Fully Event-Inspired Visual Odometry

Nurlanov Zhakshylyk, Korobov Nikita

Technische Universität München

Abstract

Event cameras are the new type of the differential sensors allowing to capture the change of the contrast in each pixel asynchronously at the high dynamic range and high-speed motion. In this work, we investigate the capability of the event-based methods in the problem of visual odometry. We implement 3 independent algorithms: 1) Event-based Feature Tracker; 2) Monocular Visual Odometry based on feature tracks; 3) Motion Compensation of event images. Moreover, we introduce a new loss function for Unsupervised Motion Compensation called Edge Maximization. Finally, we combine all the algorithms into Fully Event-Inspired Visual Odometry. To the best of our knowledge, this is the first attempt to build the Visual Odometry exploiting only Events. In the end, we provide a comparison with the existing event based Visual Odometry and tracking frameworks.

Keywords: Event cameras, Event-based Feature Tracking, Visual Odometry, Motion Compensation, Contrast Maximization, Edge Maximization.

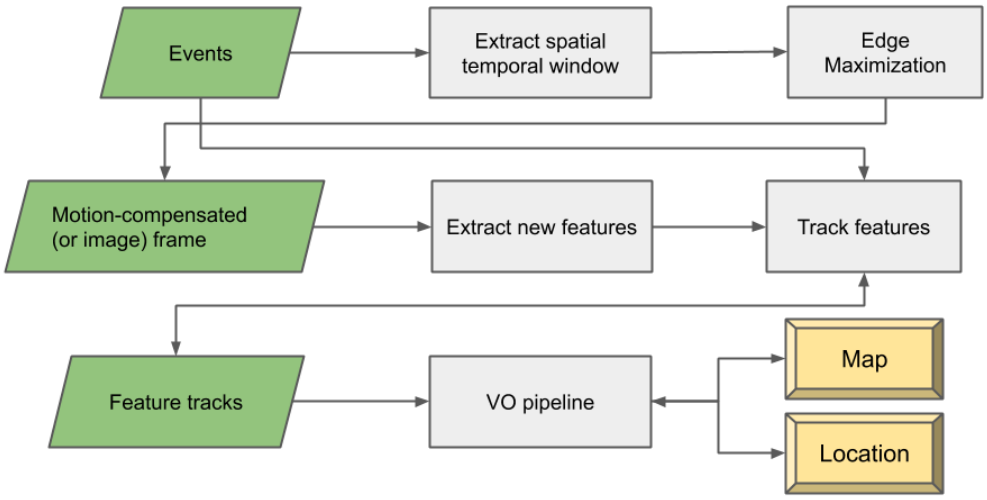


Figure 1: The high-level overview of the Fully Event-Inspired Visual Odometry pipeline. At the first stage, the upcoming events are collected into the spatial-temporal windows, and then Motion Compensation of events via Edge Maximization approach is applied. The second stage is responsible for the Feature Tracking using events and the motion-compensated (or regular) frames. The third stage is the implementation of the monocular Visual Odometry (VO) pipeline based solely on the feature tracks. Overall, the pipeline can work fully based on events producing a reasonable map and trajectories.

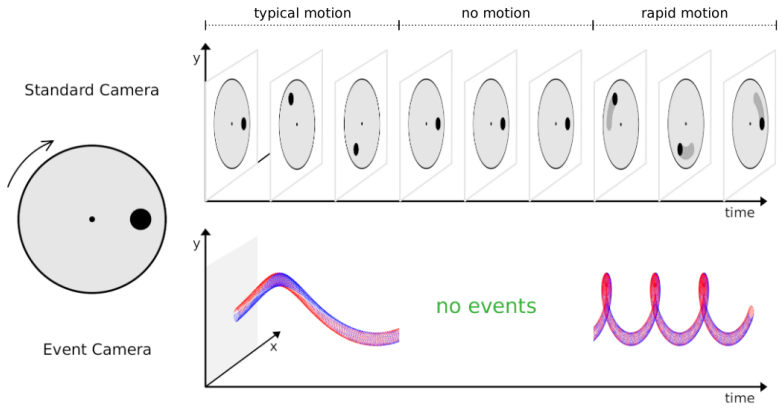


Figure 2: The figure is taken from [3]. Comparison of the output of the event camera with the standard frame-based camera capturing rotating disk. For the standard camera (top row) the images are given with the equal time differences in between even for non-rotating disk and rapid motion appears to have a motion blur. For the event camera (bottom row) the events are produced only in the location of the black circle when there is a rotation. Blue and red events mean negative and positive polarity (intensity change), respectively.

