Robust Feature Tracking in DVS Event Stream using Bézier Mapping

Hochang Seok Jongwoo Lim*
Department of Computer Science, Hanyang University, Seoul, Korea.
{hochangseok, jlim}@hanyang.ac.kr

Abstract

Unlike conventional cameras, event cameras capture the intensity changes at each pixel with very little delay. Such changes are recorded as an event stream with their positions, timestamps, and polarities continuously, thus there is no notion of 'frame' as in conventional cameras. As many applications including 3D pose estimation use 2D trajectories of feature points, it is necessary to detect and track the feature points robustly and accurately in a continuous event stream. In conventional feature tracking algorithms for event streams, the events in fixed time intervals are converted into the event images by stacking the events at their pixel locations, and the features are tracked in the event images. Such simple stacking of events yields blurry event images due to the camera motion, and it can significantly degrade the tracking quality. We propose to align the events in the time intervals along Bézier curves to minimize the misalignment. Since the camera motion is unknown, the Bézier curve is estimated to maximize the variance of the warped event pixels. Instead of the initial patches for tracking, we use the temporally integrated template patches, as it captures rich texture information from accurately aligned events. Extensive experimental evaluations in 2D feature tracking as well as 3D pose estimation show that our method significantly outperforms the conventional approaches.

1. Introduction

The event cameras, such as the Dynamic Vision Sensor (DVS) [5, 6], capture the intensity changes at individual pixels asynchronously. The intensity changes are recorded as a sequence of events, which contains the pixel x- and y-location, the timestamp of the change, and the polarity (positive or negative). Due to their asynchronous nature, there is no notion of 'frames' and a video is represented as a continuous event stream. In addition to low power consumption and low latency of the event cameras, this characteristic makes the event cameras ideal for sensing high-speed motion of the cameras as the changes are reported immediately with very little latency.

It would be ideal for the camera motion to be estimated at every event, but it is not realistic as individual events can be noisy and they do not contain enough information to constrain the camera motion. Moreover the traditional computer vision algorithms are not suited for handling event streams directly, thus conventional event-related approaches [23] make the ad hoc event images by stacking the events within certain time slots at their pixel locations. The generated event images can be used for feature detection, tracking, stereo matching, or optical flow computation, but the small motion of the camera in the time window cause misalignment and yields blurry images.

In this work, we propose a novel event alignment and feature tracking algorithm for event cameras (Figure 1). Compared to the previous work which stacks the events at their pixel locations [12, 13, 27] or along straight lines [7], the proposed algorithm aligns the events at individual patches using Bézier curves. As Bézier curves can model complex motion in the image more accurately, the aligned patch images have far less blur and more sharpness. Since the true patch motion is not known, the Bézier curves needs to be estimated from the event data. According to Gallego *et al*. [7] the alignment of events can be done by maximizing the variance of the event image patch, which is constructed by accumulating the count of events falling at the individual pixels.

By finding the Bézier curve that maximizes the event patch variance, we can track the local patch motion continuously, where the starting position of the patch at the next time window becomes the end point of the estimated Bézier curve. To compute reliable and accurate motion estimation, the feature tracks should be long and precise. Depending on the camera motion and scene texture, the edges parallel to the camera motion can be missing in the aligned image because there is little change in intensity in such regions, and using such patches for tracking is not desirable. Therefore long-term integration of events is necessary to capture the appearance of the patch correctly for reliable tracking. We propose a long-term feature tracking algorithm with temporal integration method of the events and Bézier align-

*Corresponding author.

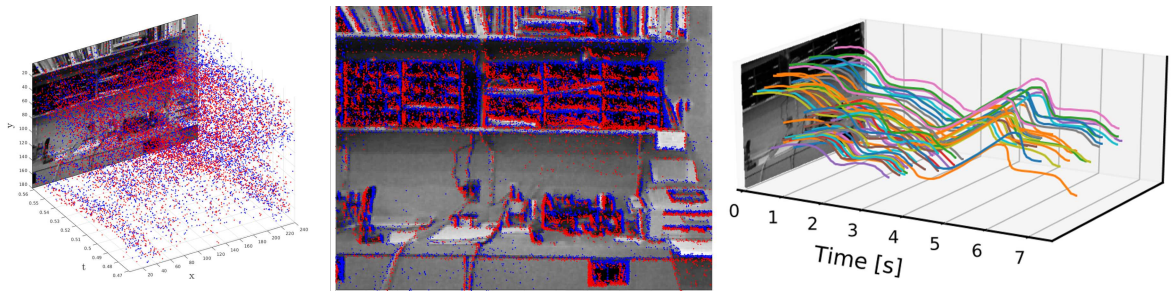


Figure 1. Visualization of an event stream in the space-time domain, colored according to the event polarity (left). Projection of aligned events (center) and the feature tracking result (right) by the proposed algorithm.

ment with the accumulated template. The proposed method solves the data association problem of the events to the feature and the motion estimation problem simultaneously.

The proposed algorithm is evaluated extensively using various synthetic event sequences. It is compared to the state-of-the-art event feature tracking algorithms in tracking 2D feature location and estimating the 3D camera poses from tracked features. This experimental validation shows the superior performance of the proposed event alignment and long-term feature tracking algorithm.

The contributions of this paper can be summarized as

(i) the event alignment algorithm using Bézier curves by variance maximization for short-term tracking (Figure 2 a),
(ii) the accurate temporal template update method for long-term tracking (Figure 2 b), and
(iii) extensive experimental evaluation of the proposed algorithm using synthetic and real datasets.

