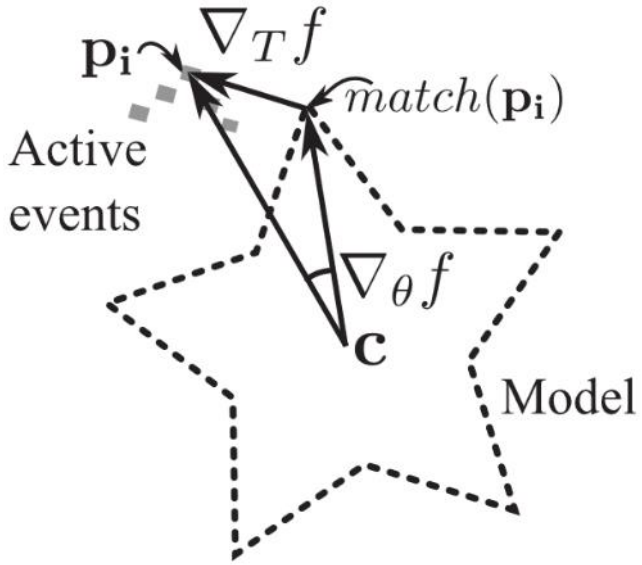
# Event-based Pattern Tracking

- **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 $\boldsymbol{\theta}$: position, orientation, etc.
     - What transformations are enabled? Translations, rotations, affine…

$$\min_{R(\theta)\in SO(2),\mathbf{T}\in\mathbb{R}^2} \sum f_i$$

$$f = \| \mathbf{q}_i - R(\theta)(match(\mathbf{q}_i)) - \mathbf{T} \|^2$$

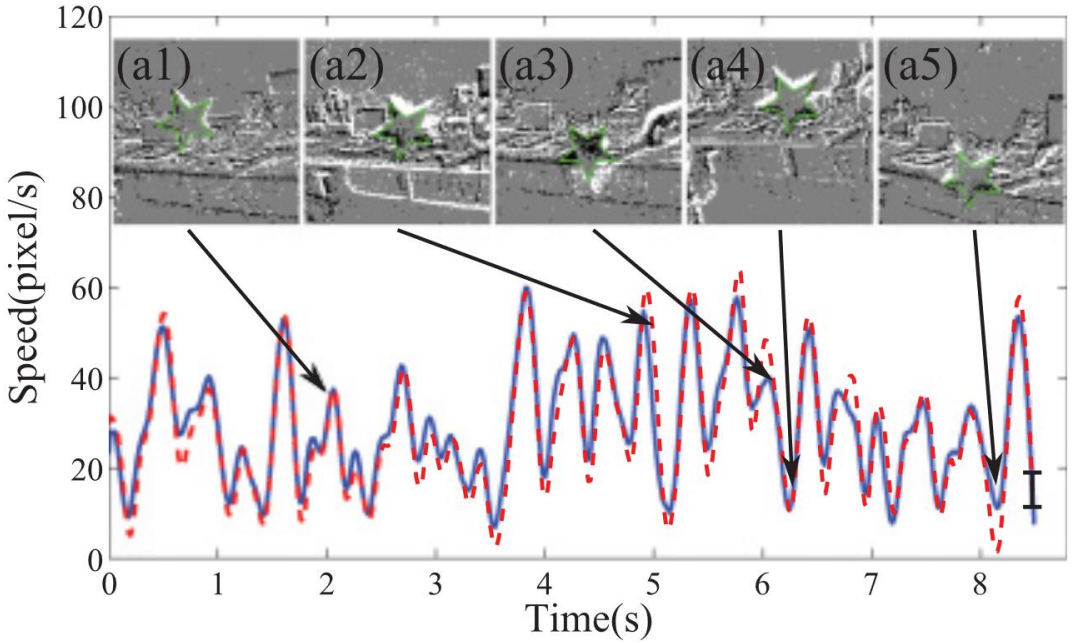
(Euclidean distance)




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

Ni et al., *Visual Tracking Using Neuromorphic Asynchronous Event-Based Cameras*, Neural Computation, 2015. PDF, YouTube

# Event-based Pattern Tracking



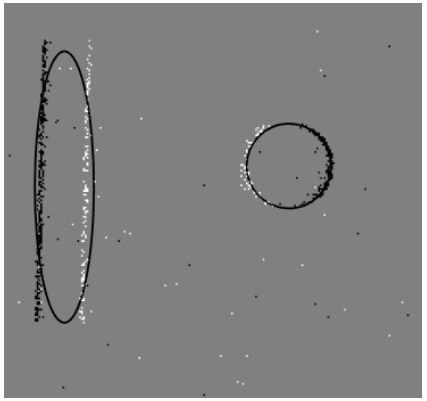

Event frames just for visualization

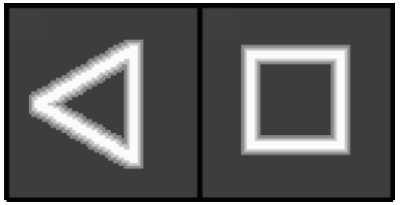
- **Pros**:
  - Works on events directly
  - Very efficient and local computations (better than ICP in tests)
  - Tracking demonstrated even if camera moves
- **Cons**:
  - Shapes are predefined, simple and of high contrast. Limited results on natural scenes.

Ni et al., *Visual Tracking Using Neuromorphic Asynchronous Event-Based Cameras*, Neural Computation, 2015. PDF, YouTube

# Tracking shapes specified by kernel masks

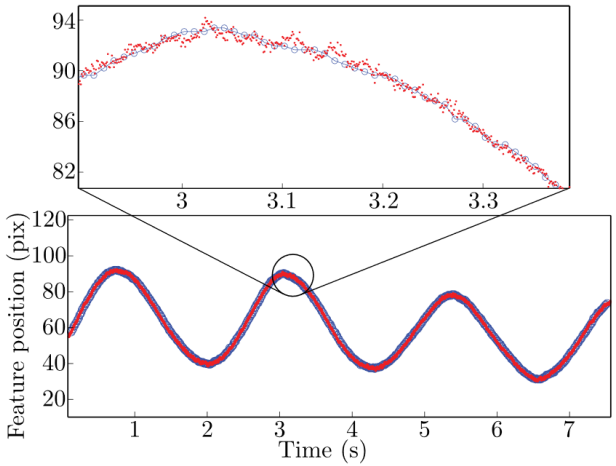
- Geometric Primitives of Objects:
  - **User-defined shapes** (“kernels”). Example: ellipse (pos, size, dir)
  - Considers events as points caused by the kernel as it moves





Kernels

Events

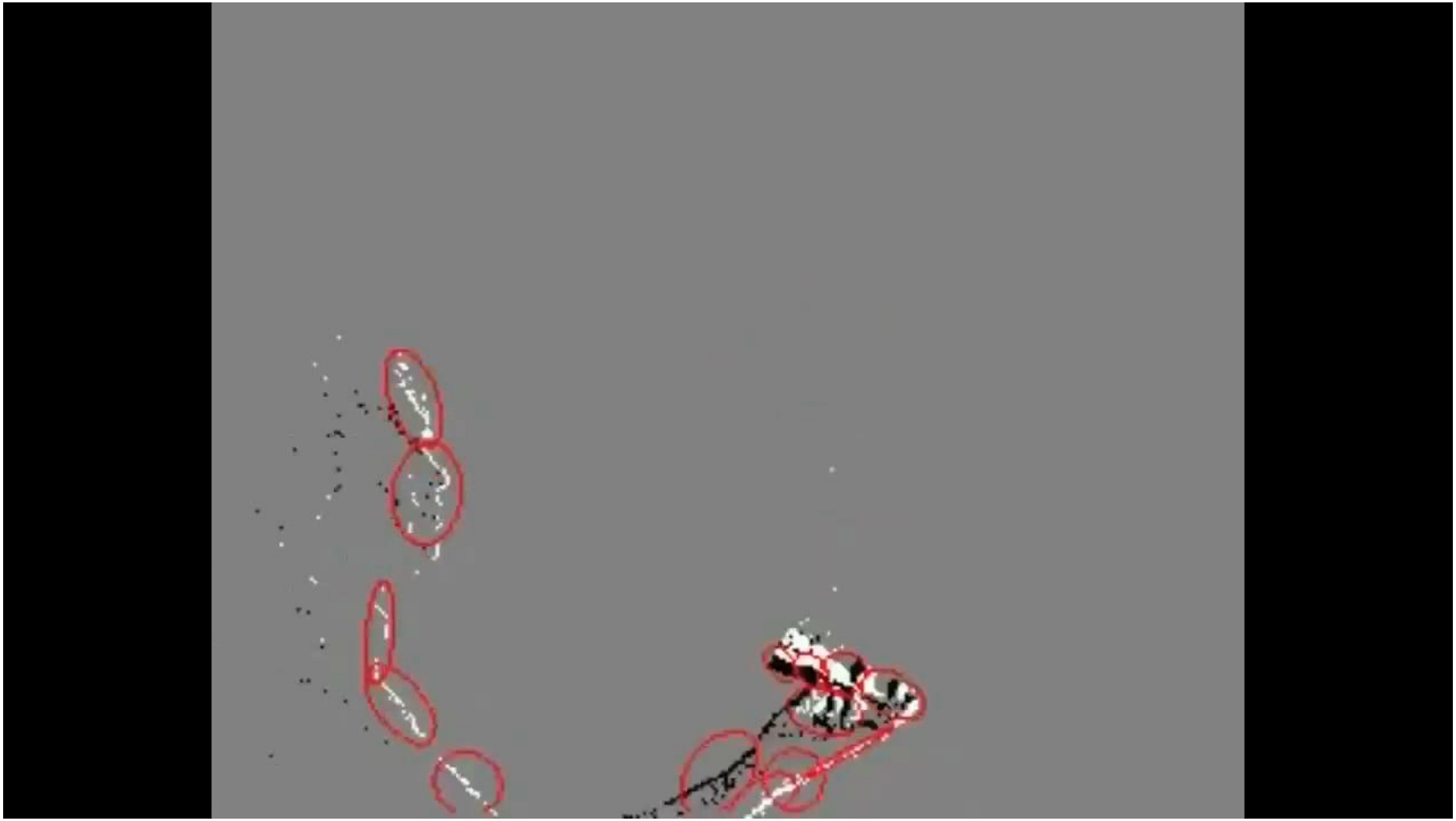



- Method:
  - **Event-by-event** processing. Low-latency & real time
  - Two layers: “activation” (for initialization) and “visible” (tracking)
  - Tracking = update parameters of the kernel. Accuracy: 1 pix
  - Position and orientation are handled separately

Lagorce et al., *Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking*, TNNLS 2015. YouTube

# Tracking shapes specified by kernel masks

- Tracking using oriented Gaussian-like shape primitives

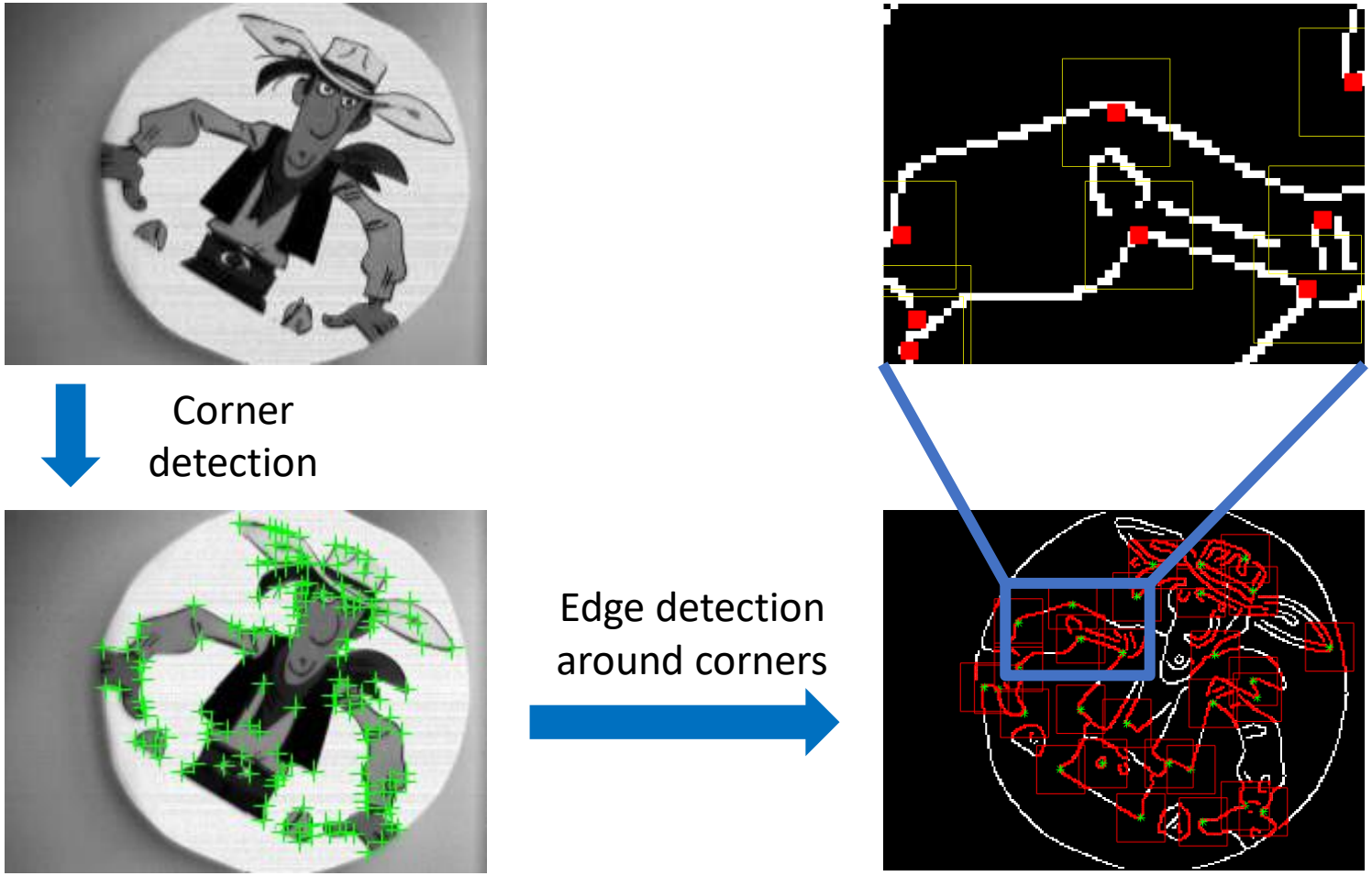


Lagorce et al., *Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking*, TNNLS 2015. YouTube