1 Introduction

Understanding the absolute and/or relative change of the pose of the camera body is the crucial component of many robotics and AR/VR applications. One of the popular types of environment perceiving sensors is an optical camera. Methods of localization using visual information are widely studied in the literature [1]. The usage domain of the optical cameras themselves is limited. Usually, algorithms of Visual Odometry (VO) and Visual SLAM (VSLAM) fail with the over/under-exposed images and scenes with fast motion.

Meanwhile, the new bio-inspired Dynamic and Active-pixel Vision Sensor (DAVIS), or so-called event-camera [2] has been developed to track the differential changes of the light intensity. It consists of the set of independent pixels that produce an "event" whenever the change of the logarithm of the light falling into the pixel exceeds some threshold. So the "event" is represented by the object consisting of the pixel position $\vec{x}$, the polarity of the relative change $p \in \{+1, -1\}$ and timestamp t when the threshold has been exceeded. Events are triggered asynchronously and can be produced at high frequency (more than 10^6 events per second). A comparison of the event camera with the regular camera on the example is shown in Fig. 2. Event cameras have lower power consumption, higher dynamic range and can capture faster motion than regular optical cameras. The usage of the data captured by the event cameras allows us to extend the limits of the domain of robotics applications and make localization algorithms more reliable.

In this work, we discover the algorithms of Feature Tracking, VO, and Motion Compensation for event images using events and regular images.

1.1 Related works

Feature tracking

The feature association is an important step in VO and VSLAM algorithms. The existing methods can be divided into a local and global association. The global methods normally use extracted feature descriptors such as ORB [4], SURF [5], etc. to search for the closest feature in the database of the previously observed descriptors. These methods do not

assume any sequential order of data and can find a matching pair reliably in the large database of images. This characteristic allows us to perform re-localization and loop closure in SLAM systems.

The local feature association algorithms (trackers) assume the sequential order of image frames and detected features (detections) on them accordingly, so the association of features between two consecutive detections can be performed. One of the most popular trackers is the KLT tracker [6]. However, the algorithm is developed for the frame-based cameras and it can not track in a blind time between the frames and fails during the association if the features can not be detected because of the motion-blurred or overexposed frames. Some of the recent studies on event cameras are aimed to overcome this problem. In [7] features are tracked using Canny edges detected around Harris corners. This image of patches is aligned using ICP to the image of integrated events, as the cause of triggering events can be imagined as moving edges. This approach has several drawbacks: the strength of the edge is not taken into account and correspondences for ICP need to be established. These problems are addressed in the work by Gehrig D. et al. [8].

The aim of this report is to implement the algorithm done by [8] with some small changes. This feature association algorithm is local and asynchronous.

Event based VO

A lot of studies are made in the last years on the visual odometry on event cameras. The first attempt to perform 6-DoF motion estimation for event cameras without depth information is made by Kueng, Beat, et al. [7]. The feature tracker used is considered in the previous subsection and the VO framework is the Semi-direct VO as in [9]. This work shows worse performance than state-of-the-art VO methods, but it proves the possibility to build a VO framework based on the events and images only. The more recent works on visual odometry [10, 11] and SLAM systems [12] include fusion of IMU, what makes the framework more reliable than standard visual odometry in scenes with extremely fast motion and poor light conditions.

In this work we generally follow the approach of Kueng, Beat, et al. [7], but use another feature tracker and different VO framework.

Motion compensation

In previous works [7, 8] on event-based trackers the authors use images from an RGB-cameras for feature detection, and then track features with the help of upcoming events. However, in case of extremely fast motion or overexposed scene (See Figures 3, 4) the regular images are not reliable for detecting features on them. It is shown, for example, in [10]. That is why there is a need to be able to eliminate the use of regular images in such extreme cases and to use event images instead.

The work of Rebecq et al. [10] shows that motion-compensated event images are sufficient to detect and track the features on them for further Visual-Inertial Odometry. However, the approach heavily relies on measurements from IMU while performing the motion-correction of the events. Gallego et al. in their paper [13] propose an approach for motion-compensation of events via the Contrast Maximization framework. It means that for detecting and tracking visual features one may need motion-corrected event images, which can be obtained solely using events.

In addition to the Contrast Maximization framework [13] we propose a new loss function, which has not been discovered in [14]. The loss function utilizes an intuitive geometric interpretation of the Structure Tensor [15] and advances the visibility of edges, so it is

Figure 3: Motion blurred image.

Figure 4: Overexposed scene.

reasonable to call it Edge Maximization.

1.2 Outline

The rest of the report is organized as follows: the feature tracker is described in Section 2.1. Then, the Visual Odometry based on the feature tracker is explained in Section 2.2. In Section 2.3 different methods of motion compensation for event images are considered, and all of the above-presented methods are combined in one event only inspired visual odometry in Section 2.4. Section 3 describes the implementation details. Experiments are presented in Section 4.

