Standard and Event Cameras Fusion for Feature Tracking

Yan Dong
dongy18@mails.tsinghua.edu.cn
Department of Automation, Tsinghua University
Beijing, China

Tao Zhang
taozhang@tsinghua.edu.cn
Department of Automation, Tsinghua University
Beijing, China

ABSTRACT

Standard cameras are frame-based sensors that capture the scene at a fixed rate. They cannot provide information between two frames and suffer from the motion blur problem in high-speed robotic and vision applications. By contrast, event-based cameras are a novel type of sensors that generate asynchronous "events" if the intensity changes at a particular pixel. The data types of these two sensors are fundamentally different. In this paper, we leverage the complementarity of event-based and standard frame-based cameras and propose a fusion strategy for feature tracking. Features are extracted in frames, tracked by the event stream and updated when new frames come. The event camera tracks and predicts features on new frames, and the standard camera modifies features. This paradigm is different from existing fusion-based tracking methods which only use the first frame for initialization and abandon the following frames. We evaluate our method on Event-Camera Dataset [17] and show that the feature update process improves the tracking qualities.

CCS CONCEPTS

• **Computing methodologies** → **Tracking**; *Vision for robotics*; *Image processing*; • **Hardware** → Sensor applications and deployments.

KEYWORDS

event camera, multi-sensor fusion, feature tracking, computer vision

ACM Reference Format:
Yan Dong and Tao Zhang. 2021. Standard and Event Cameras Fusion for Feature Tracking. In *2021 International Conference on Machine Vision and Applications (ICMVA 2021), February 20–22, 2021, Singapore, Singapore.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3459066.3459075

MULTIMEDIA MATERIAL

Code: https://github.com/LarryDong/FusionTracking

1 INTRODUCTION

Event cameras are a novel type of bio-inspired visual sensors, often referred to as silicon retinas [20], Dynamic Vision Sensors (DVS) [21], or Dynamic and Active-pixel Vision Sensors (DAVIS) [3]. Different from standard frame-based cameras which periodically generate frames, event cameras send asynchronous streams of events. To be more specific, an event is transmitted from the chip when the brightness of a pixel changes. Each event contains three key components: the pixel location (x, y), the timestamp t, and a 1-bit polarity p of the brightness change (increase or decrease), denoted as $e = (x, y, t, p)$. Event cameras have the advantage of extremely high temporal resolution, low latency and high dynamic range (HDR) [9], gaining more and more popularity in computer vision and robotics.

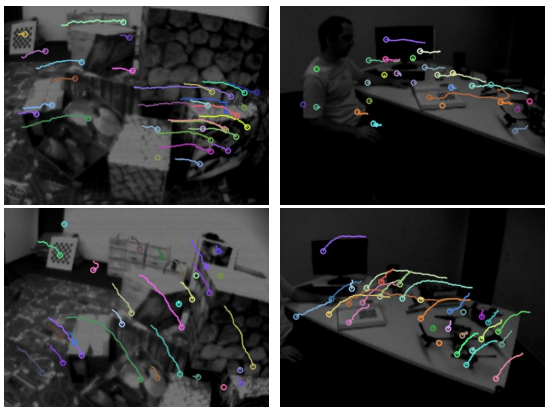

Figure 1: Tracking results on dataset [17].

Feature detection and tracking is a core block of many vision tasks such as simultaneous localization and mapping (SLAM) and visual odometry (VO). Over the past decades, many researchers have studied frame-based feature tracking and developed several methods. Unfortunately, due to the hardware limitation, frame-based feature tracking methods cannot provide features positions between two frames and have difficulties to handle the blur images.

Observing the complementarity that frame-based cameras can periodically provide complete intensity images while event cameras generate individual high-frequency events, we leverage the advantages of both sensors to develop a robust and long-term feature tracking method (see Figure 1). Features are initialized in frames from a standard camera and tracked by events, then features are updated once a new frame comes. The proposed algorithm makes full use of all frames as well as events data to achieve high-accuracy and long-term feature tracking.