2. Related Work

Event cameras are introduced recently in the field of computer vision and robotics due to theirs ultra high frequency, robustness in high-dynamic range (HDR) environment, and low power consumption. However, because of the asynchronous event output makes it difficult to use the existing computer vision algorithm. There are many other research topics for event cameras: EMVS [20] for estimating depth using event and pose, event-based visual odometry [23], event-based visual-inertial odometry [22, 32], Ultimate SLAM [28] for estimating posture using event camera. For optical flow estimation, both conventional method [4] and deep learning method [30, 31] are actively studied. Image reconstruction methods for using existing computer vision algorithms are also being studied [10, 19, 24].

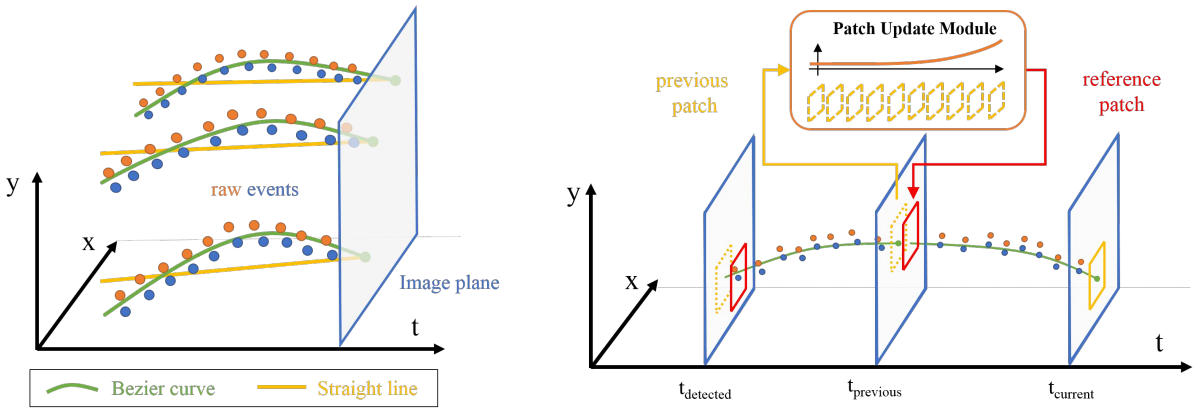
In particular, feature detection, matching, and tracking for event cameras, which are traditionally important problems in computer vision field, are of high interest. Analogous to the local feature detection and tracking algorithms for conventional images, such as Harris corners [9], SIFT [16], or the KLT tracker [3], researchers attempt to define local features in an event stream. The most straightforward way is to stack the events in a short time window at their pixel locations [27, 17, 23], and treat the stacked images just as the conventional images. This is simple and effective, but when the camera is moving the events from the same scene points do not map onto the same point. To alleviate this issue Gallego *et al*. [7] propose to estimate a straight line in the spatio-temporal XYT domain, and align the events along the line. When the camera motion is simple and steady, this approach works well, but for abrupt or irregular camera motions the event trajectories deviate from straight lines and the aligned images become blurry.

Although there are not many studies on event camera based feature matching yet, some interesting works about feature descriptor and matching for object recognition, like HOTS [15] and HATS [25], are presented. There has been several event feature tracking, using event camera only [29], [14], [2], [1] or using the events and conventional images at the same time [8]. Alzugaray *et al*. [2] proposes a fully asynchronous feature tracking method without discretizing the event stream, including new event feature detection approach. The tracker by Zhu *et al*. [29] stacks the events for a short time and computes new feature locations by ICP-based Expectation Maximization method. Since the event images by stacking are greatly influenced by the camera motion, the tracking performance is limited when the camera moves rapidly. On the other hand, Gehrig *et al*. [8] try to overcome this problem by using the gradient image of a frame which is less influenced by appearance changes due to motion. Our work is closely related to Zhu *et al*. [29], in a sense that it performs feature tracking using only an event camera. However, the main difference is that we consider the influence of the camera motion, similar to Gehrig *et al*. [8], but without using conventional intensity images.

3. Proposed Algorithm

In this paper, we propose the event alignment method using a 3D Bézier curve in XYT-domain and the long-term feature tracking algorithm. First we introduce the event alignment along a curve into a patch image, then define the variance of the image.

Suppose there exist a given a set of events $E \doteq \{\mathbf{e}_k\}$



(a) Event alignment using a Bézier curve (b) Reference patch update for tracking

Figure 2. (a) Our method aligns a set of events using a Bézier curve. (b) The reference patch is updated using the patch history volume with exponential decay for long-term tracking.

in the time window $[0, T]$, where $\mathbf{e}_k \doteq (x_k, y_k, t_k, p_k)^\top$, x_k, y_k are the pixel location, t_k is the timestamp, and $p_k \in \{-1, +1\}$ is the polarity of the event k. We define a parametric curve $C(t; \mathbf{x}_0, \theta)$ which returns the 2D location $\mathbf{x}_t \doteq (x_t, y_t)^\top$ for the given timestamp t, and passes the start point $\mathbf{x}_0$ at $t = 0$. θ determines the shape of the curve. For notational simplicity, we only consider one time window $[0, T]$ in this section, but it can be expanded to multiple time windows without loss of generality.

Then we can map the events to the 2D pixel coordinates as follows:

$$f(\mathbf{e}; \mathbf{x}_0, \theta) = \mathbf{x}_e - C(t_e; \mathbf{x}_0, \theta) + \mathbf{x}_0, \tag{1}$$

where $\mathbf{x}_e = (x_e, y_e)^\top$ is the pixel location and t_e is the timestamp of the event $\mathbf{e}$. This function computes the location offset of the event from the curve at the timestamp, and maps the event at the same offset from the start point $\mathbf{x}_0$, thus it maps the events along the curve. For example, all events exactly on the curve will be mapped to the same start point, and if the curve is a straight line parallel to the time axis (i.e., $C(t_e; \mathbf{x}_0, \theta) = \mathbf{x}_0$), all events are mapped to their original pixel locations.