2 Main Approach

2.1 Event-based Tracker

Feature trackers based on the feature appearances on the regular images usually fail in cases of a fast camera or foreground motion and poor light conditions of the scene. Meanwhile, event cameras have a higher dynamic range and lower time of the pixel response to the movement compared to the regular cameras. These characteristics may be exploited to build a feature tracker that works well even with the fast camera motion and in the dark scenes.

As an event-camera pixel triggers once the change of the logarithm of the light falling into it since the last trigger time exceeds some threshold C, the change of a pixel $\vec{x}$ intensity over the $\Delta\tau$ at t can be expressed as

$$\Delta I(\vec{x}, t) = \int_{t}^{t+\Delta\tau} C f(\vec{x}, t) dt, \tag{1}$$

where $f(\vec{x}, t)$ equals to either $p \in \{+1, -1\}$ if there is an event in position $\vec{x}$ at time t or 0 otherwise. We will call $\Delta I(\vec{x}, t)$ integrated intensity at time $t_{int} = t + \Delta\tau$, because it is a sum of all the events falling into the pixel over the time frame.

At the same time we express the pixel intensity change for the regular image as

$$\Delta I(\vec{x}, t) = I(\vec{x}, t) - I(\vec{x}, t - \Delta\tau) = \frac{\partial I}{\partial t}\Delta\tau. \tag{2}$$

From the constant brightness assumption of the particular feature on the regular image

$I(\vec{x}(t), t) = \text{const}$, it follows that

$$\frac{\partial I}{\partial t} + \nabla I \frac{d\vec{x}}{dt} = 0, \tag{3}$$

$$\Delta \hat{I}(\vec{x}, v, t) = -(\nabla I \cdot v(\vec{x}))\Delta\tau, \tag{4}$$

where $v(\vec{x}) = \frac{d\vec{x}}{dt}$ is the optical flow. We will call $\Delta \hat{I}(\vec{x}, v, t_{im})$ predicted intensity, because we are using image gradients and optical flow to predict the change of the pixel intensity from the time t_{im} to the time $t_{im} + \Delta\tau$. We will use squared patches P centered on the feature location and calculate integrated and predicted patches. Patches are used to exploit the information about the neighborhood of the feature to make tracking more robust. The dependency between the size of the patch, accuracy and performance is done in original paper [8].

Using this result a feature detected on the regular image using the Harris corner detector can be tracked between the two consecutive frames taken at the times t_{im1} and t_{im2}. Assuming the small motion of the feature for the time $\Delta\tau$, we need to find the transform of patch of image gradients taken at initial time $\nabla I_w = W(\nabla I(t_{im1}))$ to the integrated image at the time $t_{im1} + \Delta\tau$ that represents the transform of the feature location. This warp can be parameterized as the rigid body transform:

$$W(\vec{x}, p) = R(p)\vec{x} + t(p), \tag{5}$$

where $(R, t) \in SE(2)$. This warp parameters p from $se(2)$ as well as the optical flow v can be found solving the following problem:

$$\min_{p,v} \left\| \frac{\Delta \hat{I}(\vec{x}, p, v)}{\left\| \Delta \hat{I}(\vec{x}, p, v) \right\|_{L^2(P)}} - \frac{\Delta I(\vec{x})}{\|\Delta I(\vec{x})\|_{L^2(P)}} \right\|_{L^2(P)}, \tag{6}$$

where $\|f(\vec{x})\|_{L^2(P)} = \int_P f^2(\vec{x}) d\vec{x}$.

The warp is the transform of the feature location from time t_{im1} to time $t_{im1} + \Delta\tau$. The optimization process is run every time for each patch independently once the number of new events fallen into this patch is above some threshold, which means that the feature movement since the last optimization is high enough. Minimization is performed over warp parameters using the local parameterization of Lie-Group SE(2) and the optical flow direction. The optical flow and the translation part of the warp are initialized using the KLT tracker, the rotational part of the warp is set to the identity.

Feature is considered to be lost and is not tracked anymore if one of the following cases is true:

1. the running mean of the residuals in the optimization exceeds some threshold. It helps to avoid cases, where the optimization is not converging to the global minimum.
2. the patch around the feature reaches the border of the image
3. events are not falling into the patch for some period of time t_{lost}, where $t_{lost} = \frac{1}{\|v(\vec{x})\|}$, because t_{lost} is chosen such that $t_{lost} \|v(\vec{x})\| = 1$ pixel of displacement.