The remainder of this paper is organized as follows. In Section 2, we show related work on event-based feature tracking methods. Section 3 presents our method. In Section 4, the proposed method is

ICMVA 2021, February 20–22, 2021, Singapore, Singapore

ACM ISBN 978-1-4503-8955-6/21/02…$15.00
https://doi.org/10.1145/3459066.3459075

tested on Event-Camera Dataset [17] and compared with baseline methods quantitatively. We also analyze the computational performance and point out the limitations. Finally, Section 5 contains our conclusions.

Algorithm 1 The Feature Tracking from Frames and Events

Initialization:
- Set the number of features to track K, template size S, the ratio for registration r.
- Construct model templates M_k and calculate edge pixels in each template as the size of model point set N_k.
- Accumulate events to create data patches D_k.

Tracking:
for every events generated in D_k **do**
  - Update the data points set of D_k.
  if $N_k \times r$ events updated in D_k **then**
    - Estimate registration parameters between data patches and model templates.
  end if
  if a new frame comes **then**
    - Update feature model M_k and check the consistency.
    - Modify data patches.
    - Add new features if tracking features smaller than K.
  end if
end for

2 RELATED WORK

In the early time of development of the event-based feature tracking, researchers use simple patterns to track. Censi et al. [6] used high-frequency blinking infrared LEDs as active markers to track the quadcopter. Mueggler et al. [16] estimated the pose of a quadrotor by observing a known black square. Nowadays, researchers have developed lots of methods for detecting and tracking simple structures such as lines [23][4][8][12] and corners [1][18][13].

Event-based tracking methods for more general features can be divided into two categories: tracking solely on events and tracking that needs frames. The methods which need frames usually firstly extracting features in frames and then tracking by events. In [11], features are first extracted from the edge image of the frame and initialized as model templates. Then data points sets are generated from the events stream. The iterative closest point (ICP) method [19] is adopted to calculate the transformation parameters between two points sets. An event-based KLT method (EKLT) [10] calculates a predicted brightness increment image utilizing the gradient from the first frame and obtains an event accumulated image from the events stream. The warping parameters of features are calculated by optimizing a target function containing two images.

A typical tracking method solely using events is proposed by Zhu et al. [25]. The motion parameters of features are estimated under an Expectation-Maximization (EM) structure. To avoid the feature deformation during the tracking, an affine model is adopted. Another method assumes multiple motion hypotheses for features and the best parameters are selected by a voting strategy [2]. As [10] points out, intensity images can be reconstructed from events so that some tracking methods can get free of a standard camera if intensity images can be reconstructed from events.

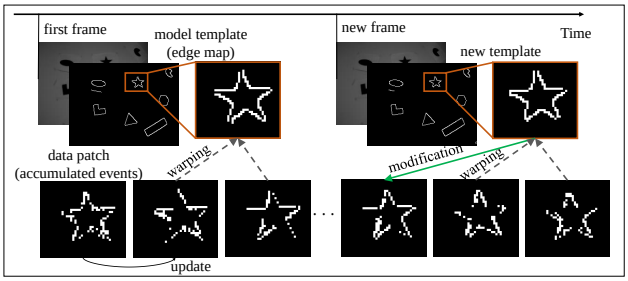

Figure 2: Feature tracking with frames and events. Before the new frame comes, data patches are tracked by calculating the warping parameters to corresponding model templates from the last frame. When the new frame comes, new templates are extracted, and then data patches are modified and tracked based on new templates.

To leverage the properties of both standard cameras and event cameras, some researchers use all frames and events for different tasks. [14] proposes an object tracking strategy that the rough position of the object is provided from the event stream and a convolutional neural network (CNN) is adopted to detect the object on frames. In [24], FAST corners are detected and tracked on both standard frames and event-accumulated images, then the reprojection errors on both images are optimized to estimate the camera pose.

3 METHODS

The method we proposed in this paper is inspired by [11] and has two main stages: initialization and tracking. We first select model templates on edge maps extracted from frames and accumulate events to construct corresponding data patches. Then in the tracking stage, the newest event replaces the oldest to update the data patches, and the ICP method is adopted to calculate the registration. When new frames come, we update model templates and modify corresponding data patches. These steps are summarized in Algorithm 1.