# 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”

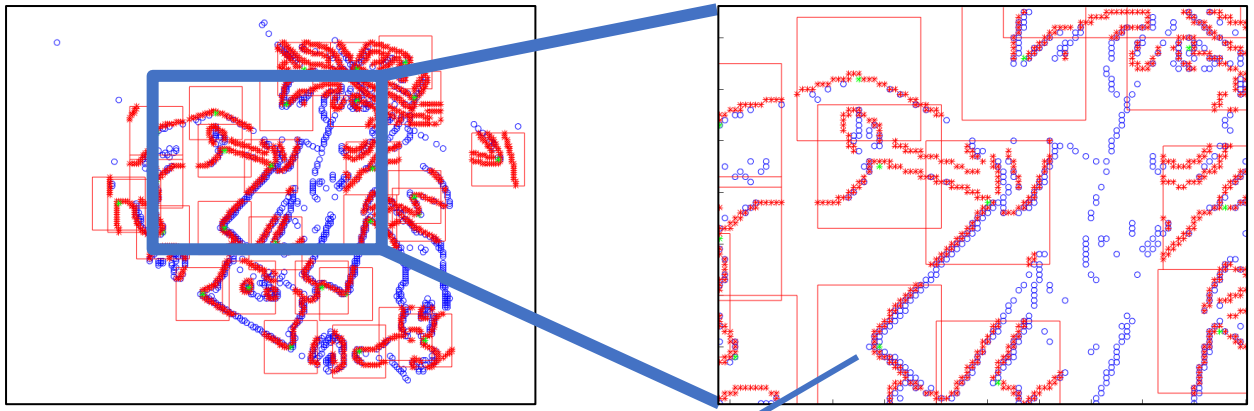


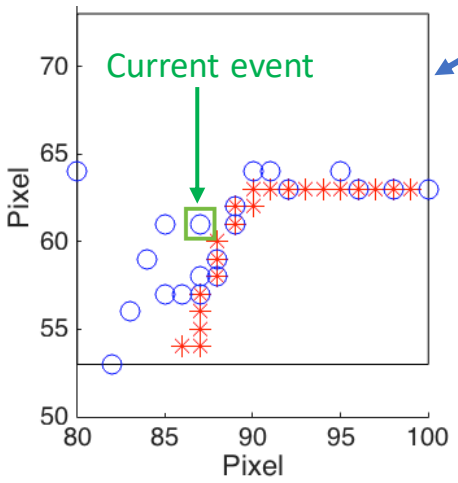

Tedaldi et al., *Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS)*, EBCCSP, 2016. Slide by E. Mueggler

# Feature Tracking using Events

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

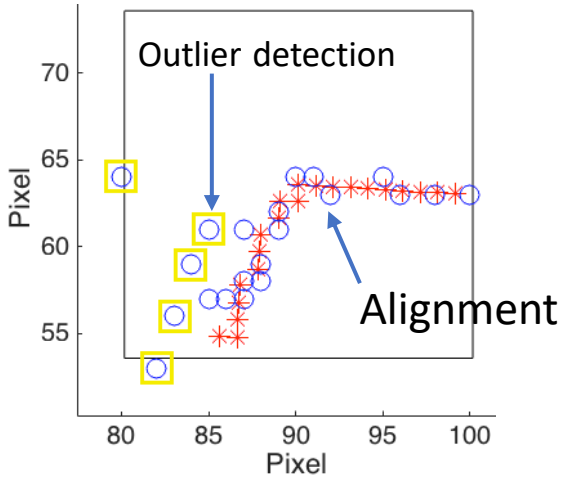
Shape models vs Events






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

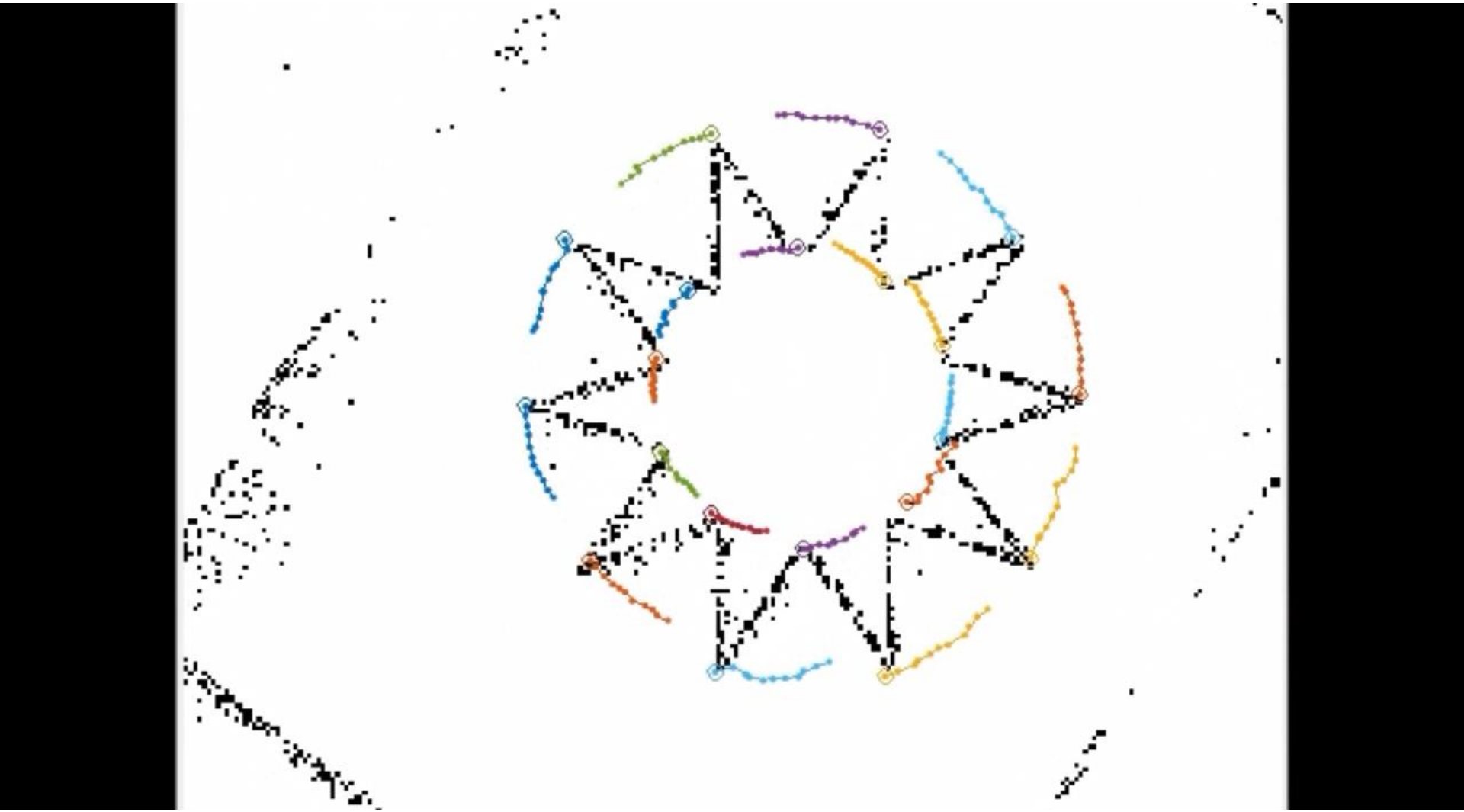
$$\arg\min_{\mathbf{A}} \sum_{(\mathbf{p}_i,\mathbf{m}_i)\in \text{Matches}} \|\mathbf{A}(\mathbf{p}_i) - \mathbf{m}_i\|^2$$




Tedaldi et al., *Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS)*, EBCCSP, 2016.

Slide by E. Mueggler

# High Speed Tracking. Even with just two edges!

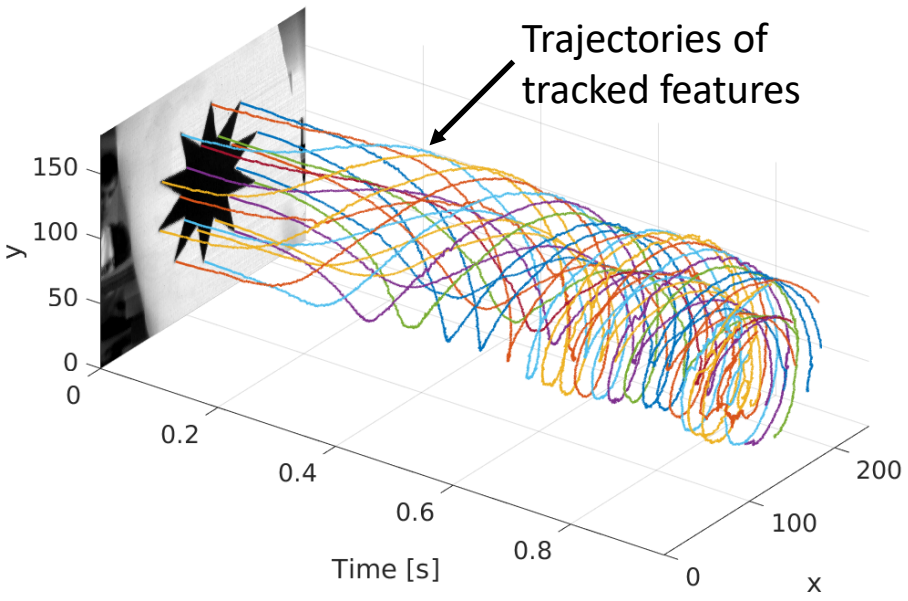


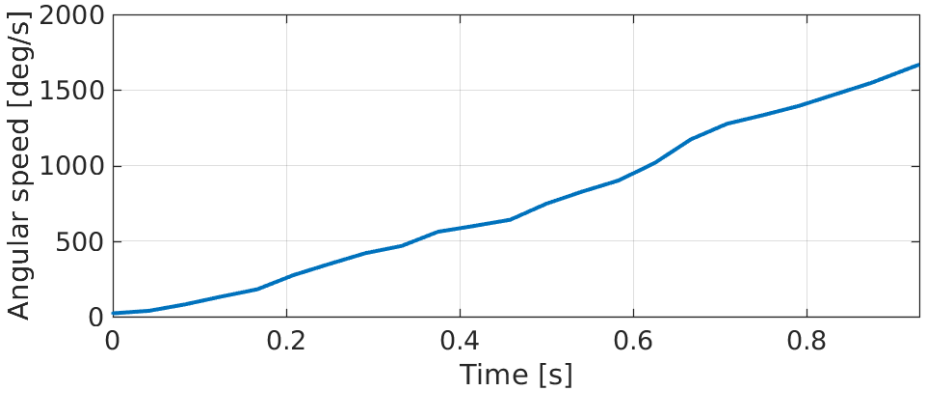
Tedaldi et al., *Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS)*, EBCCSP, 2016. https://youtu.be/nglfEkiK308

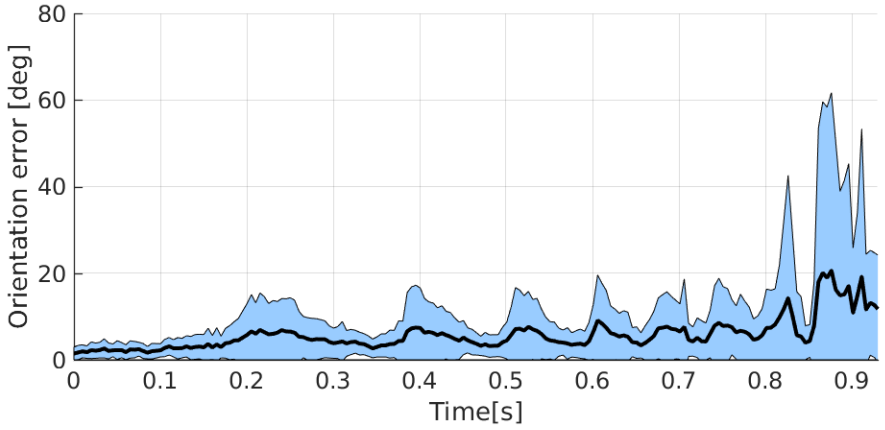
# Feature Detection & Tracking with the DAVIS

Results on Rotating Star

- Acceleration from rest to 1.600°/s
- 1.800 pixels/s on image plane
- Mean error: 6.3°




Tedaldi et al., *Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS)*, EBCCSP, 2016.