2.2 Tracker-based Monocular Visual Odometry

Tracking the detected features as described above we get the different observations of the features in the different frames. We keep the number of the tracked features for each frame constant (100 features) to make the visual odometry work as expected.

A keyframe is added at the timestamps of the frames of the regular images. Since we are not aiming for real-time performance, we are trying to add every keyframe. Once the framework is initialized, the decision on adding the keyframe is done based on the following heuristics:

1. The number of the feature inliers after the RANSAC algorithm in the absolute frame formulation is above some threshold;
2. The number of shared features with the set of the active keyframes is below some threshold.

At the very beginning of the work, the framework needs to be bootstrapped. We use the RANSAC version of the eight-point algorithm [16] and the essential constraint [17] to initialize the first two frames and also those frames that have failed in the first criterion of the adding keyframe criteria above. If the number of active keyframes exceeds the threshold (as we keep the number of the active keyframes constant) we simply delete the oldest active keyframe. We track the observations of the features from the different keyframes. Once the number of the observations of the particular feature is equal to two we triangulate the feature location.

To refine the camera and feature poses, we solve this visual odometry problem as the bundle adjustment problem and minimize the reprojection error between the frames where the features are observed. Feature participates in the optimization if it is observable at least from the two keyframes. We keep the 2 oldest keyframes of the active frames constant for the optimization to prevent the scale gauge.

2.3 Motion Compensation

The implemented feature tracker still relies on the regular RGB or grayscale images to detect features. Motion blur or overexposure scenes may cause errors in the corner detection. Consequently, we want to build an approach that exploits the information from the event camera only without the usage of the standard images. According to the previous work in this direction [10] the motion correction of events is necessary for the reliable detection of features on the compensated event images.

Def 1. *Compensated Event Image* is the image of the intensities of summed up events according to some motion model θ, i.e.

$$I(x;\theta) = \sum_{k=1}^{N_e} b_k \delta(x - x'_k(\theta)) \tag{7}$$

where $b_k = p_k$ if polarity is used or $b_k = 1$ otherwise, and x'_k is a new motion compensated position of the event $e_k = (x_k, t_k, p_k)$, according to the motion model.

$$x'_k(t_{ref};\theta) = W(x_k, t_k, t_{ref};\theta) \tag{8}$$

In our work we model translation motion in each spatial patch $p \in \{1, \dots, N_{patches}\}$, i.e. we move the event along the linear velocity θ_p:

$$x'_k = x_k + (t_{ref} - t_k) \cdot \theta_p \tag{9}$$

where $x_k \in X_p$.

In practice, the Dirac delta δ is replaced by a smooth approximation, such as a Gaussian.

2.3.1 Tracking-based

In the Motion-Compensation problem the unknown is the motion parameters θ. This motion velocity can be directly extracted from the previously introduced feature tracker as

$$\theta_j = \frac{x_j^{new} - x_j^{old}}{t_j^{new} - t_j^{old}} \tag{10}$$

for each successfully tracked feature x_j.

This velocity is available only in the locations of the features and not available in all the other pixels. We interpolate the motion field θ_j from the detected corners to the whole image plane to perform motion compensation. We compare different strategies for the interpolation: filling with the average motion, filling with the nearest motion, and the L1-, L2- Total Variance minimization. According to the visual results and the computational efficiency comparison, the average-fill interpolation method performs sufficiently more accurate and efficient.

The main drawbacks of this approach are the following. Firstly, we still need the regular images to initialize the tracker to perform motion compensation, and secondly, the accuracy of motion compensation depends on the accuracy of the feature tracker. It means that we need regular images to do tracking in order to compensate events with the ultimate idea not to use regular images. To overcome this issue we introduce an independent motion compensation stage based on the Contrast Maximization framework.

2.3.2 Contrast Maximization

According to [13] the set of events can be motion corrected in an unsupervised manner. The intuition of the approach is that the images with corrected event positions have larger contrast comparing to the naively summed up (blurred) event images. In this work we implement the Contrast Maximization framework by maximizing the patch-wise variance σ_p^2, $p \in P$, and minimizing the inter-patch difference of the motion field - Total Variance (TV). As a result, the maximization problem is the following:

$$\sum_{p \in P} \left[\sum_{x_k \in p} \frac{1}{N_p} (I(x'_k(\theta_p)) - \mu_p)^2 - \lambda \sum_{p' \in \mathcal{N}_p} \left\| \theta_p - \theta_{p'} \right\|^2 \right] \xrightarrow{max} \vec{\theta}. \tag{11}$$

2.3.3 Edge Maximization

Instead of maximizing variance (or other loss functions considered in [14]), we propose our own geometrically intuitive loss function – Edge function. It is based on the properties of the eigenvalues of the structure tensor.