The pixel values of the event image patch aligned along the curve is the number of events falling at the pixel locations. Formally the event image patch $I(\mathbf{x})$ is defined as

$$I(\mathbf{x}; \theta) = \left| \left\{ \mathbf{e} \in E \mid f(\mathbf{e}; \mathbf{x}_0, \theta) = \mathbf{x} \right\} \right|. \tag{2}$$

Note that we map the bilinearly interpolated value for real coordinate $\mathbf{x}$. In the following subsections, we formulate the variance maximization using a Bézier curve, and the long-term feature tracking algorithm utilizing the variance maximization.

3.1. Variance Maximization for a Bézier Curve

Previous methods generate an event patch by stacking the event using simply their pixel locations, or aligned along a straight line. Although the straight lines are easy to estimate, the events that change their positions rapidly cannot be modeled by a straight line as shown in Figure 2 (a). Note that this case happens quite frequently and we will discuss this issue in more detail in the following section.

The quadratic Bézier curve is parameterized with three control points $\mathbf{p}_0$, $\mathbf{p}_1$, and $\mathbf{p}_2$ as

$$\begin{aligned} B(t; \mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2) = (1-t)\big((1-t)\mathbf{p}_0 + t\mathbf{p}_1\big) \\ + t\big((1-t)\mathbf{p}_1 + t\mathbf{p}_2\big), \quad 0 \le t \le 1. \end{aligned} \tag{3}$$

We use the quadratic Bézier curve because it can represent simple curves with minimal number of parameters. In our case the start point is set to the end point of the curve of the previous time window, and we assume that the timestamp of the control points are fixed at 0, $T/2$, and T, and the event timestamp t_e is normalized by dividing by T. Thus there are only 4 location parameters in 2 control points to be estimated from the event stream, i.e., C_{qb} is the quadratic Bézier curve function for $\theta \doteq (x_1, y_1, x_2, y_2)^\top$.

Gallego *et al.* [7] propose to use the alignment path that maximizes the variance of the patch values. The variance of an event image patch is defined as

$$Var(I; \theta) = \frac{1}{N} \sum_{\mathbf{x}} \big(I(\mathbf{x}; \theta) - \mu(I; \theta)\big)^2 \tag{4}$$

$$\mu(I; \theta) = \frac{1}{N} \sum_{\mathbf{x}} I(\mathbf{x}; \theta), \tag{5}$$

where N is the number of pixels of the patch. Intuitively the monotonic images have 0 variance, and the images with high contrast has high variance. In our application maximizing the patch variance is maximizing the contrast, and minimizing the blur by misalignment. The Bézier curve for the given events can be estimated by

$$\theta^* = \arg\max_{\theta} \; Var(I; \theta). \tag{6}$$

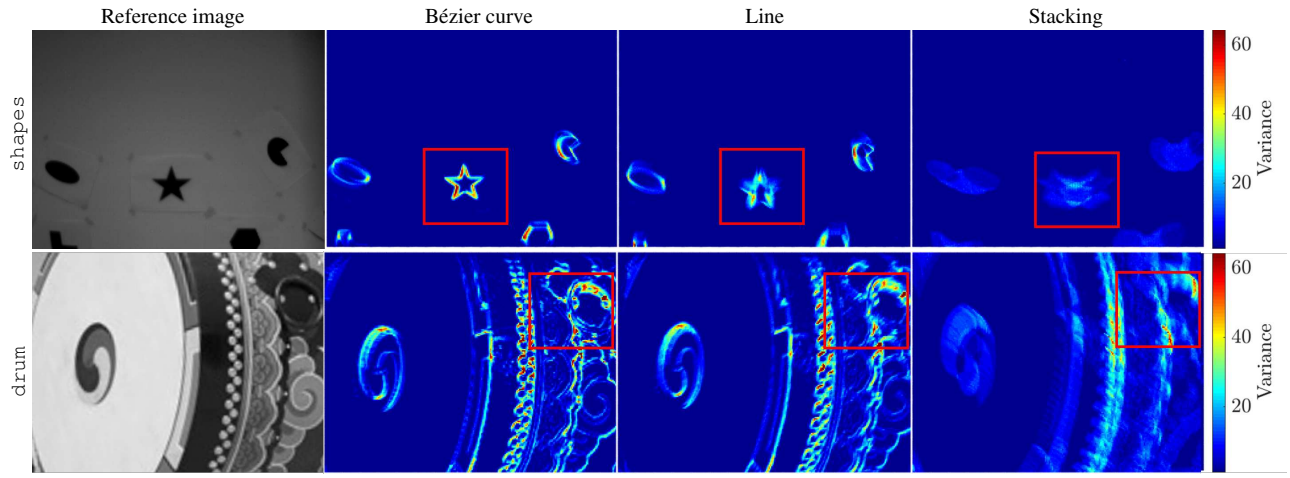


Figure 3. Event alignment test results. Right three columns show the aligned event images by the proposed Bézier curves, lines, and stacking respectively. Color indicates the variance at each pixel.