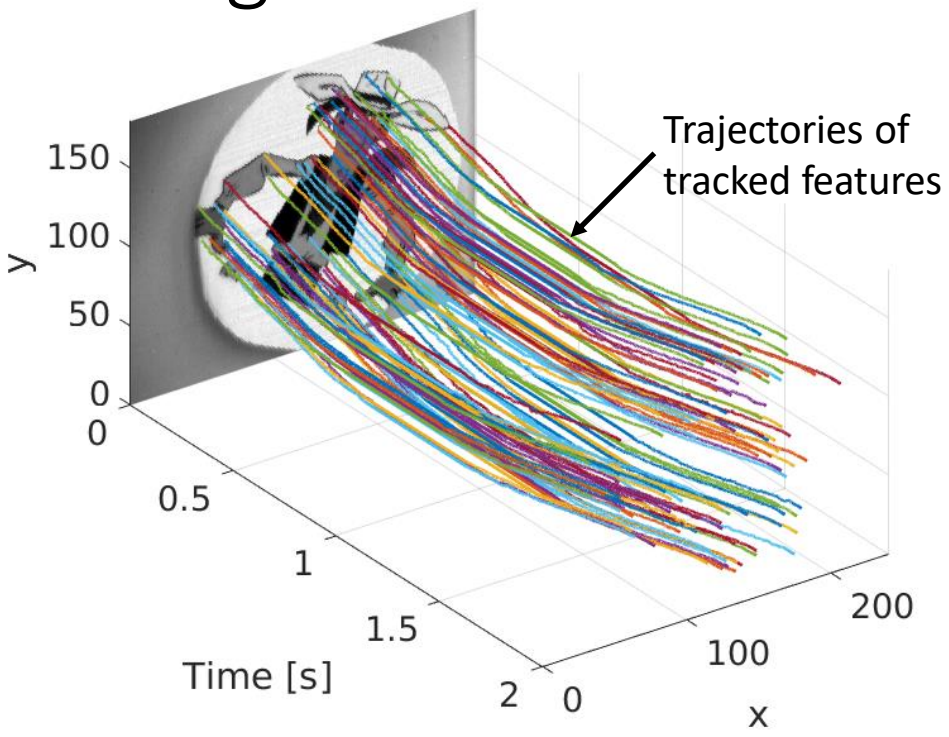
Slide by E. Mueggler

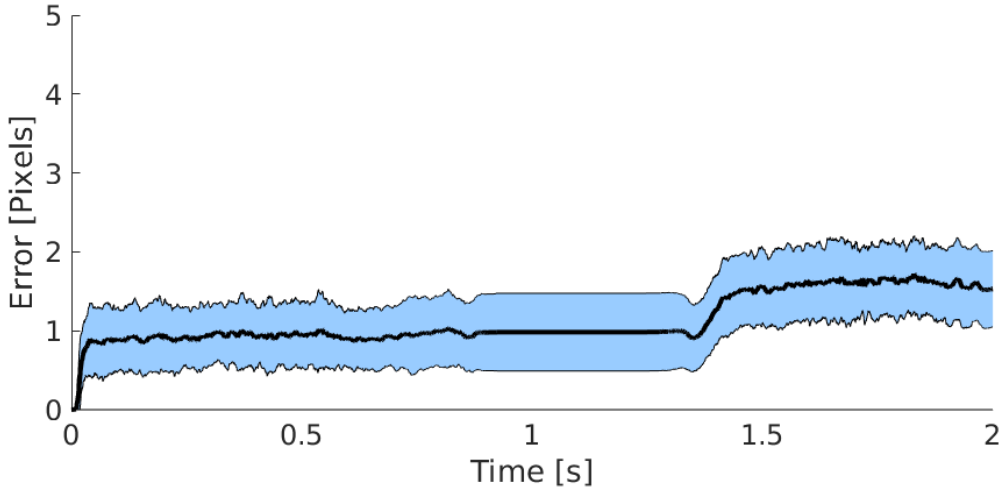
# Feature Detection & Tracking with the DAVIS

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




Tedaldi et al., *Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS)*, EBCCSP, 2016.

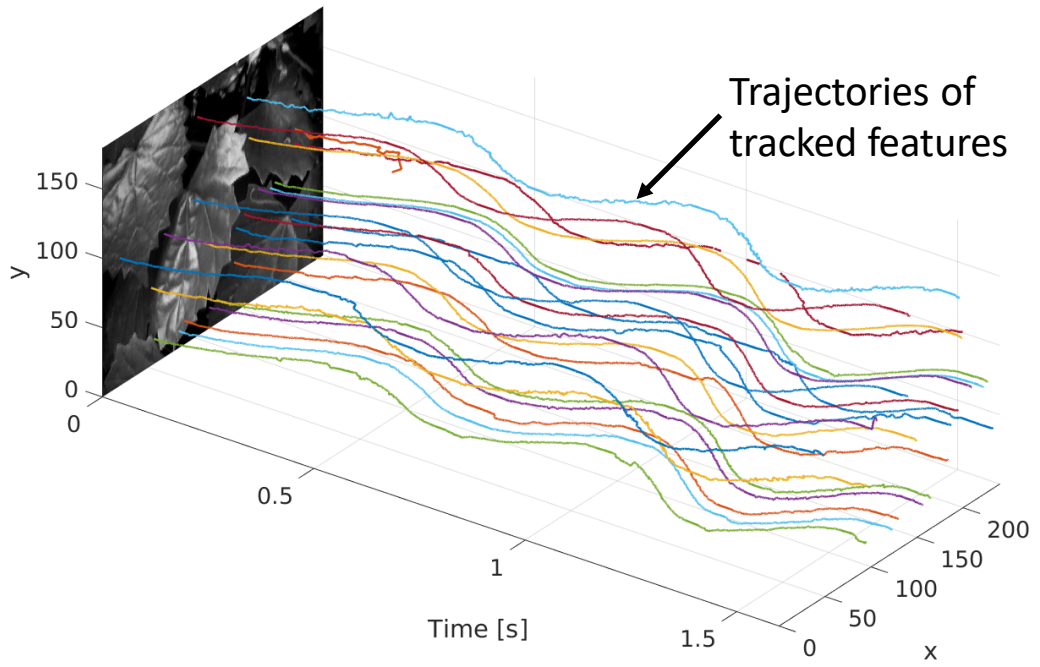
Slide by E. Mueggler

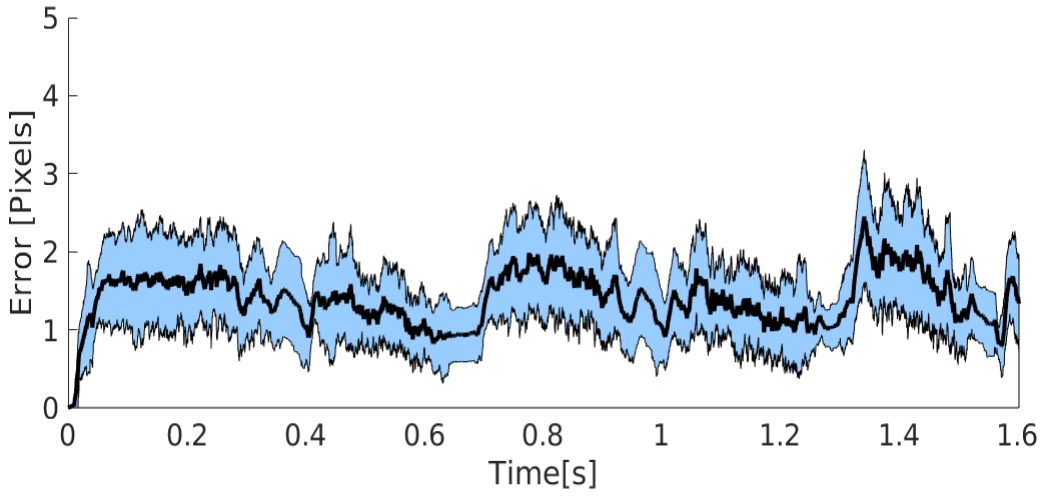
# Feature Detection & Tracking with the DAVIS

Results on Leaves

- Natural scene
- Oscillatory motion
- 20 features
- Mean error: 1.48 pixels

- Ground truth obtained from frame-based feature tracking




Tedaldi et al., *Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS)*, EBCCSP, 2016.

Slide by E. Mueggler

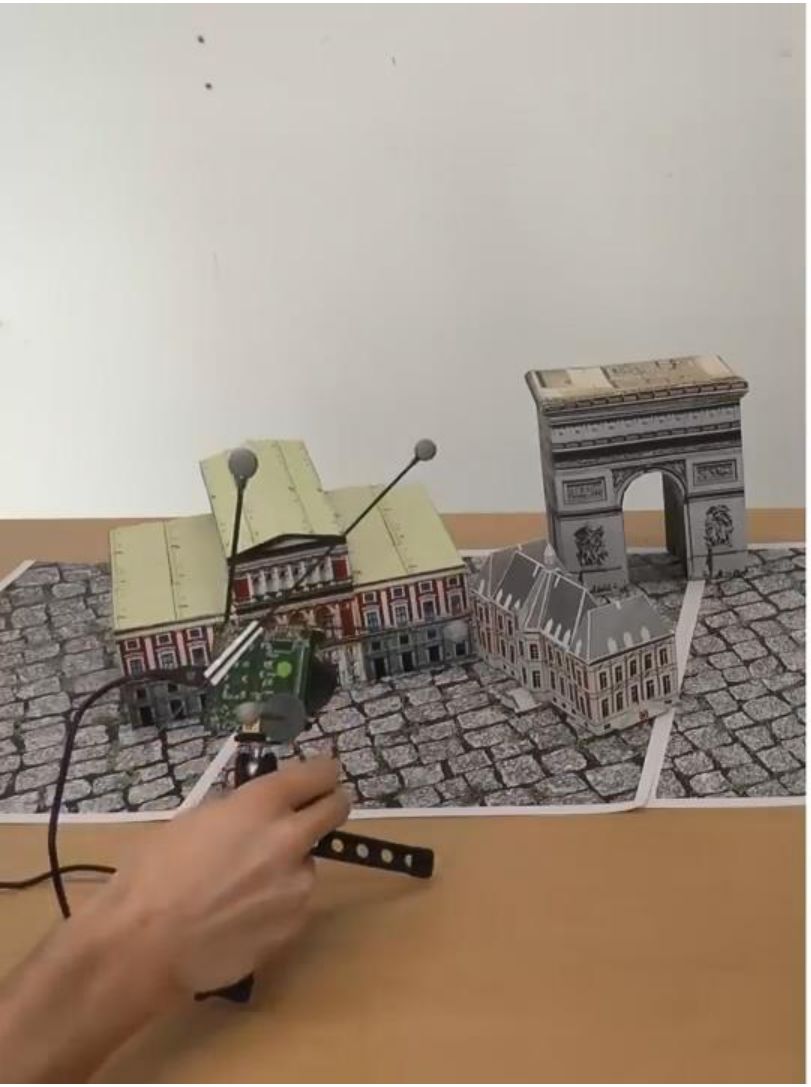
# Feature Detection & Tracking with the DAVIS

- **Pros**:
  - Works on the **events directly** (no event frames) → Efficiency: only pixels with events are processed
  - Allows for **high-speed** tracking (> 1 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

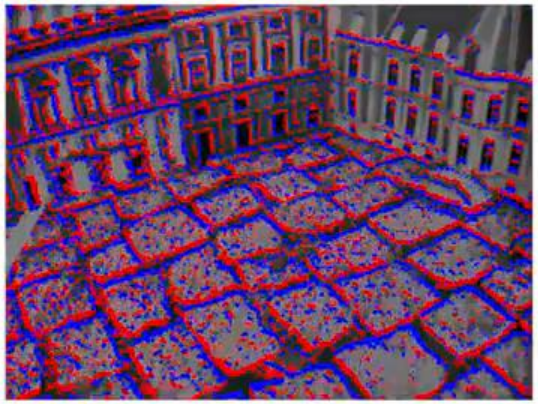
Tedaldi et al., *Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS)*, EBCCSP, 2016. https://youtu.be/nglfEkiK308

Kueng et al., *Low-Latency Visual Odometry using Event-based Feature Tracks*, IROS, 2016. https://youtu.be/RDu5eldW8i8