Def 2. *The structure tensor (ST)* is a matrix derived from the gradient of a function. It summarizes the predominant directions of the gradient in a specified neighborhood w of a point u, and the degree to which those directions are coherent. The structure tensor can be expressed as

$$S_w(u) = \begin{pmatrix} \iint w(v) I_x^2(u-v) dv & \iint w(v) I_x(u-v) I_y(u-v) dv \\ \iint w(v) I_y(u-v) I_x(u-v) dv & \iint w(v) I_y^2(u-v) dv \end{pmatrix}. \tag{12}$$

We exploit the following geometrical interpretation of eigenvalues of ST [15]: the first eigenvalue $\lambda_1(S)$ corresponds to the magnitude of the gradient in the dominant direction.

If we maximize it, we encourage the sharpness of the both corners and edges on the compensated image. So the edge maximization is the following (see Figure 5):

$$\lambda_1(S(\vec{\theta})) \xrightarrow{max} \vec{\theta}. \tag{13}$$

To maximize a local dominant direction within a neighborhood we perform a non-maximum suppression of the computed first eigenvalues.

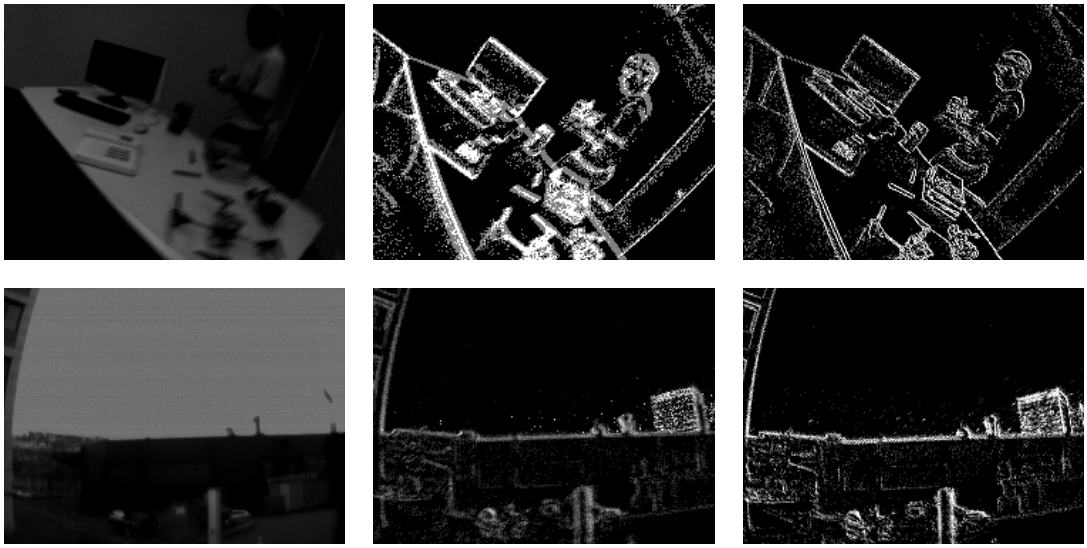


Figure 5: The results of the Motion Compensation of events via Edge Maximization. Top row: an image from the indoor sequence with high camera motion and low light, bottom row: an image from the outdoor sequence with an overexposed scene. Columns left to right: the grayscale image from a regular optical camera, the integrated event image without motion compensation, the compensated event image.

2.4 Fully Event-Inspired Visual Odometry

The combination of all the above-presented stages into one gives a Fully Event-Inspired Visual Odometry. The high-level overview of the pipeline is given in Figure 1. Starting with a set of Events, we group them into the spatial patches and collect in the time window for the further patch-wise motion compensation. Then given the compensated event images we detect features on them with the help of the Harris detector and perform tracking of the detected features. Monocular VO based on the tracks stays the same.

The disadvantage of the current implementation is that the gradients of the compensated event images are not the same as the gradients of the image from regular camera. An event image has "doubled" gradients, i.e. from the both sides of edges. So we use $\hat{I}_x = max(0, I_x)$ and $\hat{I}_y = max(0, I_y)$ in order to filter out the differences from one of the sides. We use directly the event image instead of the fused gradients (dot product of optical flow and gradients). However, there is still a need for a precise analysis of the tracker on compensated event images.

3 Implementation details

For the development, c++17 is used. We use Sophus library [18] to work with Lie-Group and Lie-Algebra, Ceres-solver library [19] for the optimization, OpenCV [20] for corner detection, optical flow estimation, and some other picture related needs. OpenGV [21] is used in the VO framework for the multiple view geometry algorithms. Pangolin library [22] is used for UI and visualization. Several tools to load, write, replay, and visualize event data are developed.