3.1 Feature Detection and Initialization

Based on the event generation model, events are generated when the brightness L changes. Under the constant illumination and the brightness constancy assumptions, the intensity increment ΔL during a small time interval Δt can be approximately written as [9]:

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

where ∇L is the brightness gradient of the intensity image and v is the velocity. It implies that events are dependent on the gradient and the moving direction, which leads to the problem that event-based features are motion dependency. To avoid this problem, we do not estimate the velocity and use edges instead of the gradient as model templates.

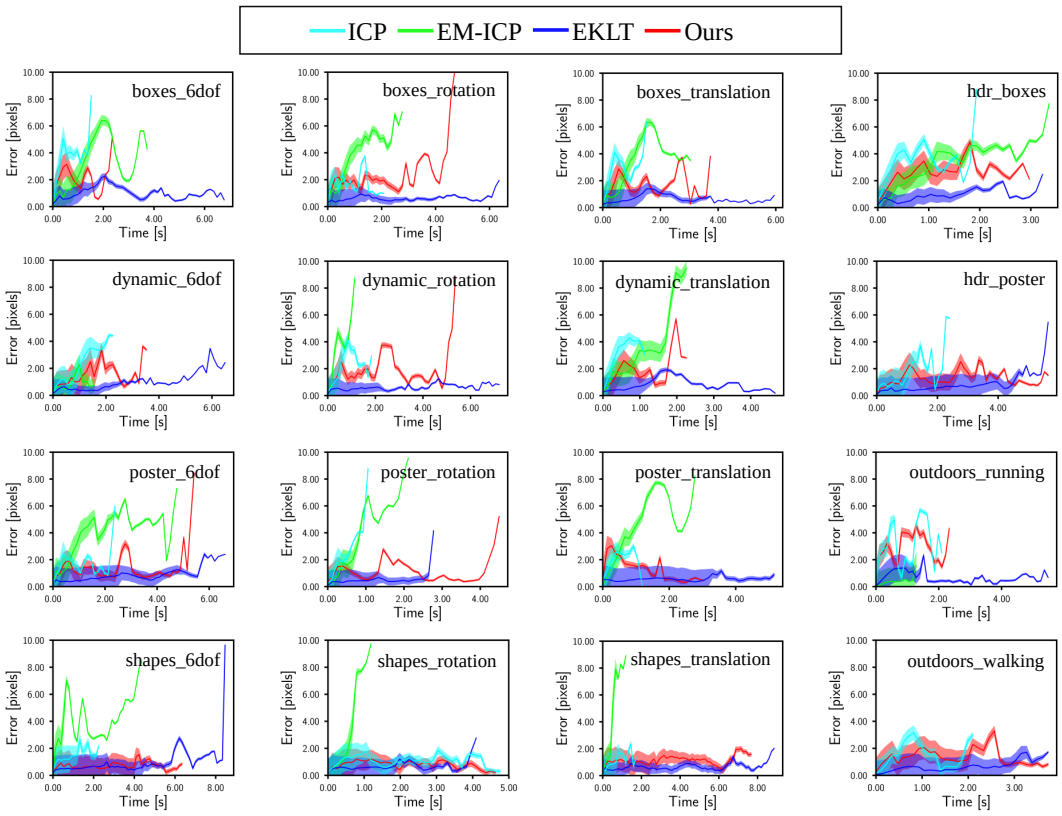

Figure 3: Feature tracking on all 16 sequences from dataset [17]. The solid lines stand for the mean tracking error of all tracked features and corresponding shading area is proportional to the number of tracked features. Note that our method has relative lower tracking errors and longer feature ages on simple sequences.

3.1.1 Model templates initialization. We detect edges on absolute brightness images by Canny's method [5]. Corners are intersecting edges, which are more informative, so edges around good features detected by [22] are initialized as model templates. We still use the name of "model point set" in [11] to describe all edge pixels in model templates.

3.1.2 Data patches initialization. To initialize features from the event stream, we simply accumulate events that occurred in the area of model templates. Accumulated events construct the "data point set" for each data patch. We set the size of each data point set the same as the size of the corresponding model point set.

3.2 Feature Tracking from Events

Features between frames are tracked using events. To be more specific, we estimate the warping parameters between data patches and model templates. Data patches are updated when new events come and the ICP algorithm is adopted for the warping parameter registration.

3.2.1 Data patches update. We maintain the size of each data point set by adding the newest event occurs in the data patch spatial window and deleting the oldest. Notice that the warping parameters of data patch change after each registration so the spatial window is related to the rotation angle and position.