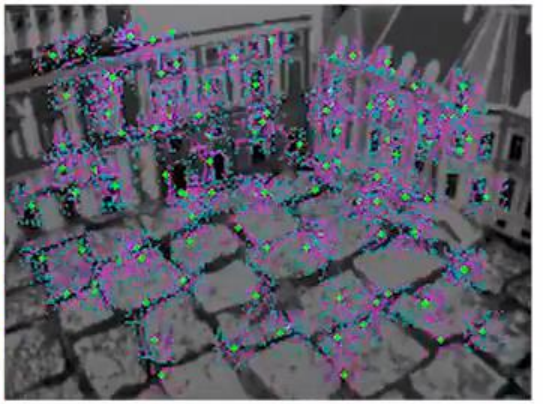
# Application to Visual Odometry



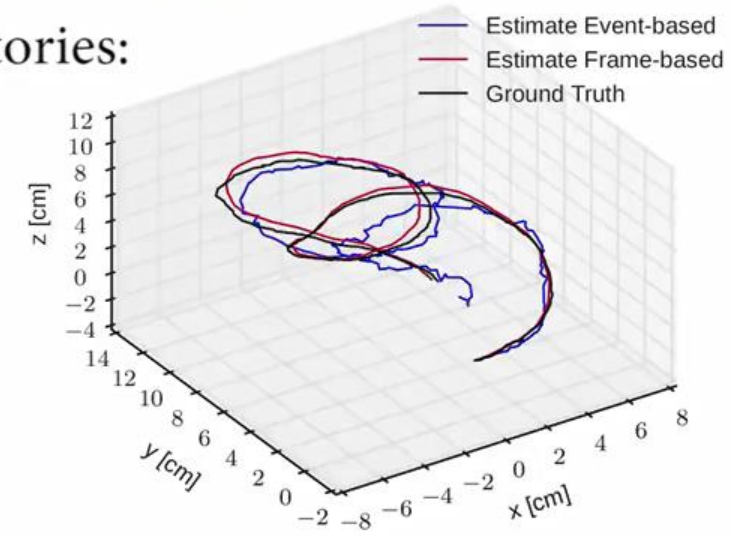
Raw Data:



Event-based Features:



3D Trajectories:

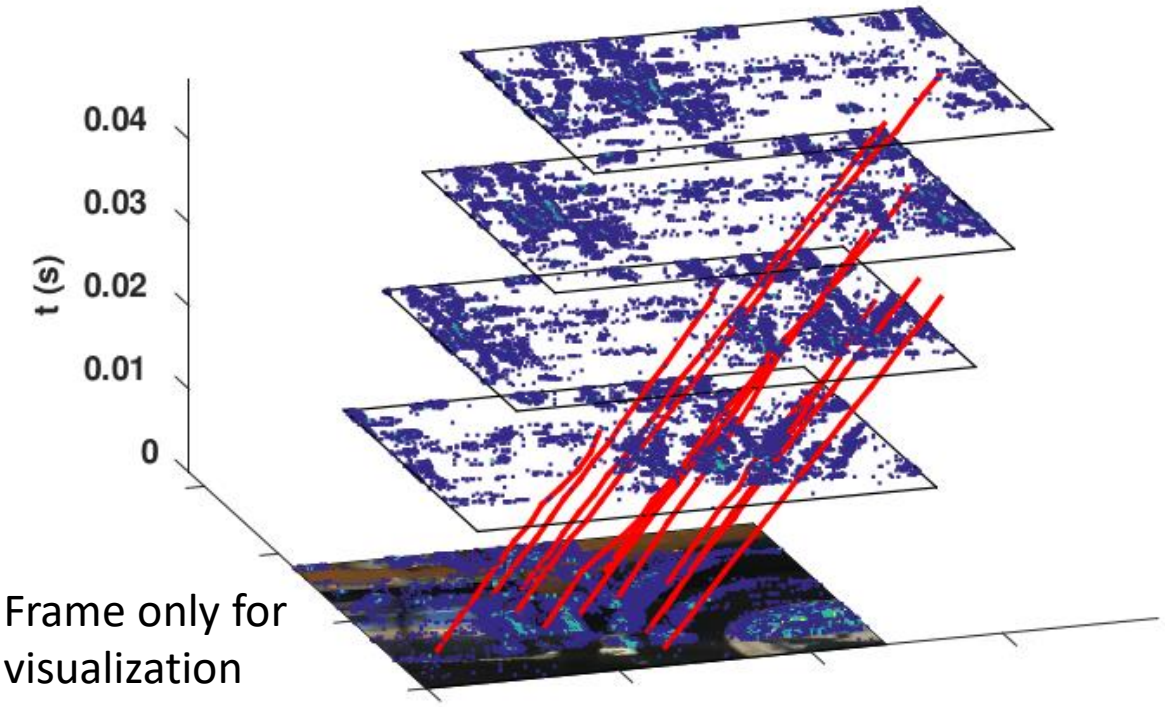



Kueng et al., *Low-Latency Visual Odometry using Event-based Feature Tracks*, IROS, 2016. https://youtu.be/RDu5eldW8i8

# 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




Frame only for visualization

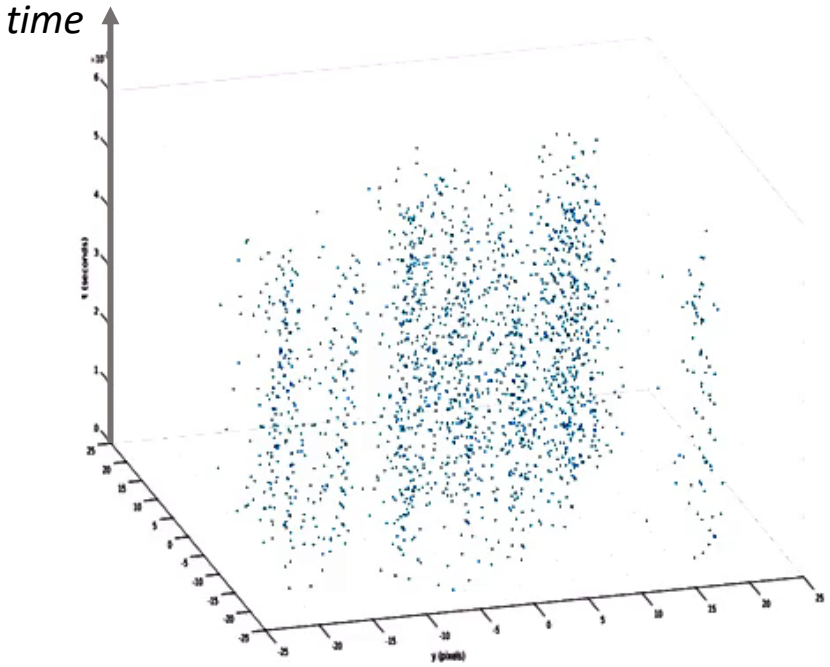
Zhu et al., *Event-based Feature Tracking with Probabilistic Data Association*, ICRA 2017. PDF, YouTube

# 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)



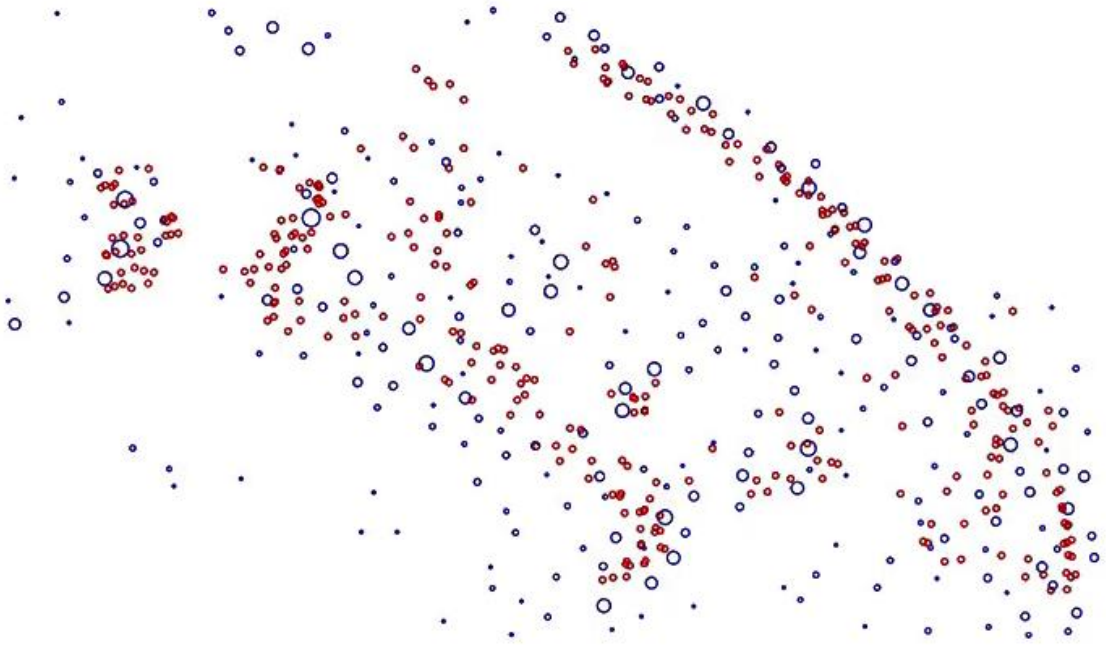

Motion compensated 2D point set
(like a top-view of space-time)

Zhu et al., *Event-based Feature Tracking with Probabilistic Data Association*, ICRA 2017. PDF, YouTube

# 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)



Point set $t - \Delta t$
Point set $t$

Zhu et al., *Event-based Feature Tracking with Probabilistic Data Association*, ICRA 2017. PDF, YouTube