4 Experiments

Experiments are carried out on a DAVIS-240C dataset [23]. The dataset consists of several sequences of different scenes. Each scene is represented as a sequence of grayscale 240×180 images, a sequence of events with polarity, and a ground truth recorded with the help of the motion capture system. The scenes are of the different complexity: from the easy ones with checkerboards and black-white figures to the complex outdoor and dynamic ones. Also the dataset sequences differ in the camera trajectory and speed characteristics. One part of the dataset is recorded with the DAVIS 240C camera, while the other part is simulated in the event camera simulator (providing the depth map, which is used to generate the ground truth for a feature tracker).

The tracker and motion compensation are not capable to run in real-time and achieving real-time performance is not the part of this work.

4.1 Tracker experiments

The experiment is designed similarly to [8]. The ground truth for the tracks is obtained using either the KLT tracker (this one can be used as the ground truth in scenes with slow motion and good light conditions) or reprojection of the detected features on the first frame (for the simulated sequences). The life-time of the feature is an important characteristic of the tracker. Thus, the experiment is the following: we detect features on the first image, use the second frame to initialize optical flow and warp, and then track features until all of them are lost, so no new tracks are initialized during the experiment except for those in the beginning. Parameters are chosen according to the parameters study in [8].

The main characteristics to compare with other trackers are the life-time of the features in seconds, the number of tracked features for each frame and the mean distance error between tracks and ground truth in pixels averaged among all the features for each frame. For the evaluation of the results the framework by Gehrig et al. [24] is used.

The results of the comparison of the tracker implemented by ourselves with the tracker from [8] are shown in the Figure 6 and in the Tables 1 and 2. The mean error of our tracker is higher than the error from the tracker from [8], but the feature age is higher as well. This means that hyper-parameters differ and the parameters for Gehrig et al. implementation are stricter than ours. However, we are not capable to reach the accuracy of the Gehrig et al. implementation via the hyper-parameter tuning. So the differences of the obtained results are to be studied more precisely. At the same time, our implementation of the tracker is pixel-level accurate, and it outperforms the previous trackers from [7] and [25].

	Our method	Gehrig, Daniel, et al. [8]	Kueng [7]	Zhu [25]
shapes_6dof	1.67	**0.65**	1.75	3.04
poster_6dof	1.69	**0.64**	2.86	2.99
simulation_3walls	1.17	**0.66**	-	-

Table 1: Mean distance trajectory error in pixels

	Our method	Gehrig, Daniel, et al. [8]	Kueng [7]	Zhu [25]
shapes_6dof	**4.44**	4.07	1.53	1.30
poster_6dof	**5.56**	2.9	0.65	2.56
simulation_3walls	**0.5**	0.27	-	-

Table 2: Feature age in seconds

4.2 Visual odometry experiments

We compare our implementation of visual odometry (using both regular images and events) with the visual odometry implemented by Kueng et al. [7]. Since the exact names of the sequences are not given in the paper [7], we carried out this experiment on the two sequences of the DAVIS 240C datasets with approximately the same scene depth as given in the paper [7]. Using the sequences with approximately the same scene depth we can compare the relative to the scene depth localization error. The comparison of relative errors and plots of absolute errors and the obtained trajectories can be found on Table 3 and Figure 7.

As you can see, the relative error for our implementation is lower than the error from Kueng et al. It can happen due to several reasons: (i) such a method of comparison is not fair enough, and the trajectories of the camera for the sequences in work by Kueng et. al might be much more complex than in the sequences we use for evaluation, (ii) the work by Kueng et al. is one of the earliest works on 6-DoF event-based VO and the tracker we used for our implementation performs better than the tracker provided in their paper (see Section 4.1).

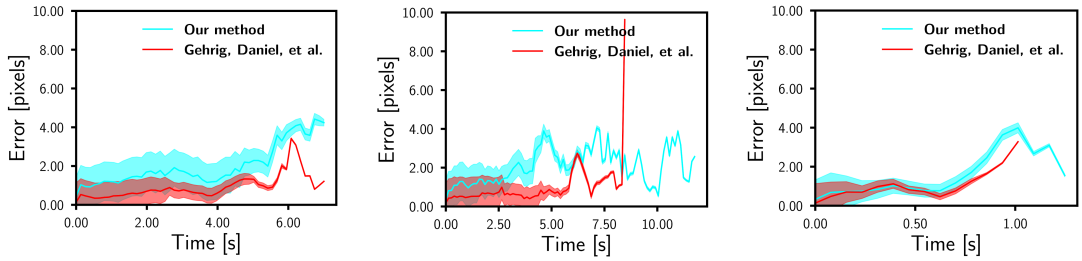
	1st sequence	2nd sequence
Gehrig, Daniel, et al. [8]	4%	7.5%
Our method	0.36%	3.5%

Table 3: Mean trajectory error relative to the maximum scene depth

4.3 Event-only Visual Odometry experiments

We compare the two versions of our implementation of Visual Odometry, the first is based on both events and regular images, and the second is based solely on events.