3.2.2 Warping parameters registration. The feature is described by its position and rotation. We calculate the transformation (warping parameters) between moving data patches and model templates on receiving new events.

Since data patches are initialized at the same positions of model templates, and the rotation matrices are set to identity, we only need to calculate the increment for transformation. More mathematically, we denote ${}^{m}_{d}T_{pre}$ and ${}^{m}_{d}T_{cur}$ as the transformation from the data patch to the model template at the previous and the current time respectively, and the increment is ΔT. We have the following equation:

$$ {}^{m}_{d}T_{cur} = \Delta T \cdot {}^{m}_{d}T_{pre} $$

To obtain the increment ΔT, we first warp points in the data point set of the current data patch by the previous transformation parameter ${}^{m}_{d}T_{pre}$ which is already known. Then the warped data points, $\{d_i\}$, is registered to points in model points set, $\{m_i\}$, minimizing the Euclidean distance between the sets:

$$ \arg\min_{\Delta T} |\Delta T(d_i) - m_i|^2, $$

where ΔT is exactly the registration parameters from the data point set to model point set and is calculated by the ICP method. Readers can refer to [11] for more details.

We point out that warping parameters can be updated every time if only one event is updated. However, information carried by each event is limited, and processing the registration for every event is time-consuming. In practice, we operate for a batch of events and the batch size depends on the size of the data point set, *e.g.*, when 50% of points in the data point set are updated.

3.3 Feature Update from Frames

Features are updated once absolute intensity images are received. The update has three steps: new model templates extraction, the feature consistency check, and the data patches modification. After the update, data patches are tracked based on the new corresponding model templates (see Figure 2). New features are added to the tracker if some features are lost during tracking.

3.3.1 New model templates extraction. We use the KLT tracker [15] to track features on frames. Since features are tracked from events between frames, features positions obtained from events are used as initial values for the KLT tracker. After obtaining new features positions, we extract edge maps again in spatial windows of features as new model templates and reconstruct the model point sets.

3.3.2 Consistency check. During tracking, feature positions may drift because we update model templates on every frame and the position and angle errors may accumulate. We adopt the enhanced correlation coefficient criterion [7] to check whether current model templates are consistent with initialized templates under a homography motion model. If the current template cannot be aligned to the initial template, we consider the feature is lost.

3.3.3 Data patches modification. Data patches are modified based on model templates. First, the outlier points in data point sets are removed if their minimum distances to points in new model point sets are larger than a threshold (*e.g.* 2 pixels). Then warping parameters are modified because positions of model templates in the image have changed. After both model templates and data patches are updated, features will be tracked by events again (Section 3.2) until the next frame comes. If some features are lost, we can immediately add new features from the current frame (Section 3.1).

The event-based tracker provides good initial values for the KLT tracker, making large displacement features tracking more robust. If features are moving too fast, however, the intensity image may blur, as a result, we estimate the feature speed from events and avoid updating model templates on blurred images. Besides, this method always modifies model templates on the newest frame, solving the features deformation problem due to the perspective change of the camera. The outliers removal in data point sets prevents accumulating noises from events.

4 EXPERIMENT

4.1 Introduction to Dataset and Baselines

We evaluate our algorithm on Event Camera Dataset [17]. The dataset generated by a 240×180 pixel DAVIS camera contains the events, images, IMU measurements, and camera calibration as well as ground truth from an external positioning system. We select 16 sequences: namely pure rotation ("rotation"), pure linear movement ("translation"), and 6 degrees of freedom movement ("6dof") on shapes, poster, boxes, and dynamic scene; "poster" and "boxes" under HDR illumination; "walking" and "running" in the outdoor environment.

We compare the proposed feature tracker against three baselines: ICP [11], EKLT [10], and EM-ICP [25], which we introduce further. For all methods, features are created at the initial time by own strategies and parameters and then tracked.

The ICP method [11] extracts Canny edges on the first grayscale frame to represent features and then tracked on event stream by point sets registration. Unfortunately, the implementation of this method has not been published by authors. Since our proposed method is an improvement for this ICP method, we simply disable the feature update process as the implementation of the ICP method.