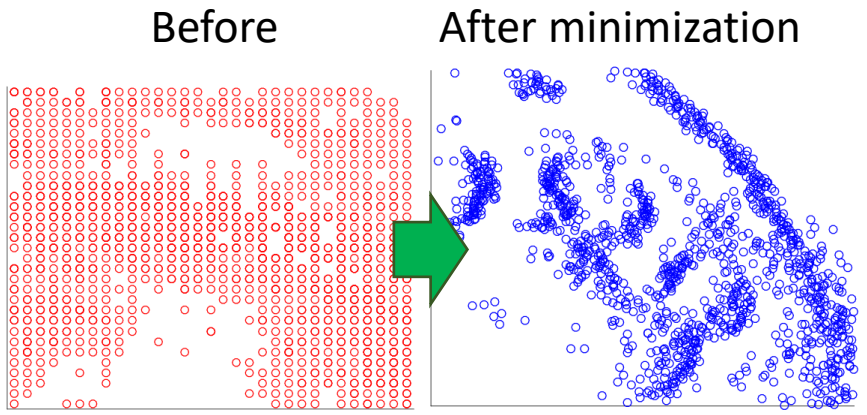
# Mathematical Details

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

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

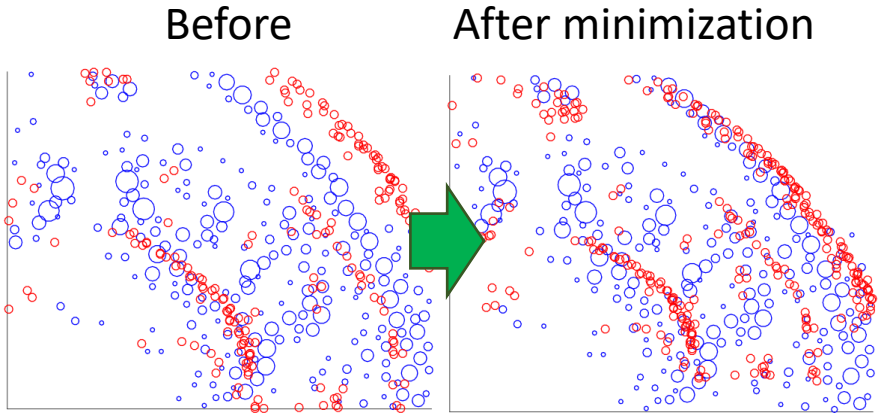
- EM Optical Flow Estimation

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



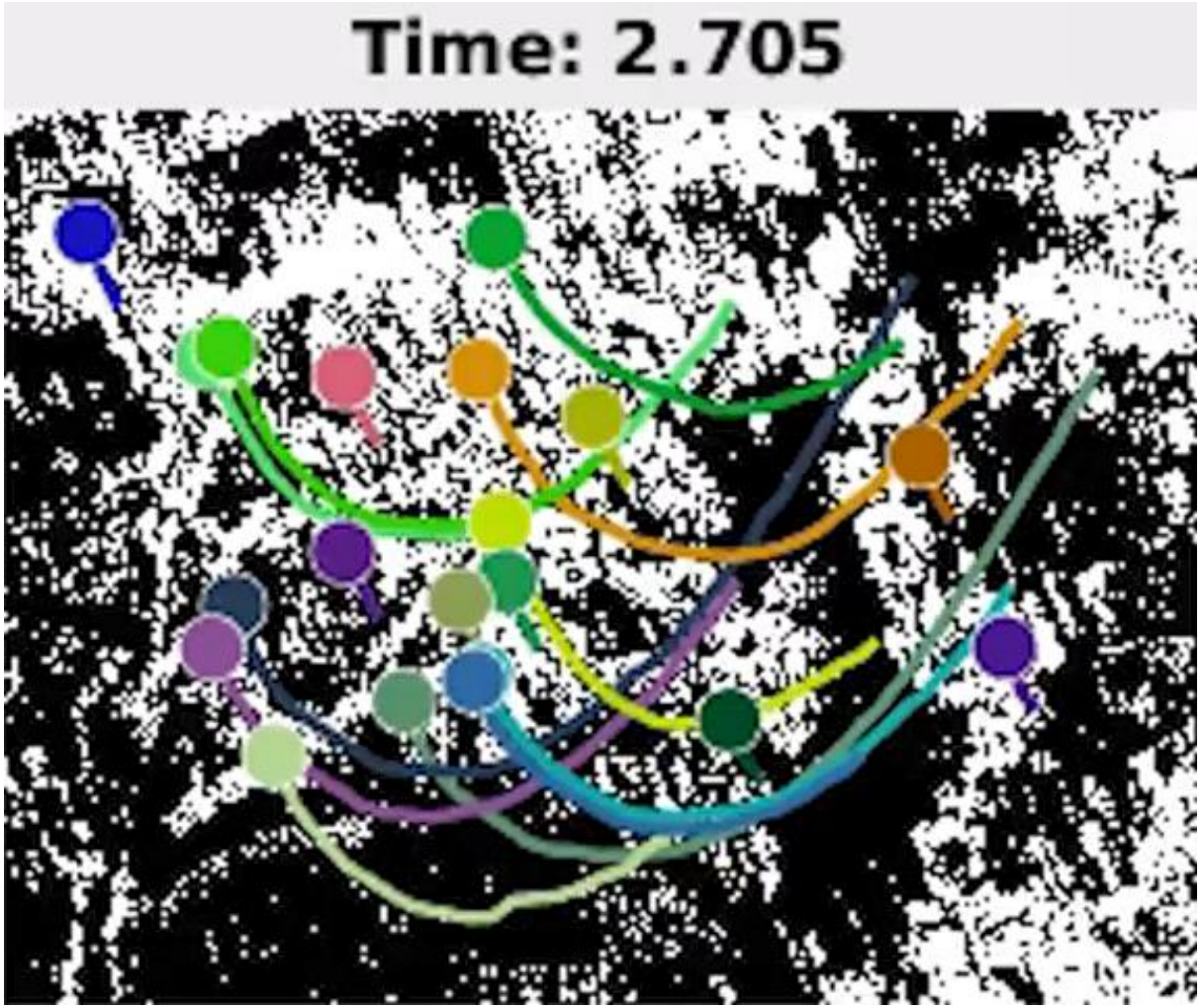

- Affine Feature Registration

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



# Tracking fast motions

- In normal conditions, tracking accuracy ~ 1 pix

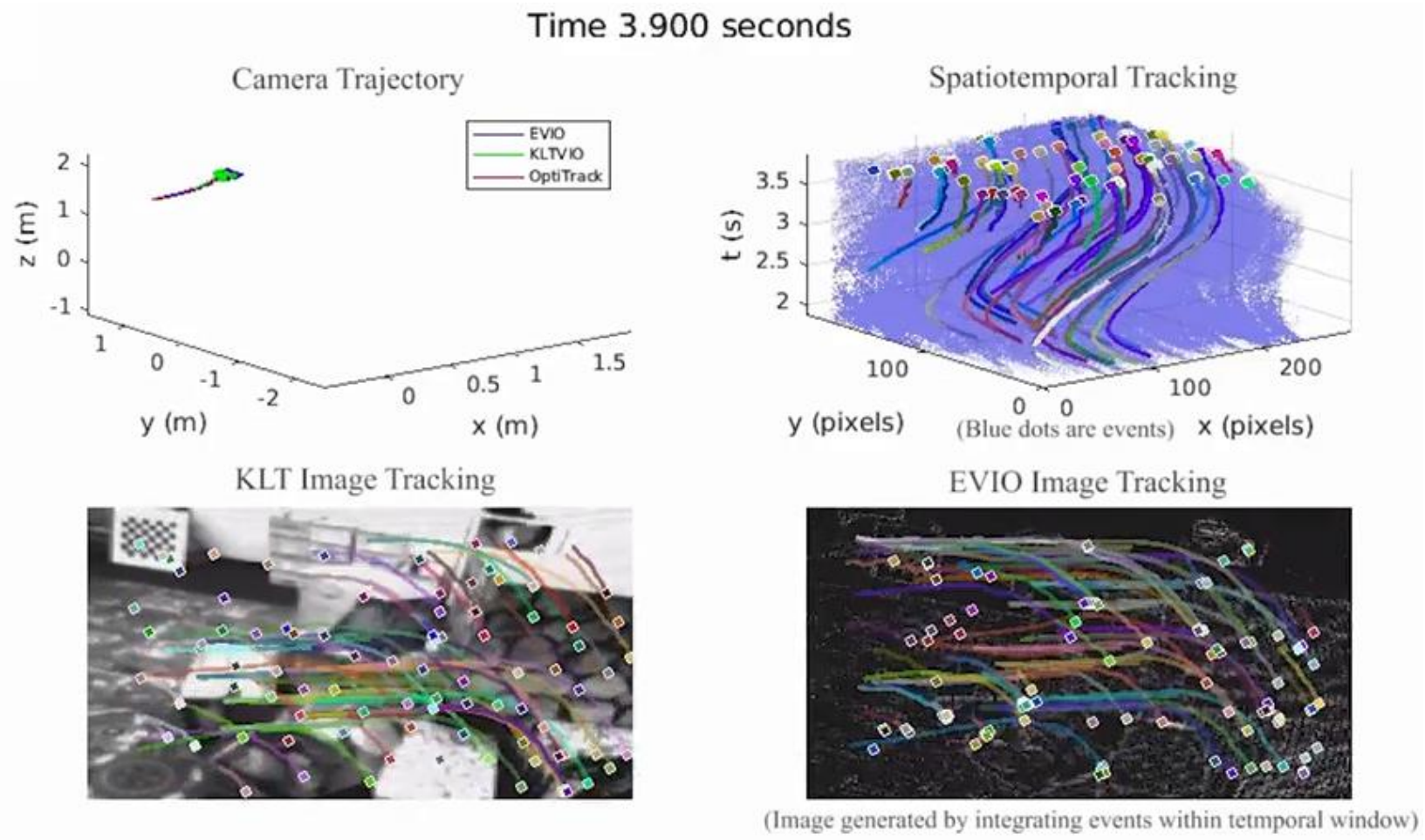



Slowed down 1/20x

Zhu et al., *Event-based Feature Tracking with Probabilistic Data Association*, ICRA 2017. PDF, YouTube

# Application to Visual-Inertial Odometry

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



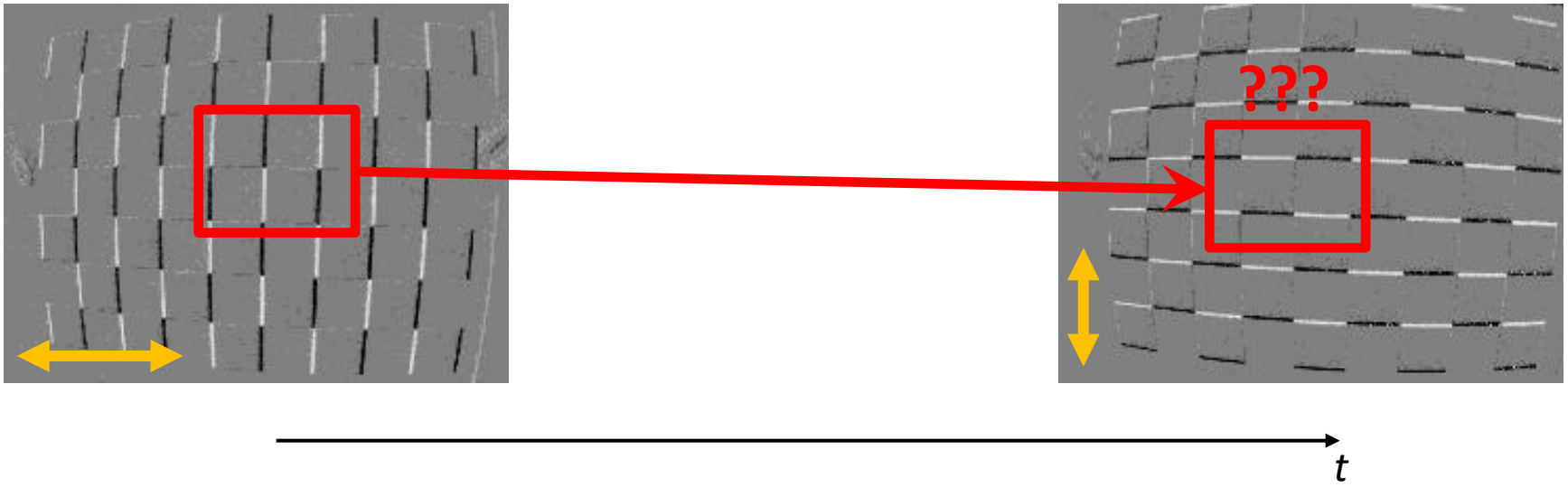

Zhu et al., *Event-based Visual Inertial Odometry*, CVPR 2017. YouTube

# More accurate?

(albeit more computationally expensive)

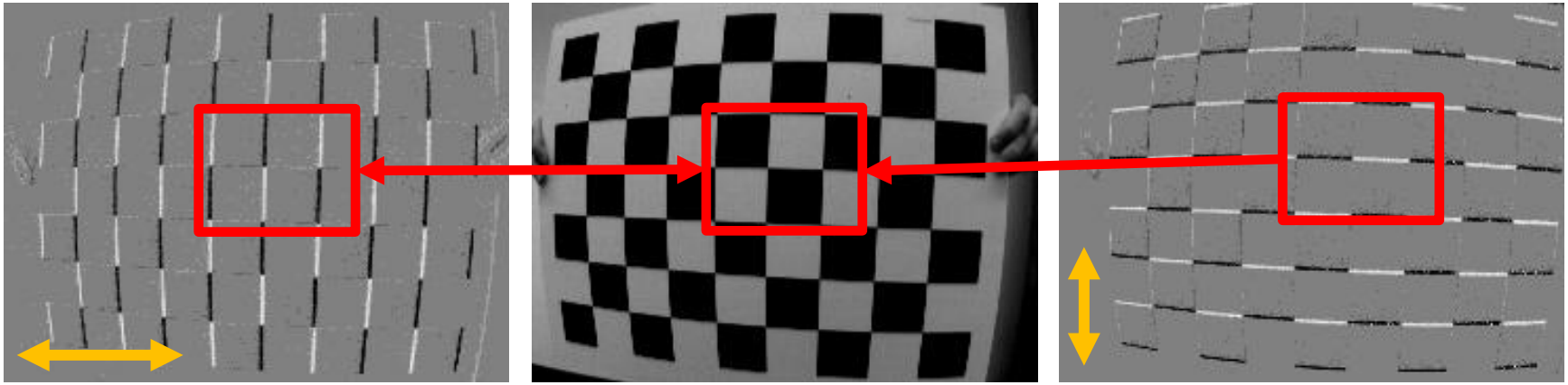
# The Problem of Data Association

- 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)

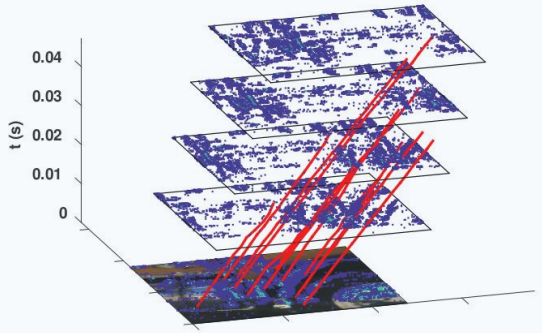
- Built from a **short time window of events**
  - Papers: Zhu ICRA’17, Rebecq BMVC 2017
  - ✓ Pros: it is not expensive and applies to natural scenes
  - ○ Cons: the edge-like pattern depends on motion. Drift will arise