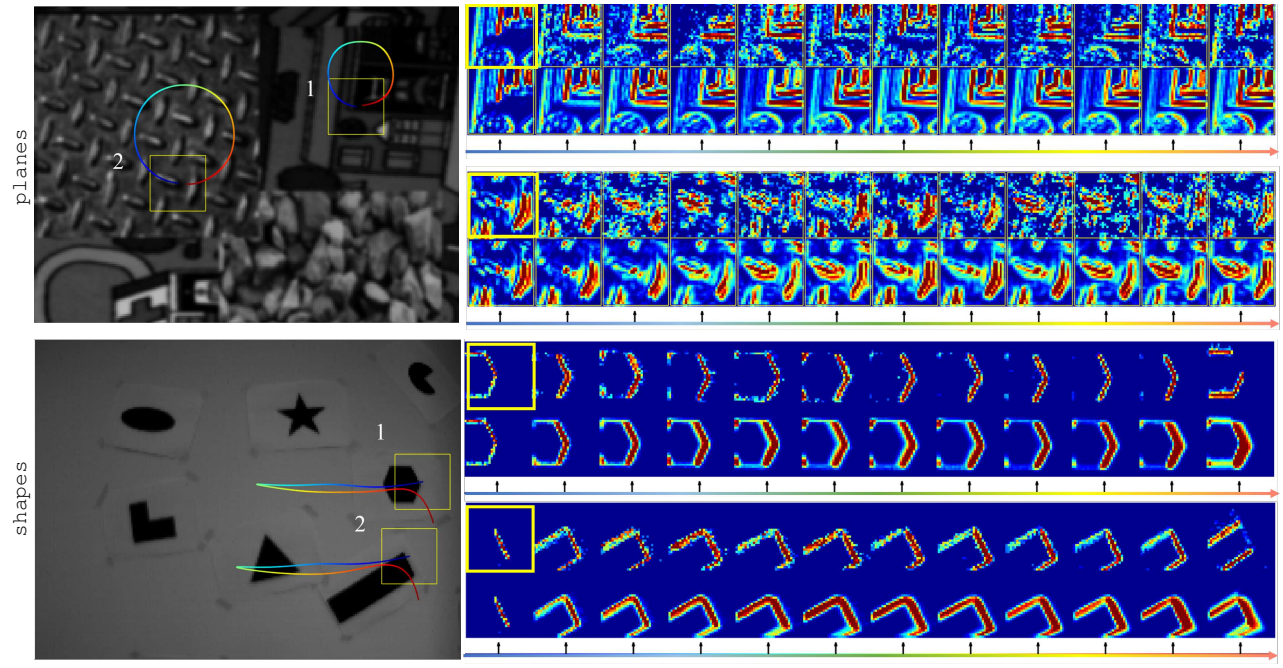


Figure 4. Temporal template update results of `planes` and `shapes` datasets. The feature trajectories are shown in the reference grayscale images, and the aligned event patches (upper row) and the integrated templates (lower row) along the time windows are shown on the right. Note that the aligned event patches are noisy and some edges are missing depending on the camera motion, whereas the integrated templates learn and maintain all scene structures. This property helps more robust feature tracking in challenging situations.

Datasets	Mean Variance		
	Bézier curve	Line	Stacking
`shapes`	**22.30**	20.21	7.77
`drum`	**79.08**	73.15	29.18

Table 1. Quantitative evaluation of event alignment. The mean variance of the event images aligned by the proposed method is much higher than those by the other algorithms.

3.2. Template Update for Long-term Tracking

As in the previous subsection, the event feature is defined as a patch of the aligned event image. Fitting a Bézier curve to the event stream gives the 2D motion of the patch for the given time window as well as the aligned 2D event image patch. One may perform feature tracking in the event stream just by fitting Bézier curves consecutively, but this has several drawbacks. First there is no reference patch (template) for the tracked feature, thus the tracker can easily drift from the target position. Second the motion estimation is very sensitive to the choice of temporal window. If the integration time window is too small, the event trajectories are close to straight lines but the number of event patches to track is too large (in conventional video, too high frame-per-

second), and the integrated image patch would not contain enough texture. For robust feature tracking it is important to have a patch with enough texture. If the integration window is too large, the 2D motion in the window may not be represented with a quadratic Bézier curve, which degrades the tracking quality.

To fully utilize the modeled patch appearance, when computing the patch variance we use both the values in the template patch and those integrated in the current event stream,

$$Var_T(I;\theta,I_T) = \frac{1}{N}\sum_{\mathbf{x}}\left(I(\mathbf{x};\theta) + I_T(\mathbf{x}) - \mu_T(I;\theta)\right)^2 \quad (7)$$

$$\mu_T(I;\theta) = \frac{1}{N}\sum_{\mathbf{x}}\left(I(\mathbf{x};\theta) + I_T(\mathbf{x})\right), \quad (8)$$

where $I_T(\mathbf{x})$ is the learned template patch. The best Bézier parameter θ^* for the current event stream is estimated by maximizing Var_T in the same way as Equation 6.

For robust feature tracking the tracker needs to know how each patch looks like, to prevent drift and to find the accurate motion. Depending on the scene texture and camera motion, some edges may not be observable when the camera moves along the edge direction. Such edges may be revealed when the camera moves differently. Therefore the reference template patch needs to be updated continuously, but keep most of important visual information. In this process, we use a simple temporal update model with exponential decay. In every iteration, the current patch is stacked to maintain the history of tracked patches. Then we generate the new reference template patch by integrating the stack of patch history. The new template patch is calculated as,

$$I_T^{(new)}(\mathbf{x}) = \sum_{\mathbf{i}}^{\mathbf{n}} I^i(\mathbf{x}) * \mathbf{e}^{\rho(n-i)}, \quad (9)$$

where ρ is the update rate for the template image, $\mathbf{n}$ is the number of stacked patches. To update the appearance of template changes along the time, we multiplied the large exponent values to the more recent patches. Through this simple method, long-term feature tracking is possible.

As shown in Figure 4, the temporally integrated template image contains richer texture which reflects original scene structure faithfully, whereas the aligned patches are noisy and miss some of edges.

To initialize patch locations for feature tracking, we align all events in the entire image for a short time period to construct an aligned event image, and run the Harris corner detector. We use this simple approach to bootstrap the proposed feature tracker, and we plan to investigate better ways for feature detection in event streams.