The EKLT method [10] is a state-of-the-art event features tracker which extracts gradient features on frames and tracks using events based on event generative model. The c++ implementation is available[1] and we use all default parameters for comparison.

The EM-ICP [25] proposed by Zhu et al. computes association probabilities of feature point sets in an intertwined EM scheme. This method tracks features relying solely on raw events and the source-code implemented by MATLAB is released by the author[2]. We use the default parameters except for changing the patch size.

4.2 Quantitative Analysis

To evaluate the performance of the proposed method and baseline methods mentioned above, we compare the track-normalized tracking error and the relative feature age [10] of each method. We use the open-source feature tracking evaluation package[3] to generate the ground truth and calculate tracking errors as well as feature ages. The results are shown in Figure 3 and Table 1.

We notice that the proposed method outperforms the ICP method and EM-ICP method in general, although comparing to EM-ICP is unfair because EM-ICP uses only events. As for the original ICP method, tracking errors of our method are decreased by more than 30% and feature ages are doubled on most sequences. However, although on some sequences our method's feature ages are longer than EKLT's, the accuracy of our method is relatively lower. One reason is that the EKLT method adopts gradient templates which contains much more information than our edge templates. Generally speaking, our method performs better on bright and simple scenes (*e.g.* "shapes" and "poster") than highly textured or dark environments (*e.g.* "boxes" and "outdoors").

4.3 Performance of a Full Tracking

In Section 4.2, we compare tracking qualities only for features initialized at the beginning because implementations of baseline methods are not able to extract new features after the first frame. However, our method utilizes all frames from standard cameras which can add new features on every coming frame. Figure 4 shows

[1] https://github.com/uzh-rpg/rpg_eklt
[2] https://github.com/daniilidis-group/event_feature_tracking
[3] https://github.com/uzh-rpg/rpg_feature_tracking_analysis

Table 1: Tracking error and feature age of different feature tracking methods. The track error and feature age are calculated by the feature tracking analysis package in [10]. The best results on different sequences are highlighted in italic.

Sequence	Tracking error (px)				Feature age (s)			
	Ours	ICP	EKLT	EM-ICP	Ours	ICP	EKLT	EM-ICP
boxes_6dof	1.79	3.19	*0.95*	2.64	1.43	0.74	1.62	*1.93*
boxes_rotation	1.09	1.65	*0.56*	3.10	1.05	0.57	*1.65*	1.41
boxes_translation	1.59	2.44	*0.64*	2.47	1.33	0.71	*1.73*	1.38
dynamic_6dof	1.74	1.94	*0.56*	0.89	*2.57*	1.68	1.14	1.63
dynamic_rotation	1.42	1.93	*0.50*	3.54	*1.52*	1.01	1.46	1.07
dynamic_translation	1.98	2.68	*0.62*	3.59	1.47	0.74	1.26	*2.04*
hdr_boxes	2.09	3.08	*0.76*	2.72	*1.80*	0.78	1.07	1.67
hdr_poster	1.02	1.38	*0.61*	-	3.29	1.47	*3.73*	-
outdoors_running	1.62	2.81	*0.92*	0.41	0.97	0.62	*1.44*	1.36
outdoors_walking	1.19	1.40	*0.57*	-	1.26	1.00	*2.62*	-
poster_6dof	1.02	1.52	*0.66*	3.06	2.94	0.83	*3.55*	1.92
poster_rotation	1.07	1.44	*0.42*	1.84	*1.92*	0.48	1.38	0.88
poster_translation	1.28	1.85	*0.53*	3.13	1.27	0.59	*3.19*	1.11
shapes_6dof	0.92	1.31	*0.65*	3.42	3.66	1.45	*4.09*	0.41
shapes_rotation	0.90	1.08	*0.60*	2.09	1.79	*1.94*	1.89	0.25
shapes_translation	0.68	0.92	*0.59*	3.08	*3.56*	0.97	2.50	0.58

the results of the full tracking on "shapes_6dof". For "shapes_6dof", the speed of the event camera movement is increasing and we can see that the number of lost features increases sharply from 28s. Since new features are added to tracking from new frames, our algorithm can maintain to track around 8 features during the full sequence.