- 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

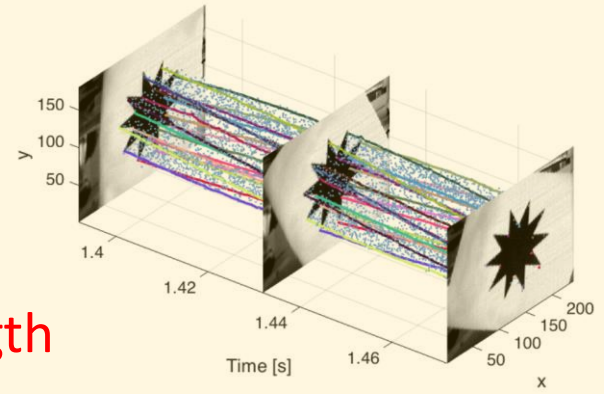



Events only

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

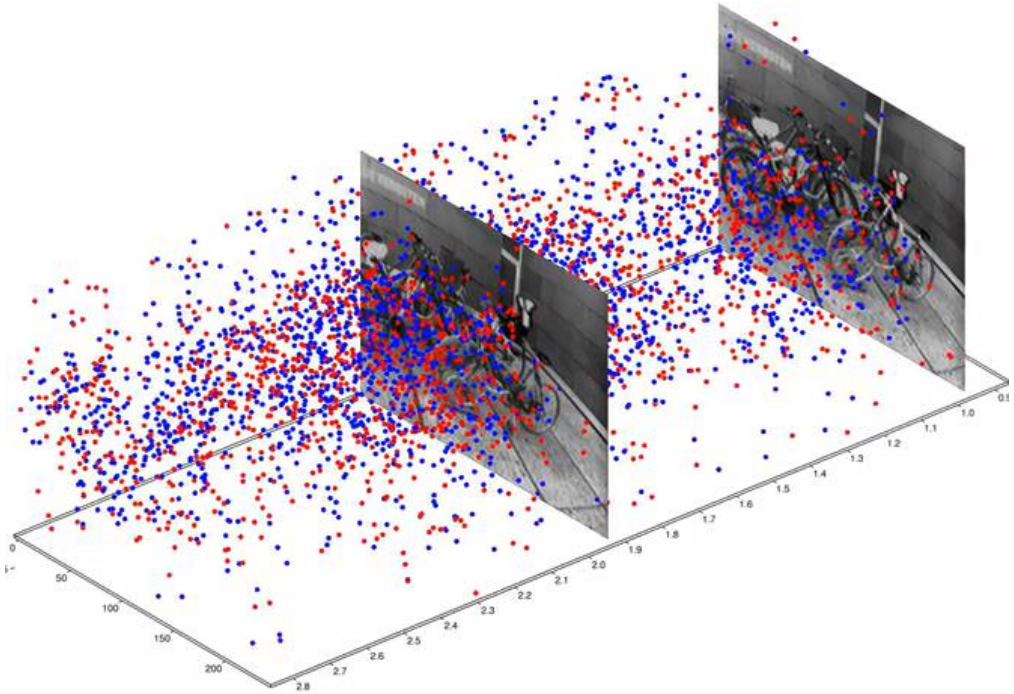
EKLT:

static appearance from frames

considers edge strength

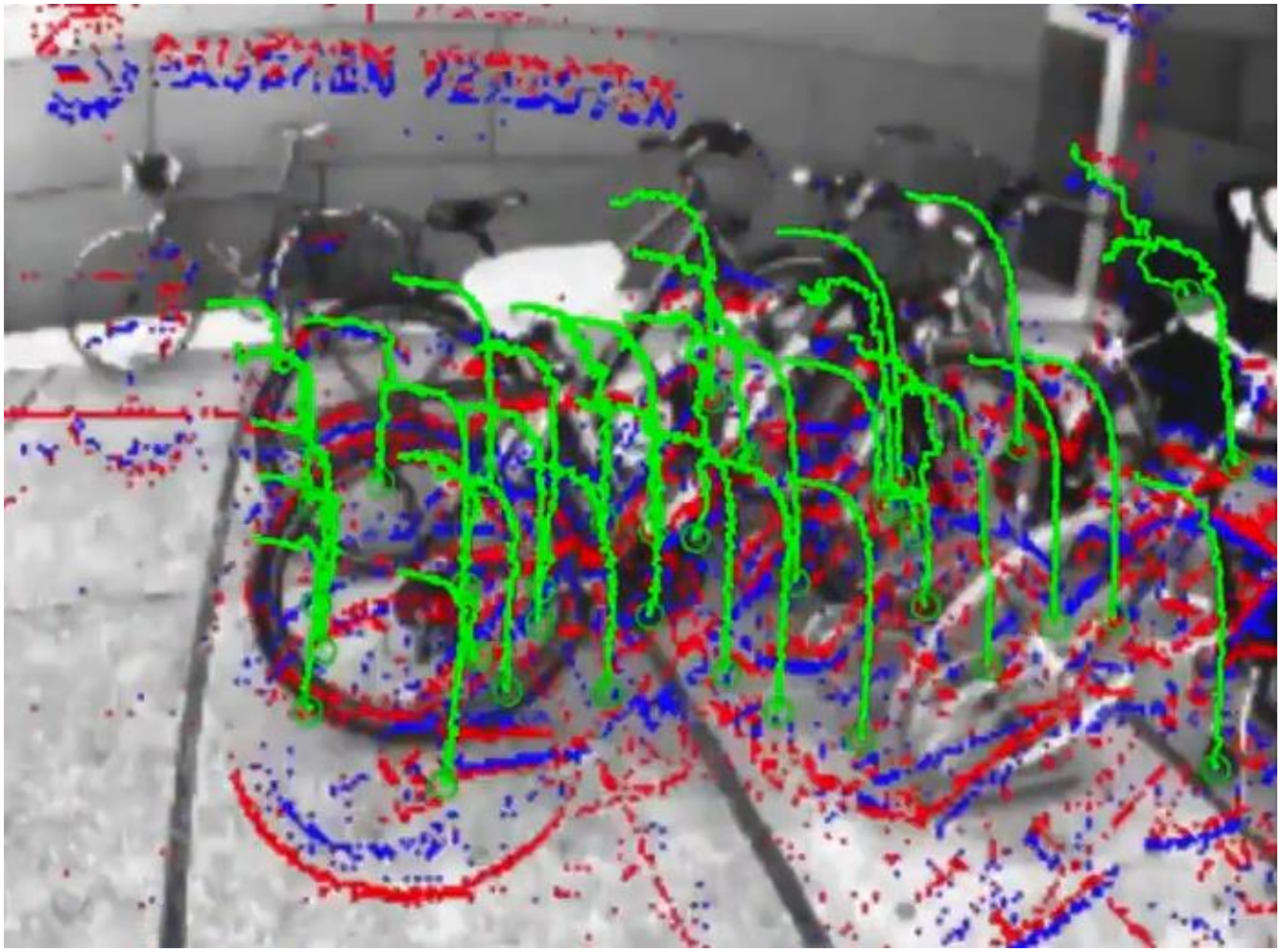
# EKLT: Event-based Kanade-Lucas-Tomasi Tracker

- **Goal**: Track features in the blind time between frames using events
- **Approach**: Extract features on frames and track them using events
  - **Asynchronous**, low-latency (µs), 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)



Events (ON, OFF)

EKLT tracks

Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

# EKLT Event-based Kanade-Lucas-Tomasi Tracker

**Brightness Increment**

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

**Brightness Constancy**

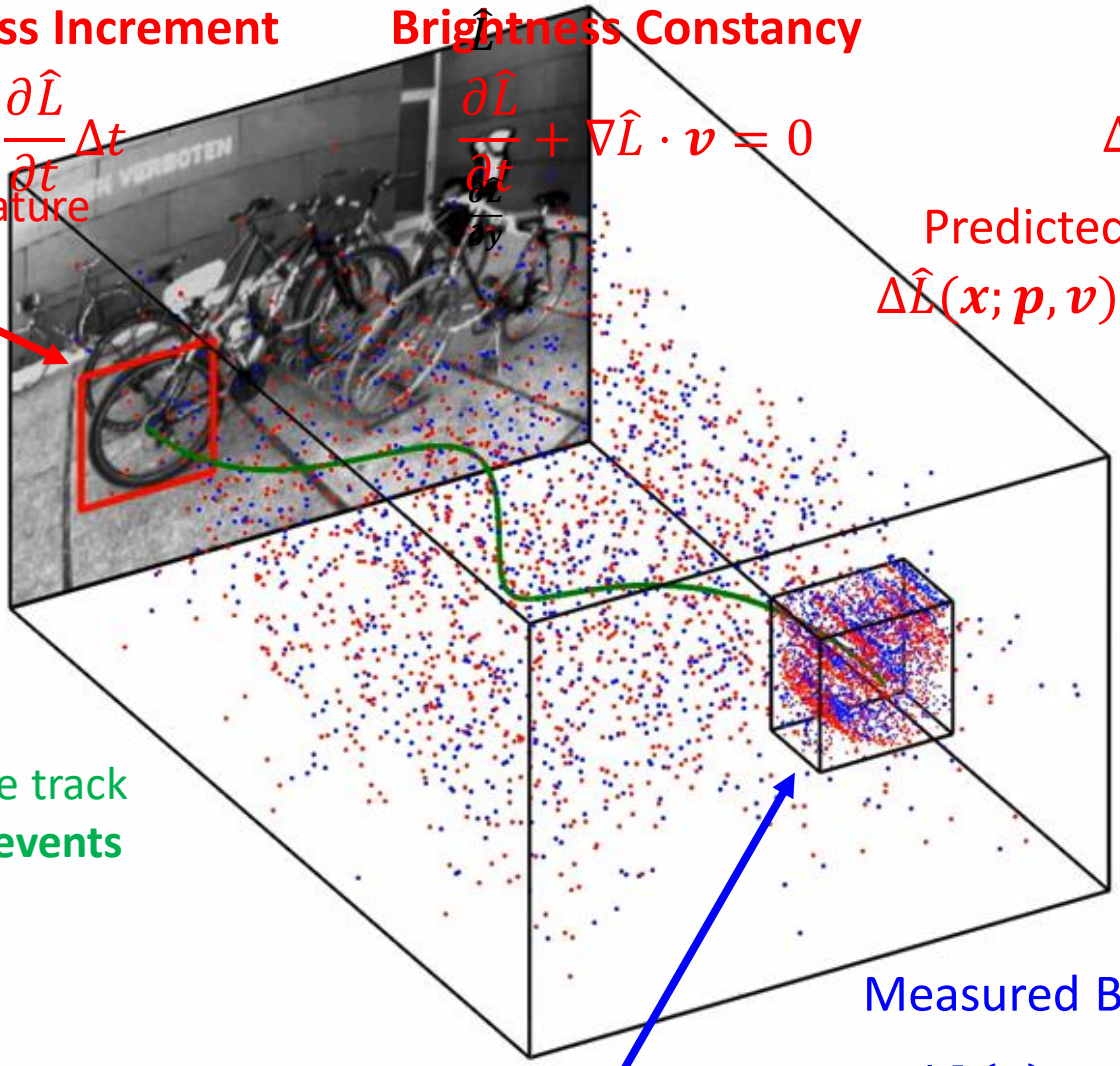
$$\frac{\partial \hat{L}}{\partial t} + \nabla \hat{L} \cdot \boldsymbol{v} = 0$$

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

Predicted Brightness Increment

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

Extract feature on **frame**



Feature track from **events**

**Events** in spatio-temporal window

Measured Brightness Increment

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

Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

# EKLT: Event-based KLT tracker

**Predicted** Brightness Increment

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

**Measured** Brightness Increment

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

**Maximum Likelihood** approach:

$$p\big(\mathcal{E}\big|\boldsymbol{p},\boldsymbol{v},\hat{L}\big) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}\int_{\wp}\Big(\Delta L(\boldsymbol{x}) - \Delta\hat{L}(\boldsymbol{x};\boldsymbol{p},\boldsymbol{v})\Big)^2 d\boldsymbol{x}\right)$$

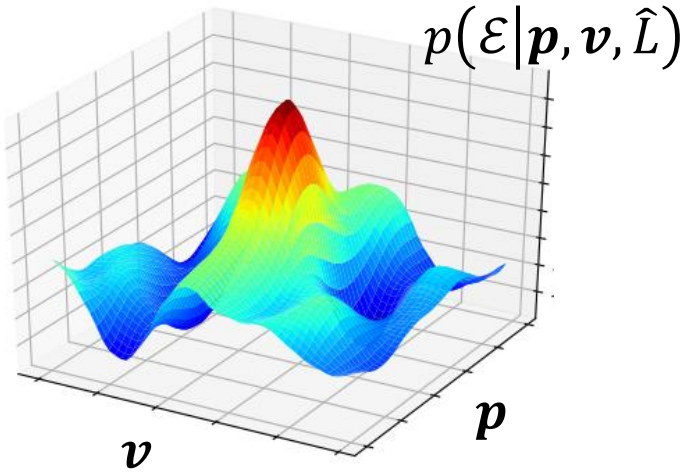
Events

Intensity frame

Optical flow

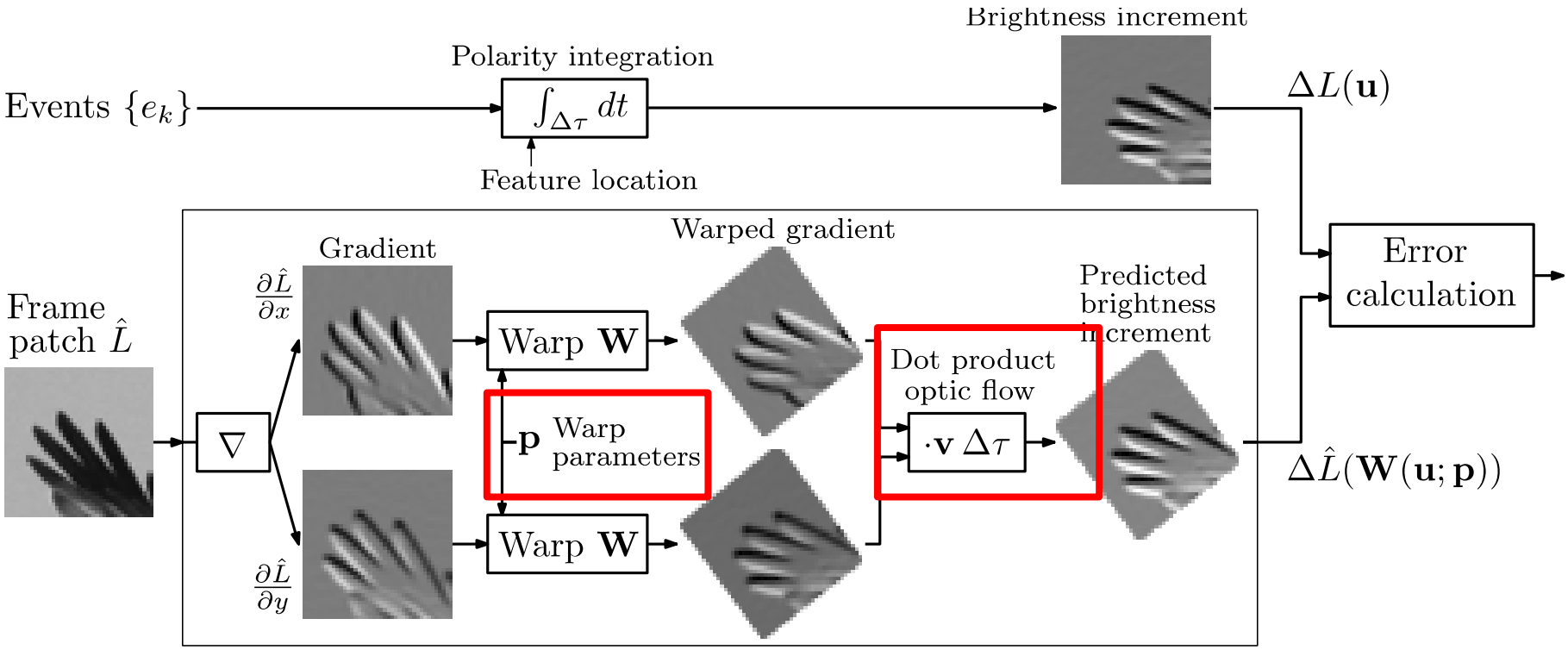
Feature parameters (warp)