4. Experiments

We demonstrate the proposed event alignment and feature tracking algorithm with synthetic and real datasets. Since the implementations of other tracking algorithms are not publicly available, we compare the proposed algorithm with Zhu *et al*. [29]. In the first experiment, we compare the alignment performance with Bézier curves to with straight lines using the same variance maximization. In the second experiment, our feature tracking algorithm is compared to Zhu *et al*. quantitatively in terms of tracking accuracy and feature age for both synthetic and real sequences, and also the qualitative comparison is shown. Finally, we show the 3D camera pose estimation accuracy using the tracked features since the main application of event feature tracking is camera pose estimation such as visual SLAM or visual-inertial odometry. The variance maximization is implemented in Matlab by using the built-in optimization function, and the initial Bézier curve is initialized as the line parallel to the temporal axis.

4.1. Test Datasets

To test our work, three real sequences from the public dataset and six synthetic sequences are used. Among the real sequences in [18], we choose a simple black and white scene (`shapes`) and two highly textured scenes (`posters`, `boxes`). The six synthetic sequences are called `planes`, `desk`, `gate`, `wall`, `roof`, and `drum`. In all sequences, the ground-truth feature trajectories for any feature points are available since we rendered both depth and camera poses. The `planes` sequence in [18] is synthetically captured by a camera moving in a circle without any rotation in front of three planes in different depth. The other five synthetic sequences are generated in-house using ESIM simulator [21] while the camera moves rapidly along random 3D spline trajectories seeing a single plane, to generate more realistic complex event streams. All rendered datasets and rendering codes will be made public when the paper is published.

4.2. Event Alignment Evaluation

In this section, we evaluate the performance of the proposed event alignment algorithm using Bézier curves. For quantitative evaluation, the variance of the pixel values of the aligned image is compared since higher variance represents better event alignment. We compare three algorithms, by stacking [29, 23], along a straight line [7], and along a Bézier curve (proposed). The experiment is performed using event streams sampled from the `shapes` and `drum` sequences. Each event streams consists of 0.5 million events, and the whole image of 240x180 is aligned by the estimated poses. The quantitative and qualitative experimental results are shown in Table 1 and Figure 3.

As shown in Table 1, the mean variance of the proposed method is about 23.6% and 279.02% larger than those of the straight line and the stacking methods respectively. The red boxes of Figure 3 shows the alignment quality, as the

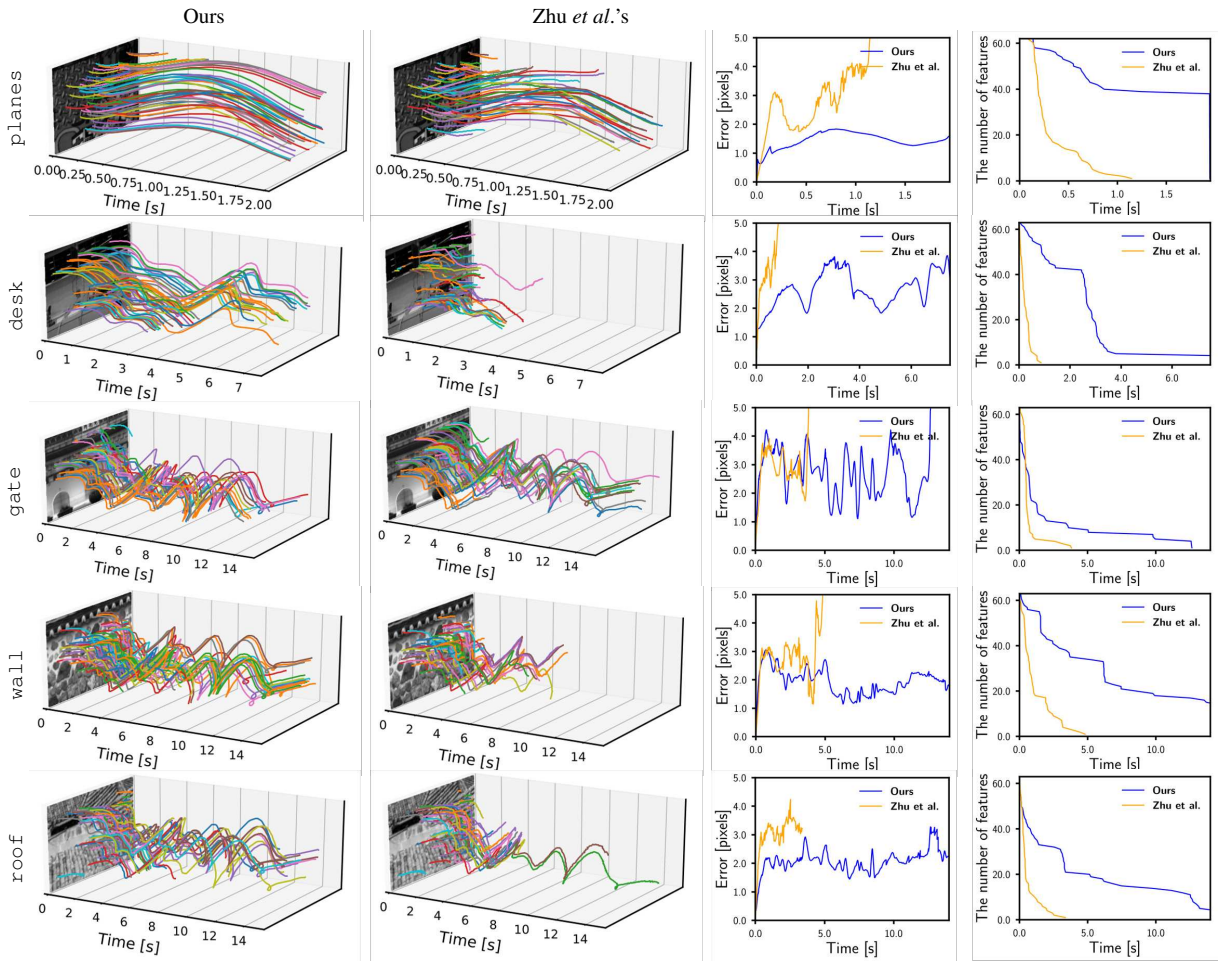


Figure 5. Qualitative results of feature tracking in synthetic dataset. The left two figures show the trajectories of tracked features in the spatio-temporal domain, and the right graphs show the average position error and number of tracked features within 5-pixel threshold. Note that our algorithm tracks the features more robustly and the error is maintained at lower level.