4.4 Computational Performance

For our proposed method, many factors can influence the computational performance, *e.g.*, the patch size (S), feature numbers (K), or batch sizes for calculating ICP (r), so it is difficult to give a thorough analysis. On "shapes_6dof" sequence, our algorithm can process about 200,000 events per second, when tracking 15 features and ICP process operated when half of the events in data point sets are updated ($S = 25$, $K = 15$, $r = 0.5$). This limit is computed using an unoptimized C++ single-threaded implementation on a laptop of Intel i7 CPU with 1.8GHz. As for baseline methods, the open-source code of EM-ICP implemented in MATLAB runs much slower, and EKLT methods implemented by C++ can only process around 17,000 events per second on a CPU with 3.2GHz [10].

4.5 Limitations

Readers should remember that in Section 3.1 we construct model templates from the edge map, which means our method is sensitive to the threshold of the Canny edge detector. When matching the points between data point sets and model point sets in the ICP process, if the threshold is too high, weak edges cannot be detected and data points generated by these weak edges may be removed as outliers when far away from real edges; on the contrary, if we set the threshold too low, detected edges can be extremely dense on high texture areas and the data points can be matched to any edge pixels nearby. Therefore, it is impossible to set a proper threshold across a variety of scenes.

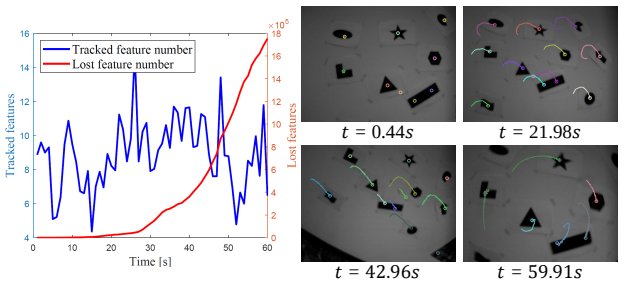

Figure 4: The full tracking on "shapes_6dof". The left figure shows the number of average tracked features per second and total lost features. Features at different timestamps are drawn on the right.

5 CONCLUSION

In this paper, we have introduced a novel approach for feature tracking frames and events fusion. Model templates are initialized from the edge map of frames and tracked by the ICP method using events. When receiving new frames, existing features are updated and new features are added to track if necessary. This method exploits all frames from standard cameras for tracking, while existing approaches using events solely or events with the first frame. The presented result reveals that by operating the feature update process, it outperforms the original ICP method in terms of the tracking error and the feature age. We also show the tracking performance over a whole sequence.

The proposed method explores how to utilize both the frame-based images and event-based data for feature tracking. We hope it can inspire researchers who devote to fuse standard cameras and event cameras for different tasks. We will further explore a full SLAM pipeline with standard and event-based cameras fusion.

REFERENCES

[1] I. Alzugaray and M. Chli. 2018. Asynchronous Corner Detection and Tracking for Event Cameras in Real Time. *IEEE Robotics and Automation Letters* 3, 4 (2018), 3177–3184. https://doi.org/10.1109/LRA.2018.2849882

[2] I. Alzugaray and M. Chli. 2019. Asynchronous Multi-Hypothesis Tracking of Features with Event Cameras. In *2019 International Conference on 3D Vision (3DV)*. 269–278. https://doi.org/10.1109/3DV.2019.00038

[3] C. Brandli, R. Berner, M. Yang, S. Liu, and T. Delbruck. 2014. A 240 × 180 130 dB 3 µs Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE Journal of Solid-State Circuits* 49, 10 (2014), 2333–2341. https://doi.org/10.1109/JSSC.2014.2342715

[4] C. Brändli, J. Strubel, S. Keller, D. Scaramuzza, and T. Delbruck. 2016. ELiSeD — An event-based line segment detector. In *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*. 1–7. https://doi.org/10.1109/EBCCSP.2016.7605244

[5] J. Canny. 1986. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8, 6 (1986), 679–698. https://doi.org/10.1109/TPAMI.1986.4767851

[6] A. Censi, J. Strubel, C. Brandli, T. Delbruck, and D. Scaramuzza. 2013. Low-latency localization by active LED markers tracking using a dynamic vision sensor. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 891–898. https://doi.org/10.1109/IROS.2013.6696456