The results we have obtained on the events-only VO are worse than the ones using both events and images, but the obtained trajectories are reasonable (see Figure 8). It shows that the approach can be used along with the standard approach for the constrained capturing conditions (speed and lightness). However, to obtain more robust results the feature tracking stage should be reworked and adapted for the compensated event images.



(a) shapes_6dof (b) poster_6dof (c) simulation_3walls

Figure 6: Comparison of the event-based trackers. Plots are obtained for our implementation and [8]. The width of the line represents the number of the tracked features at the moment: the wider line - the better.

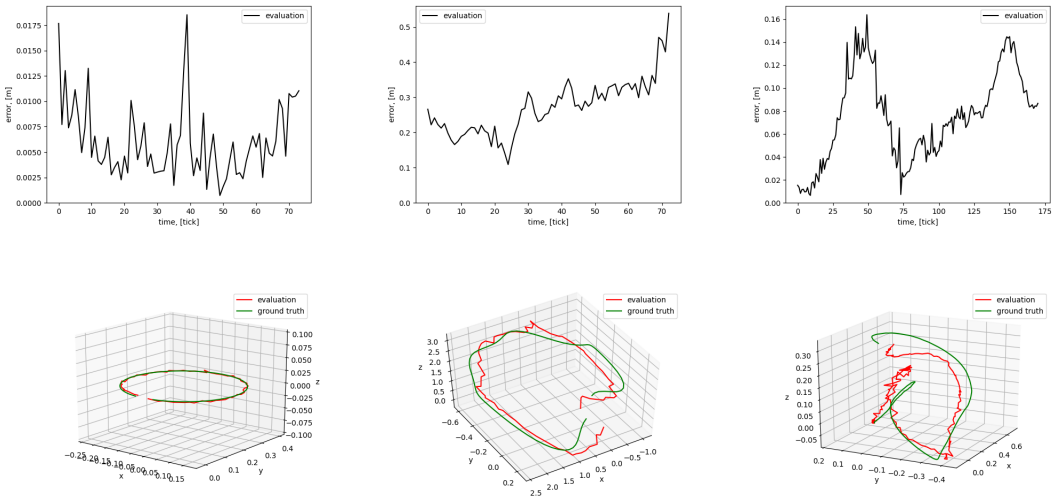


Figure 7: Plots of Visual Odometry (based on both images and events) evaluation. Top row: absolute error distance to the ground truth in meters in time. Bottom row: Trajectories of the camera. Red is the evaluation, green is the ground truth. Columns left to right: simulation_3planes, simulation_3walls and boxes_6dof.

5 Discussion

5.1 Performance

We have seen that the data produced by the event cameras along with the regular images can be used to provide the robotics or AR/VR systems with the localization. Unfortunately, the performance of the algorithms using events is not real-time yet because of the enormous amount of events to be processed per second. If, for example, we consider the tracking of 100 features during 50ms, approximately 50000 events will come into the system, that is around 500 events per patch. It results in (depending on the hyperparameters and current optical flow value) from 1 to 10 optimizations per feature to track it during this time, where each optimization takes 1-3ms till the convergence. Thus, for a single-thread system overall time of tracking of all 100 features during 50ms is between 100ms and 3000ms. But thanks to the asynchronous nature of the tracker and events in general,