Datasets	thr [px]	RMSE [px]			Tracking age [s]			Tracking length [px]			total time [s]
		Zhu *et al.*	Ours(w/o TU)	Ours	Zhu *et al.*	Ours(w/o TU)	Ours	Zhu *et al.*	Ours(w/o TU)	Ours	
planes	<10	3.66	2.11	**1.52**	0.68	0.71	**1.43**	57.42	60.31	**110.06**	2
	<5	2.00	1.78	**1.41**	0.33	0.32	**1.43**	27.74	25.66	**107.43**	
	<3	1.25	1.56	**1.24**	0.19	0.20	**1.26**	15.64	18.27	**96.11**	
desk	<10	4.29	3.15	**2.38**	0.61	0.62	**4.4**	47.43	51.33	**183.76**	7.5
	<5	2.28	2.14	**1.72**	0.24	0.51	**2.73**	26.82	28.93	**117.15**	
	<3	1.37	1.41	**1.26**	0.14	0.31	**1.29**	17.84	21.02	**69.78**	
gate	<10	4.87	3.88	**3.62**	2.54	2.43	**4.95**	118.99	47.43	**222.83**	14
	<5	2.81	2.96	**2.61**	0.62	0.67	**2.05**	49.26	53.10	**101.74**	
	<3	**1.25**	1.41	1.30	0.37	0.34	**0.40**	35.94	34.75	**39.59**	
wall	<10	3.84	2.46	**2.15**	2.22	2.41	**6.86**	115.69	106.55	**315.25**	14
	<5	2.35	2.58	**2.03**	1.13	1.27	**6.5**	65.50	68.80	**298.73**	
	<3	**1.26**	2.01	1.60	0.51	0.77	**1.89**	39.48	40.56	**101.60**	
roof	<10	4.31	4.67	**2.35**	1.47	0.67	**4.76**	77.43	72.11	**216.11**	14
	<5	2.23	2.34	**1.94**	0.56	0.40	**4.27**	40.94	40.08	**193.10**	
	<3	**1.22**	1.56	1.43	0.30	0.27	**2.06**	28.31	22.79	**98.32**	
drum	<10	4.31	4.51	**2.42**	1.69	1.03	**5.20**	78.23	54.21	**241.39**	6
	<5	2.55	2.43	**2.16**	0.60	0.05	**4.04**	38.17	22.10	**188.82**	
	<3	1.90	1.76	**1.65**	0.23	0.21	**0.43**	22.9	17.5	**41.82**	

Table 2. Quantitative evaluation of feature tracking in synthetic seqeunces. For each dataset, the root-mean-squared error, the average tracking length and the average feature age of the feature tracks within the error threshold (thr) are compared. In addition, we test ours without the template update(TU) for ablation study. Overall, our algorithm tracks the feature much longer than Zhu *et al.* [29], and the error is smaller considering that the numbers, lengths and ages of our features are more than Zhu *et al.*'s. However, Our method without TU shows similar performance to Zhu *et al.*

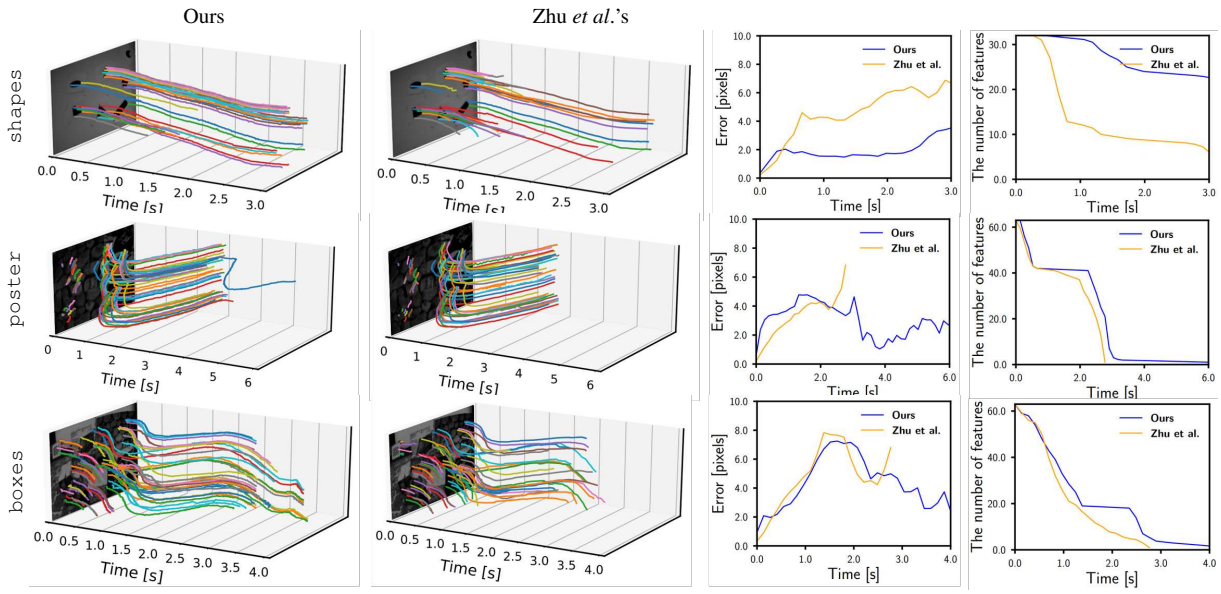


Figure 6. Qualitative results of feature tracking in real dataset. The left two figures show the trajectories of tracked features in the spatio-temporal domain, and the right graphs show the average position error and number of tracked features within 10-pixel threshold.