Joint optimization




Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

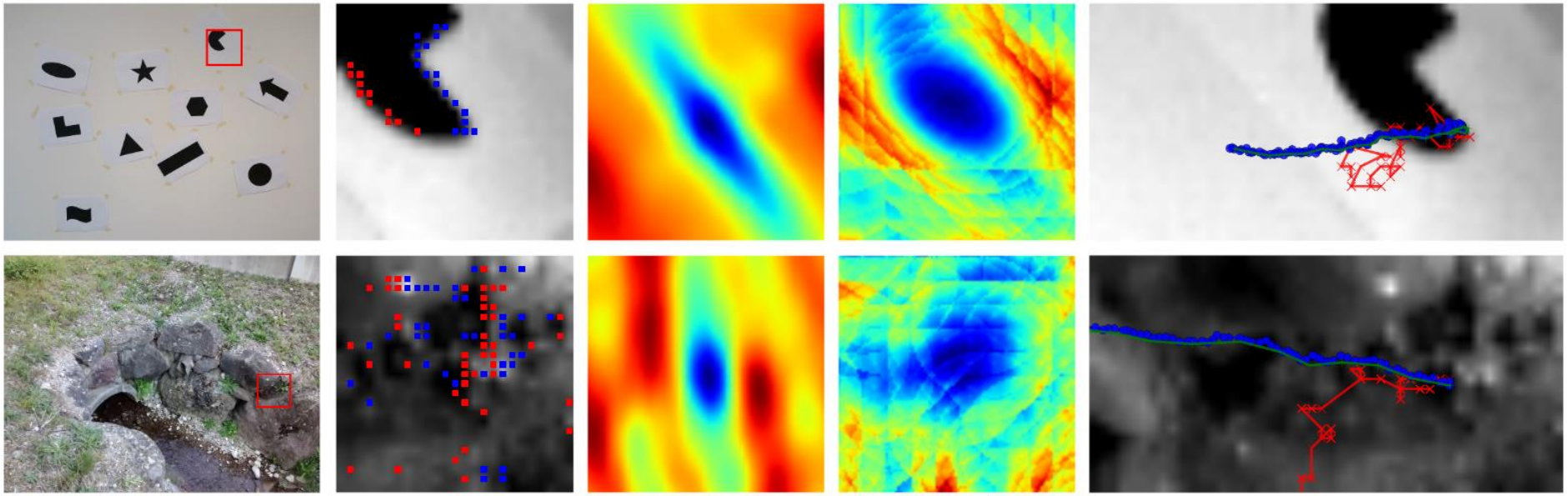
# Detail of Generation Model



# Tracking Comparison

- EKLT
- Kueng et al.
- Ground Truth

EKLT 2019 | Kueng IROS 2016



- ICP performs better on simpler scenes (sparse edges)

$$\sum_{(\mathbf{p}_i,\mathbf{m}_i)\in\text{Matches}} b_i \left\|\mathrm{R}\mathbf{p}_i + \mathbf{t} - \mathbf{m}_i\right\|^2$$

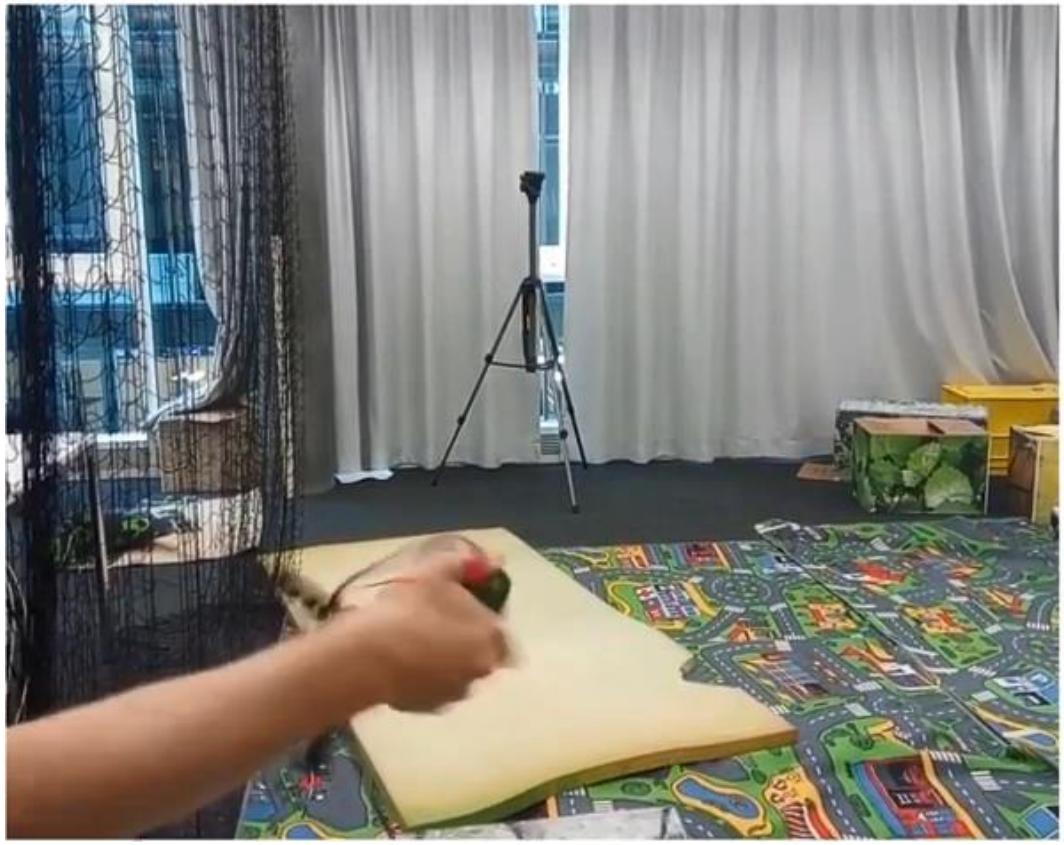
- The objective function in EKLT is better behaved → more robust in natural scenes

$$\left\| \frac{\Delta L(\mathbf{u})}{\|\Delta L(\mathbf{u})\|_{L^2(\mathcal{P})}} - \frac{\Delta \hat{L}(\mathbf{u};\mathbf{p},\mathbf{v})}{\|\Delta \hat{L}(\mathbf{u};\mathbf{p},\mathbf{v})\|_{L^2(\mathcal{P})}} \right\|^2_{L^2(\mathcal{P})}$$

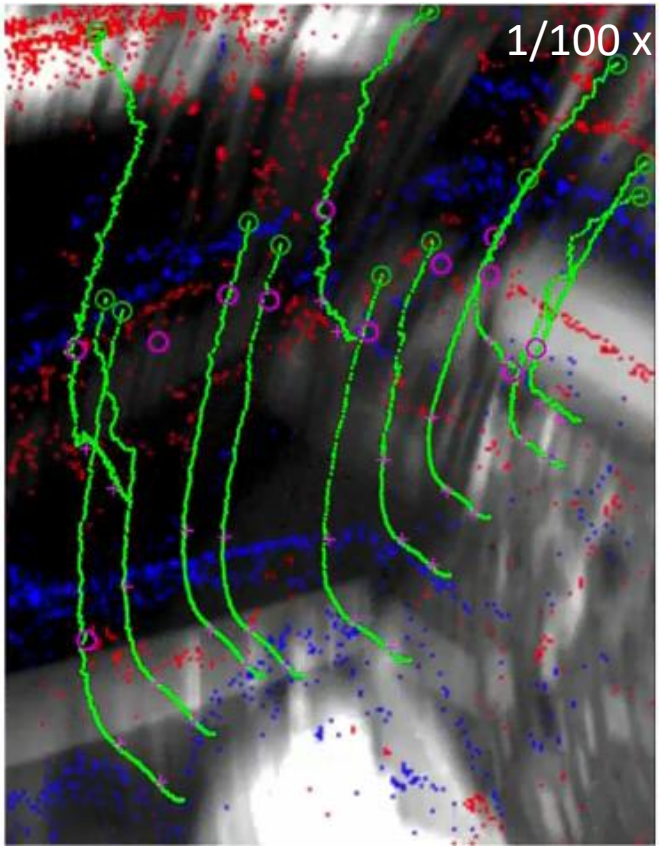
Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

# Tracking fast motions

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



Events (ON, OFF)




KLT tracker vs. EKLT tracker

Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

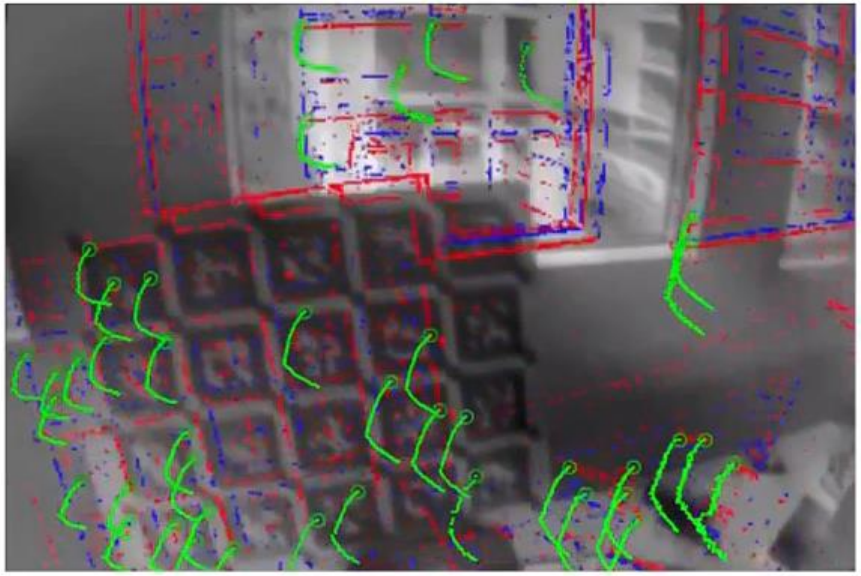
# Tracking in High Dynamic Range Scenes

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

Overexposed frame

Underexposed frame

HDR Frame reconstructed from events (ON, OFF)