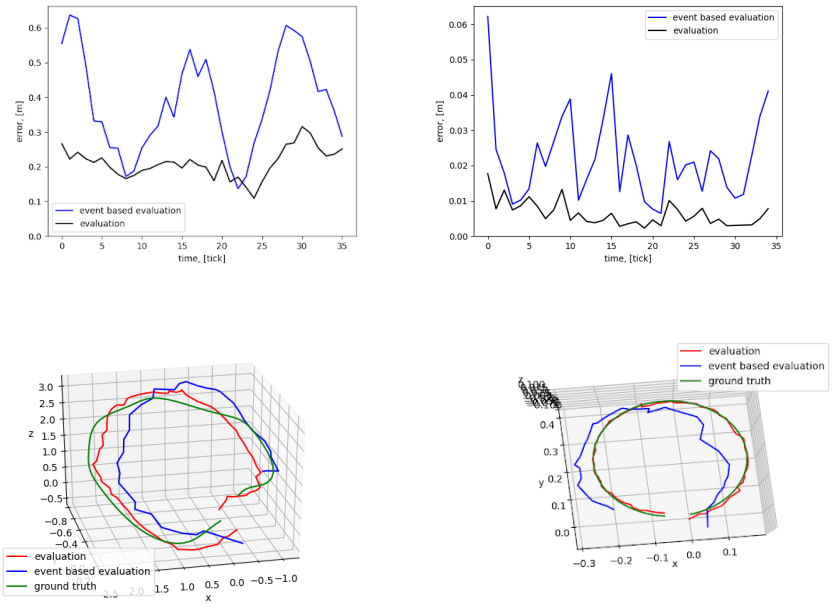


Figure 8: Comparison of the Visual Odometry evaluation for VO using both events and images (*evaluation*), and VO using only events (*event based evaluation*). Top row: the absolute error distance to the ground truth in meters in time. Bottom row: Trajectories of camera. Columns left to right: simulation_3walls, simulation_3planes.

some distributed and parallelized techniques on software as well as on hardware level can be done to decrease the system latency.

5.2 Other experiments

We have carried out some other experiments during our work. For example, we have tried to include rotation into the patch-wise motion model for the motion compensation of the events (it is only translation in the current implementation). But it does not work out due to the overparameterized model, and possibly there is no enough information in the event images for the convergence.

We have also tried to model the noise in pixels of the event camera. We associate Bernoulli random variables ξ_i to each pixel $i \in I$ of the event camera, such that $\xi_i = 1$ if the pixel i tracks correct brightness change, and $\xi_i = 0$ if the pixel i of event camera is triggered by a noise event. Such modeling hypothesizes that some pixels might be broken and produce noise events with high probability. So we predict the probability of pixel to produce a correct event p_i adding these probabilities as optimization parameters into the motion compensation stage. During the integration of the corrected events we multiply the event value b_i with its associated correctness-probability $p_i \in [0, 1]$. The optimized functional is the Edge function. In the end, it turns out to be an over-complicated problem, and the optimization converges to non-realistically shrinked images. However, the same idea can be tested using post-processing techniques in the future work.

6 Conclusion and Future work

To sum up, we have implemented the feature tracker based on images and events, implemented the visual odometry based on this tracker, implemented and proposed one method for motion compensation of events to obtain sharp event images, and to use them for visual odometry based on events only. We have carried out an experimental comparison of results obtained for tracker and visual odometry with the other respective methods.

The results of the fully-event based visual odometry are not reliable yet because the used feature tracker is not carefully adapted for the compensated event images. Thus, the research in this direction should be continued. And the question about making the whole pipeline performance more efficient should be addressed in the future work.

References

[1] Nathan Piasco, Désiré Sidibé, Cédric Demonceaux, and Valérie Gouet-Brunet. A survey on visual-based localization: On the benefit of heterogeneous data. *Pattern Recognition*, 74:90–109, 2018.

[2] Christian Brandli, R. Berner, Minhao Yang, Shih-Chii Liu, and T. Delbruck. A 240 180 130 db 3 s latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49:2333–2341, 2014.

[3] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision*, pages 349–364. Springer, 2016.

[4] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[6] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. 1991.

[7] Beat Kueng, Elias Mueggler, Guillermo Gallego, and Davide 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.

[8] Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. Asynchronous, photometric feature tracking using events and frames. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 750–765, 2018.

[9] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2016.

[10] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. 2017.

[11] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based visual inertial odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5391–5399, 2017.

[12] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide 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.

[13] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. *CoRR*, abs/1804.01306, 2018.

[14] Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza. Focus is all you need: Loss functions for event-based vision. *CoRR*, abs/1904.07235, 2019.

[15] Thomas Brox, Rein van den Boomgaard, François Lauze, Joost van de Weijer, Joachim Weickert, Pavel Mrázek, and Pierre Kornprobst. Adaptive structure tensors and their applications. In *Mathematics and Visualization*, pages 17–47. Springer Berlin Heidelberg, 2006.

[16] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[17] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004.

[18] *Sophus library.* `https://github.com/strasdat/Sophus`.

[19] *Ceres-solver library.* `http://ceres-solver.org/`.

[20] *OpenCV library.* `https://opencv.org/`.

[21] *OpenGV library.* `https://laurentkneip.github.io/opengv/`.

[22] *Pangolin library.* `https://github.com/stevenlovegrove/Pangolin`.

[23] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide 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.

[24] Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. Eklt: Asynchronous photometric feature tracking using events and frames. 2019.

[25] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based feature tracking with probabilistic data association. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4465–4470. IEEE, 2017.