Datasets	thr [px]	RMSE [px]		Tracking age [s]		Tracking length [px]		total time [s]
		Zhu *et al.*	Ours	Zhu *et al.*	Ours	Zhu *et al.*	Ours	
shapes	<10	4.79	**2.03**	1.37	**2.77**	38.66	**142.05**	3
	<5	2.98	**1.8**	0.85	**2.6**	27.26	**131.503**	
	<3	1.72	**1.4**	0.51	**1.85**	19.06	**86.62**	
poster	<10	3.81	**3.00**	1.72	**1.96**	171.84	**212.52**	6
	<5	2.69	**2.31**	**1.37**	1.32	**130.34**	124.68	
	<3	**1.54**	1.83	**0.91**	0.41	**87.24**	50.94	
boxes	<10	4.76	**4.20**	1.07	**1.36**	86.57	**126.89**	4
	<5	2.51	**2.36**	0.65	**0.72**	51.86	**60.45**	
	<3	1.65	**1.59**	0.44	**0.50**	33.50	**35.36**	

Table 3. Quantitative evaluation of feature tracking in real sequences. We use same evaluation metric as shown in Table 2. In the shapes and boxes sequence, our method shows better results than Zhu *et al.* in all metrics. Zhu *et al.* shows slightly better performance than ours in poster sequence, except for the error threshold 10. Refer to the text for detailed discussion.

blurring is much smaller in Bézier results. Both quantitative and qualitative evaluation shows that the proposed method significantly outperforms the conventional methods.

4.3. Evaluation of Feature Tracking

In this section we evaluate the proposed feature tracking algorithm in various settings. In quantitative evaluation, the accuracy is evaluated by the Root Mean Squared Error (RMSE) between the estimated feature locations and the ground truth locations, and the persistency of the feature tracker is measured by the tracking length and the feature age. The tracking length and tracking age mean the pixel length of feature trajectory and the time of tracked feature, respectively. In this experiment, we use three error thresholds, 3, 5 and 10 pixels, to reject the erroneous features from evaluation.

To show the performance of our method against the previous work, we compare our results to [29] which is one of the state-of-the art event feature tracker. In this experiment, we use the authors' public implementation*, and tune its best parameters for our dataset. Although [29] are not the latest work, it shows good performance compared to [1, 8] and there is no public implementations of [1] and [8], thus we use Zhu *et al.*, as the baseline. For comparison, we modified the authors' code to track only initial features without feature re-detection.

In both methods, we use same 63 Harris corners [9] and its corresponding 31×31 patches extracted at the initial locations. The corners are extracted uniformly from the event-aligned image. The temporal update parameter ρ in equation 9 is fixed as 0.05 in whole sequences. The results of feature tracking evaluation are given in Table 2, 3 and Figure 5, 6. The `planes` sequence is an easy one as due to its simple camera motion there is little change in scale or orientation. Zhu *et al.* [29] tracks the features for less than 1 second, which is only 34% of the whole sequence regardless of the threshold. On the other hand, the proposed method

*https://github.com/alexzzhu/event_feature_tracking

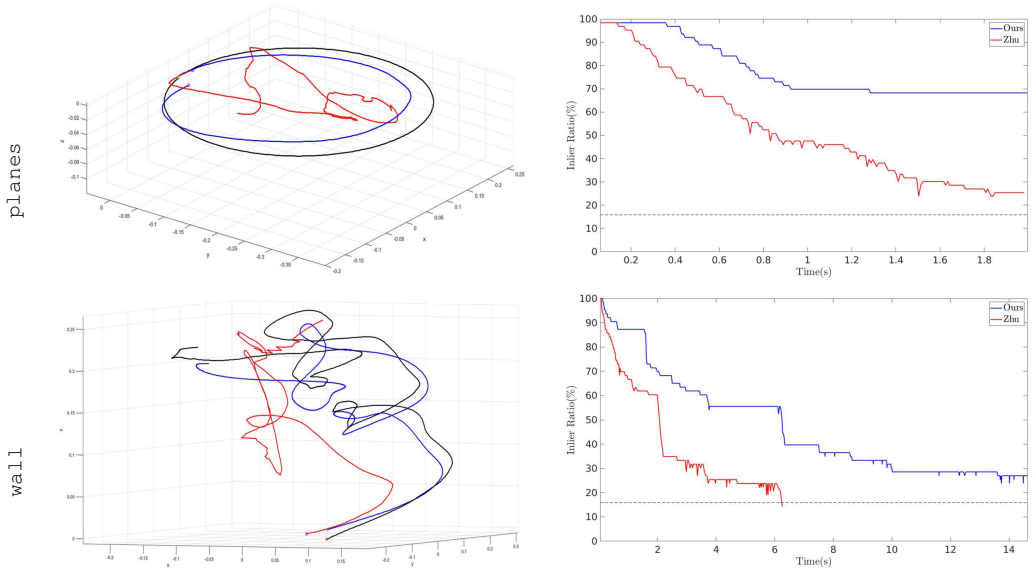


Figure 7. 3D pose estimation results in `planes` and `wall` datasets. The ground-truth trajectory (black) and the trajectories estimated by ours (blue), and Zhu *et al.*'s [29] (red) are shown on the left. The right plot shows the inlier ratio of tracked features. Note that our trajectories are much better than Zhu *et al.*'s and the inlier ratio is much higher.

tracks the features for more than 1.2 seconds (≃70%). Especially when the threshold is 3 pixels, the average feature age is 6 times higher.

To evaluate long-term feature tracking and robustness, the other synthetic test datasets are longer and captured with random rapid motions. As shown in Table 2, the average feature age of the proposed method is 4 times longer than that of Zhu *et al*. Note that the RMSE are computed only for the features within the threshold, thus the error values are roughly similar, whereas the feature age of the proposed method is much longer. The temporal template update method plays important role in tracking features longer by constantly updating the template with most recent events.