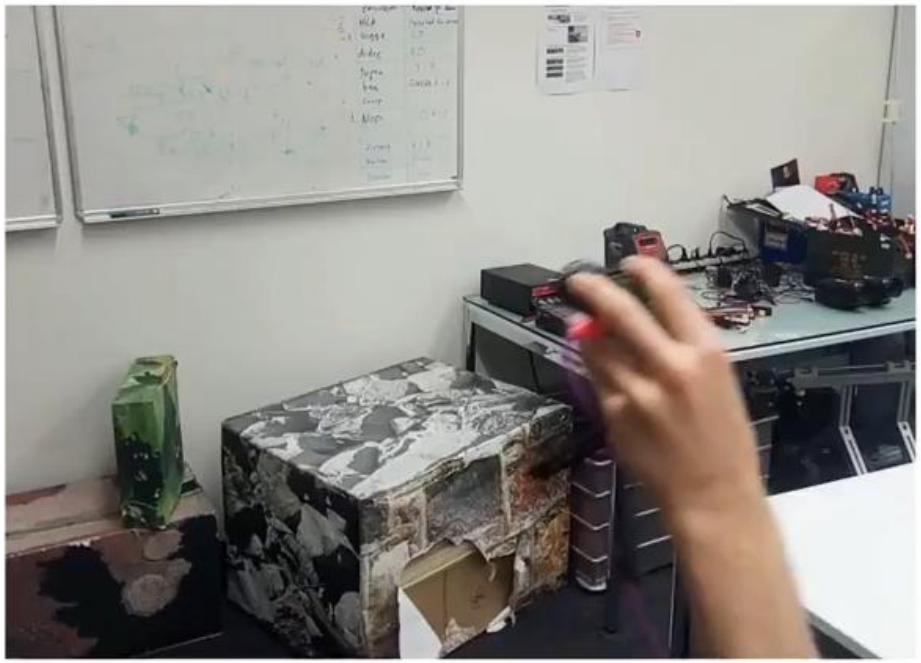
EKLT tracker

Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

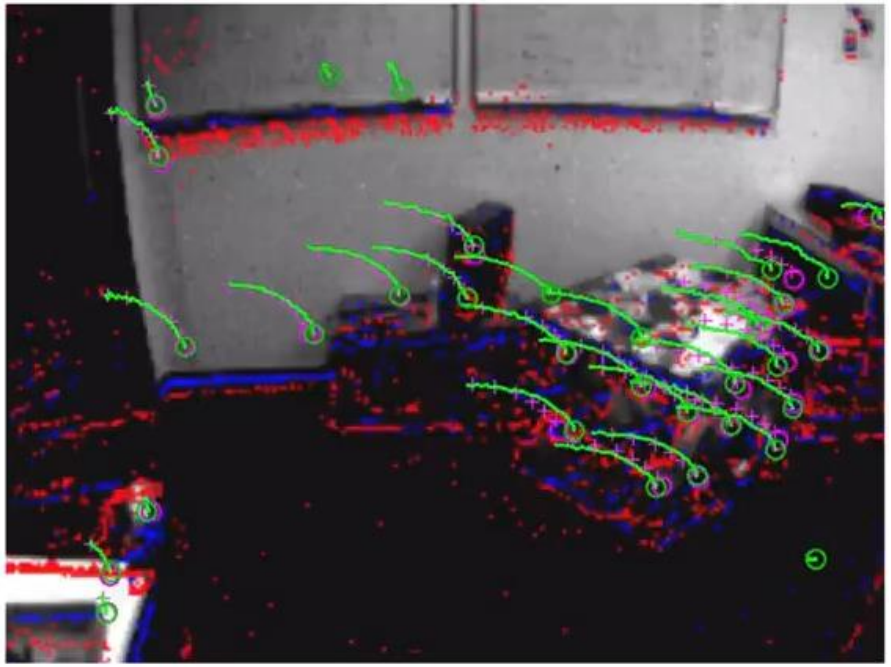
# Tracking in difficult illumination conditions

- Features tracked in low light and through abrupt illumination changes

Lights ON **/ OFF**



Events (ON, OFF)



KLT tracker vs. EKLT tracker

Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

# How do trackers compare?

# Comparison of five trackers

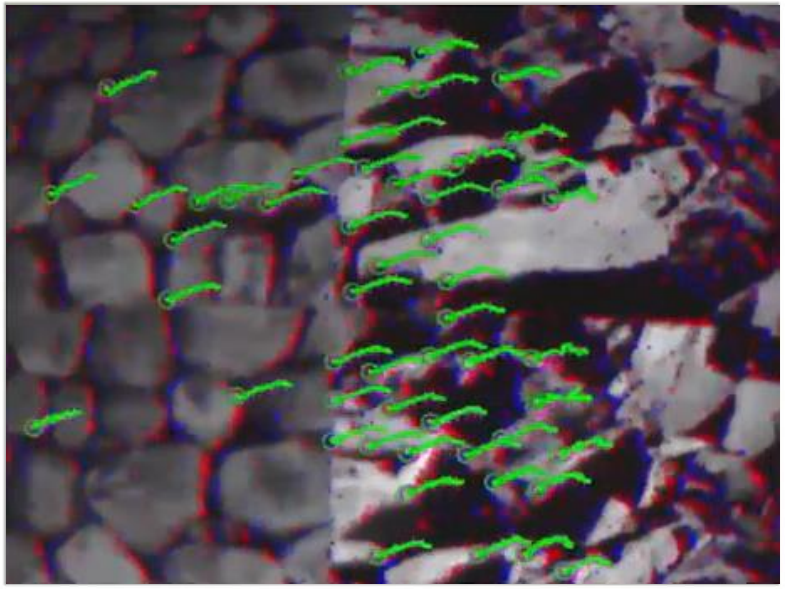
- **ICP** (Kueng et al. IROS 2016, Tedaldi EBCCSP 2016)
  - Iterative Closest Point algorithm on two point sets
- **EM-ICP** (Zhu et al. ICRA 2017)
  - Tracking on motion-compensated points sets
- **KLT-MCEF** (Rebecq et al. BMVC 2017)
  - Classic KLT tracker on motion-compensated images
- **KLT-HF**
  - 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

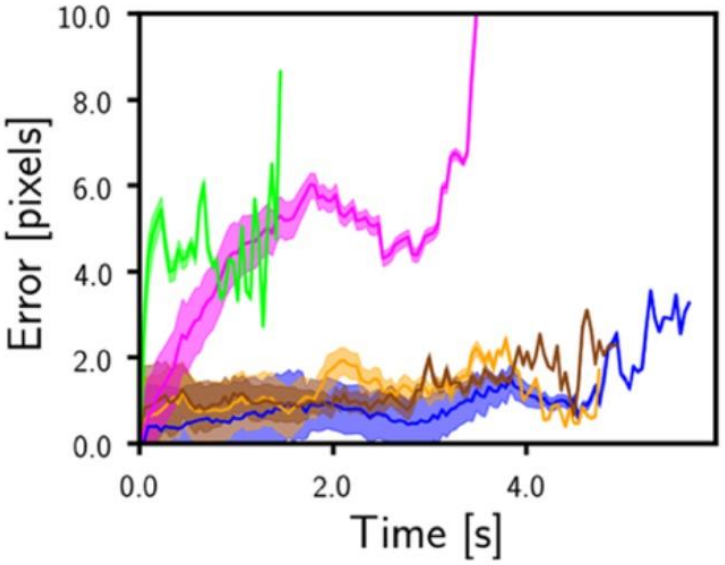
Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

# Comparison of multiple (five) trackers

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

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



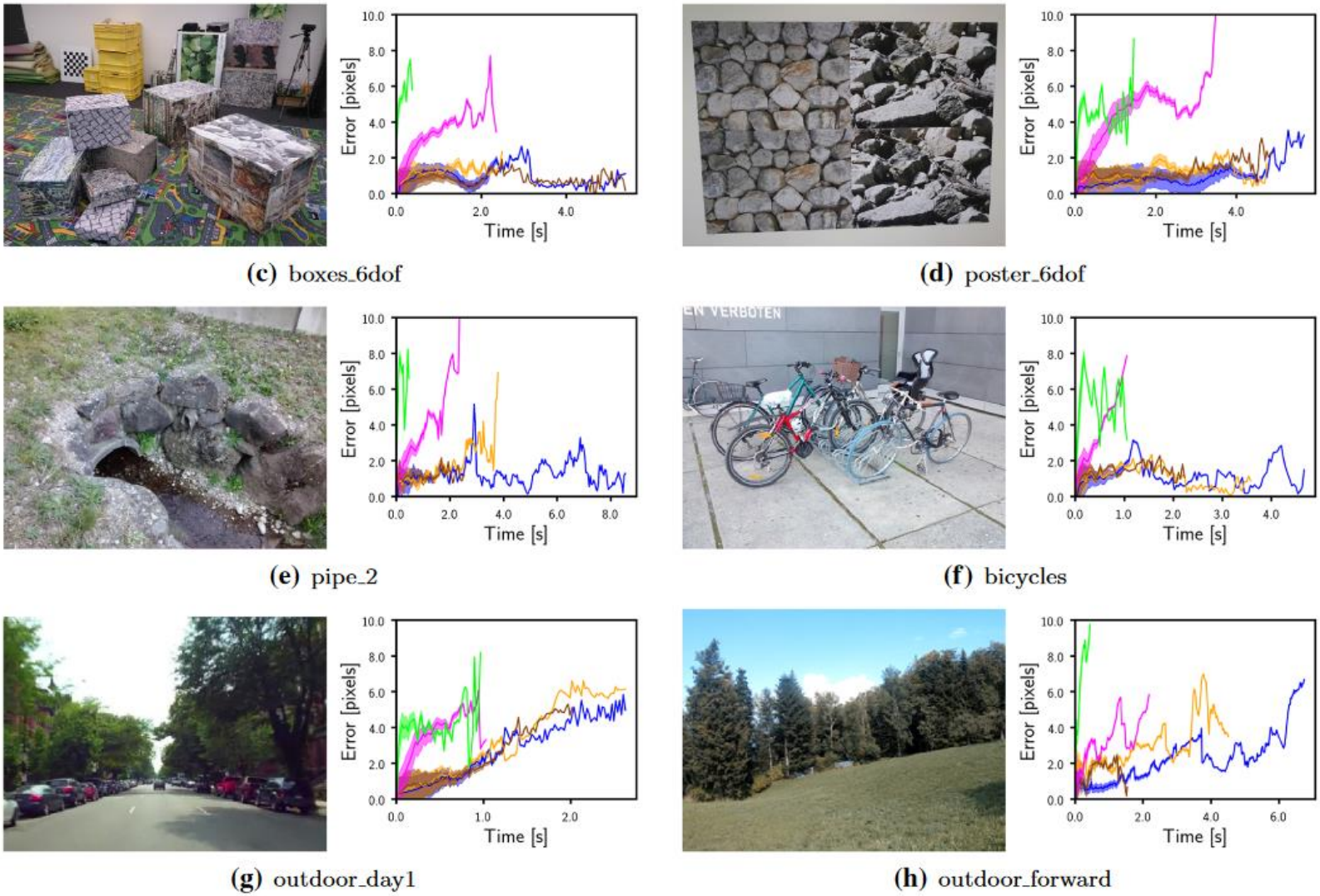



Trackers: ICP EM-ICP KLT-MCEF KLT-HF EKLT

Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

# 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

Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

# 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 | **0.80** | 1.49 | 2.31 | 0.94 | 2.43 |
| | checkerboard | **1.21** | 1.92 | 2.30 | 2.30 | 1.75 |
| High texture | poster_6dof | **0.64** | 2.48 | 3.10 | 0.97 | 1.18 |
| | boxes_6dof | **0.72** | 4.59 | 1.60 | 0.80 | 1.24 |
| Natural | bicycles | **0.76** | 4.22 | 1.50 | 1.26 | 1.21 |
| | pipe_2 | **0.78** | 4.90 | 1.63 | 1.04 | 1.06 |
| | outdoor_day1 | **0.71** | 2.96 | 2.30 | 2.00 | 2.52 |
| | outdoor_forward5 | **0.80** | 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

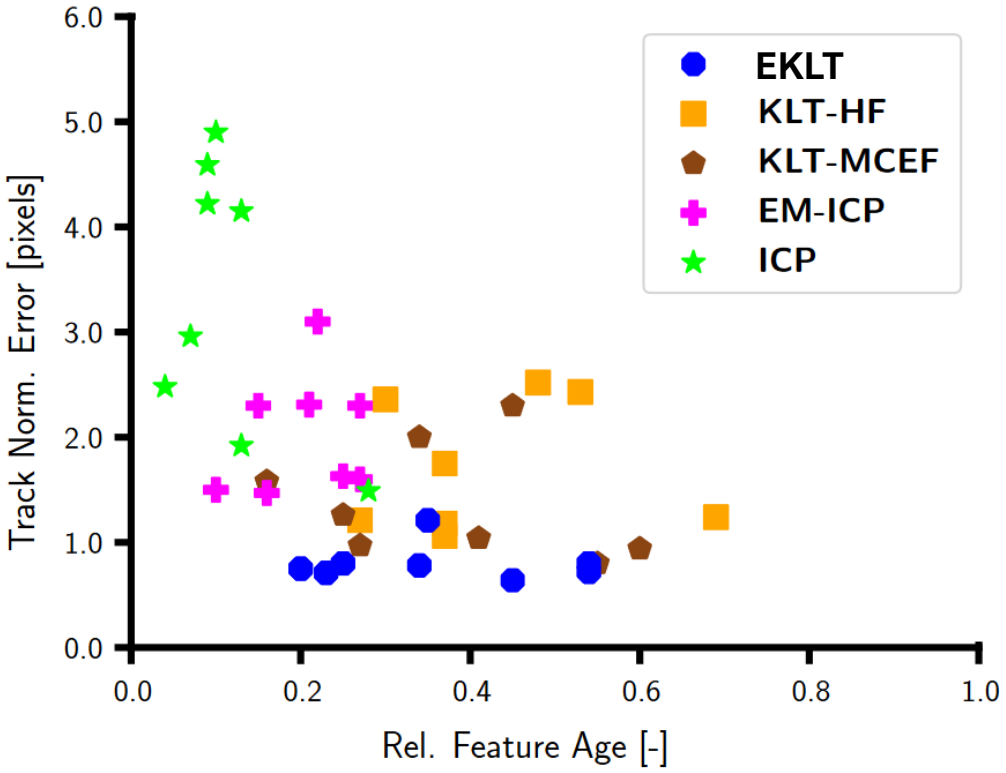
| Scene | Datasets | Relative feature age | | | | |
|---|---|---|---|---|---|---|
| | | Ours | ICP | EM-ICP | KLT-MCEF | KLT-HF |
| Black and white | shapes_6dof | 0.54 | 0.28 | 0.21 | **0.60** | 0.53 |
| | checkerboard | 0.35 | 0.13 | 0.27 | **0.45** | 0.37 |
| High texture | poster_6dof | **0.45**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

The best result per row is highlighted in bold

Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

# 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

Gehrig et al., *EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames*, IJCV 2019. YouTube, Tracking code, Evaluation code

# 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.