[7] G. D. Evangelidis and E. Z. Psarakis. 2008. Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 10 (2008), 1858–1865. https://doi.org/10.1109/TPAMI.2008.113

[8] L. Everding and J. Conradt. 2018. Low-Latency Line Tracking Using Event-Based Dynamic Vision Sensors. *Frontiers in neurorobotics* 12 (2018), 4. https://doi.org/10.3389/fnbot.2018.00004

[9] G. Gallego, T. Delbruck, G.M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and et al. 2020. Event-based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), 1–1. https://doi.org/10.1109/tpami.2020.3008413

[10] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza. 2020. EKLT: Asynchronous Photometric Feature Tracking Using Events and Frames. *International Journal of Computer Vision* 128, 3 (2020), 601 – 618. https://doi.org/10.1007/s11263-019-01209-w

[11] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza. 2016. Low-latency visual odometry using event-based feature tracks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 16–23. https://doi.org/10.1109/IROS.2016.7758089

[12] K. Li, D. Shi, Y. Zhang, R. Li, W. Qin, and R. Li. 2019. Feature Tracking Based on Line Segments With the Dynamic and Active-Pixel Vision Sensor (DAVIS). *IEEE Access* 7 (2019), 110874–110883. https://doi.org/10.1109/ACCESS.2019.2933594

[13] R. X. Li, D. X. Shi, Y. J. Zhang, K. Y. Li, and R. H. Li. 2019. FA-Harris: A Fast and Asynchronous Corner Detector for Event Cameras. arXiv:1906.10925 [cs.CV]

[14] H. Liu, D. P. Moeys, G. Das, D. Neil, S. Liu, and T. Delbrück. 2016. Combined frame- and event-based detection and tracking. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2511–2514. https://doi.org/10.1109/ISCAS.2016.7539103

[15] B. D. Lucas and T. Kanade. 1981. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2* (Vancouver, BC, Canada) *(IJCAI'81)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 674–679.

[16] E. Mueggler, B. Huber, and D. Scaramuzza. 2014. Event-based, 6-DOF pose tracking for high-speed maneuvers. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2761–2768. https://doi.org/10.1109/IROS.2014.6942940

[17] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. 2017. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *The International Journal of Robotics Research* 36, 2 (2017), 142–149. https://doi.org/10.1177/0278364917691115

[18] E. Muggler, C. Bartolozzi, and D. Scaramuzza. 2017. Fast event-based corner detection. In *British Machine Vision Conference (BMVC)*. British Machine Vision Conference (BMVC), London, 2017., 1–8. https://doi.org/10.5167/uzh-138925

[19] J. B. Paul and D. M. Neil. 1992. Method for registration of 3-D shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, Paul S. Schenker (Ed.), Vol. 1611. International Society for Optics and Photonics, SPIE, 586 – 606. https://doi.org/10.1117/12.57955

[20] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck. 2014. Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output. *Proc. IEEE* 102, 10 (2014), 1470–1484. https://doi.org/10.1109/JPROC.2014.2346153

[21] T. Serrano-Gotarredona and B. Linares-Barranco. 2013. A 128 × 128 1.5% Contrast Sensitivity 0.9% FPN 3 µs Latency 4 mW Asynchronous Frame-Free Dynamic Vision Sensor Using Transimpedance Preamplifiers. *IEEE Journal of Solid-State Circuits* 48, 3 (2013), 827–838. https://doi.org/10.1109/JSSC.2012.2230553

[22] J. Shi and Tomasi. 1994. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 593–600. https://doi.org/10.1109/CVPR.1994.323794

[23] D. R. Valeiras, X. Clady, S. H. Ieng, and R. Benosman. 2018. Event-Based Line Fitting and Segment Detection Using a Neuromorphic Visual Sensor. *IEEE transactions on neural networks and learning systems* (September 2018). https://doi.org/10.1109/tnnls.2018.2807983

[24] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza. 2018. Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios. *IEEE Robotics and Automation Letters* 3, 2 (2018), 994–1001. https://doi.org/10.1109/LRA.2018.2793357

[25] A. Z. Zhu, N. Atanasov, and K. Daniilidis. 2017. Event-based feature tracking with probabilistic data association. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 4465–4470. https://doi.org/10.1109/ICRA.2017.7989517