Figure 6 and Table 3 Compared to the synthetic sequences, the event density of the real sequence is much lower due to higher intensity threshold. In `poster`, the results are not stable as most features are lost in the middle as they goes outside of the image. The proposed algorithm achieves better performance except the `poster` sequence.

4.4. Evaluation with 3D Pose Estimation

In this experiment, we use the tracked feature trajectories to estimate 3D poses of the camera. Since the quality of tracked feature positions is critical in motion computation, this evaluation shows the effect of the improved feature tracking to the applications such as visual SLAM or visual(-inertial) odometry. We initialize the 3D feature points by the ground-truth depth, and run the standard P3P-RANSAC algorithm [11] and pose-only bundle-adjustment [26].

Figure 7 shows that the pose estimation with the features by Zhu *et al*. fails shortly after the start, while the poses from the proposed method are successfully constructed and are close to the ground-truth. It also can be shown that the number of pose inliers of the proposed method is much higher. In this experiments, we show that our feature tracking result can be used to estimate the pose accurately not only in the simple circling motion, but even in the case which contains complicated 6-DOF motion, such as `wall` case.

5. Conclusion

In this paper we propose a novel event alignment and feature tracking algorithm, which effectively aligns the event stream generated in a three-dimensional space-time space and simultaneously solves the data association problem intrinsically. The variance maximization along a Bézier curve aligns the events more accurately than the previous methods using stacking or along a straight line. The proposed local feature tracking algorithm constructs the template image patch by temporally integrating the events and estimates feature motion based on the above event alignment with the template. Extensive experiments show the proposed algorithm's superior performance compared to the state-of-the-art event alignment and feature tracking algorithms.

Acknowledgement

This research was supported by Next-Generation Information Computing Development Program through National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT(NRF-2017M3C4A7069369), the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (NRF-2019R1A4A1029800).

References

[1] I. Alzugaray and M. Chli. Ace: An efficient asynchronous corner tracker for event cameras. In *2018 International Conference on 3D Vision (3DV)*, pages 653–661. IEEE, 2018. 2, 7

[2] I. Alzugaray and M. Chli. Asynchronous corner detection and tracking for event cameras in real time. *IEEE Robotics and Automation Letters*, 3(4):3177–3184, 2018. 2

[3] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004. 2

[4] P. Bardow, A. J. Davison, and S. Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 884–892, 2016. 2

[5] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck. A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014. 1

[6] C. Brandli, L. Muller, and T. Delbruck. Real-time, high-speed video decompression using a frame-and event-based davis sensor. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 686–689. IEEE, 2014. 1

[7] G. Gallego, H. Rebecq, and D. Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3867–3876, 2018. 1, 2, 3, 5

[8] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza. Asynchronous, photometric feature tracking using events and frames. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 750–765, 2018. 2, 7

[9] C. G. Harris, M. Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988. 2, 7

[10] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison. Simultaneous mosaicing and tracking with an event camera. *J. Solid State Circ*, 43:566–576, 2008. 2

[11] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *CVPR 2011*, pages 2969–2976. IEEE, 2011. 8

[12] J. Kogler, C. Sulzbachner, M. Humenberger, and F. Eibensteiner. Address-event based stereo vision with bio-inspired silicon retina imagers. In *Advances in theory and applications of stereo vision*. IntechOpen, 2011. 1

[13] J. Kogler, C. Sulzbachner, and W. Kubinger. Bio-inspired stereo vision system with silicon retina imagers. In *International Conference on Computer Vision Systems*, pages 174–183. Springer, 2009. 1

[14] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza. Low-latency visual odometry using event-based feature tracks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 16–23. IEEE, 2016. 2

[15] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman. Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1346–1359, 2017. 2

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 2

[17] E. Mueggler, C. Bartolozzi, and D. Scaramuzza. Fast event-based corner detection. In *British Machine Vis. Conf.(BMVC)*, volume 1, 2017. 2

[18] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149, 2017. 5

[19] G. Munda, C. Reinbacher, and T. Pock. Real-time intensity-image reconstruction for event cameras using manifold regularisation. *International Journal of Computer Vision*, 126(12):1381–1393, 2018. 2

[20] H. Rebecq, G. Gallego, and D. Scaramuzza. Emvs: Event-based multi-view stereo. Technical report, 2016. 2

[21] H. Rebecq, D. Gehrig, and D. Scaramuzza. Esim: an open event camera simulator. In *Conference on Robot Learning*, pages 969–982, 2018. 5

[22] H. Rebecq, T. Horstschaefer, and D. Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In *British Machine Vis. Conf.(BMVC)*, volume 3, 2017. 2

[23] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2017. 1, 2, 5

[24] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. Events-to-video: Bringing modern computer vision to event cameras. *arXiv preprint arXiv:1904.08298*, 2019. 2

[25] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1731–1740, 2018. 2

[26] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustmenta modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999. 8

[27] V. Vasco, A. Glover, and C. Bartolozzi. Fast event-based harris corner detection exploiting the advantages of event-driven cameras. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4144–4149. IEEE, 2016. 1, 2

[28] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, 3(2):994–1001, 2018. 2

[29] A. Z. Zhu, N. Atanasov, and K. Daniilidis. Event-based feature tracking with probabilistic data association. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4465–4470. IEEE, 2017. 2, 5, 6, 7, 8

[30] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. Ev-flownet: self-supervised optical flow estimation for event-based cameras. *arXiv preprint arXiv:1802.06898*, 2018. 2

[31] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. *arXiv preprint arXiv:1812.08156*, 2018. 2

[32] A. Zihao Zhu, N. Atanasov, and K. Daniilidis. Event-based visual inertial odometry. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2