Leveraging an Active Prism to Enhance Feature Detection in Event Cameras

Botao He [3], Ze Wang [1,2], Yuan Zhou [1,2], Jingxi Chen [3], Chahat Deep Singh [3], Cornelia Fermüller [3], Yiannis Aloimonos [3], Chao Xu [1,2], and Fei Gao[1,2]

I. Introduction

Event cameras are bio-inspired vision sensors that were designed to capture motion information. Instead of recording image intensity, they record intensity change information only. Generally, assuming no significant changes in the scene illumination, the image intensity changes are due to the 3D motion of either the scene objects or the camera. For this reason, event cameras recently have been used in classic applications of visual motion processing, such as SLAM [1] and Structure from Motion [2], motion segmentation [3], [4], gesture recognition [5], human pose estimation [6], as well as niche areas such as space [7] or microscopy [8].

Although capturing motion information (assuming constant illumination) is beneficial for many application scenarios, it comes with a cost. Specifically, the events generated by event cameras are dependent on the motion (either camera ego-motion or object motion). This dependency on the motion will make the features in the generated events keep changing as the motion of the objects or the camera is changing. We call this effect "unstable event features". As an example, if a square-like object is translating horizontally while observed with a static event camera, what we will see is that its top and bottom horizontal edges will not generate any events during this horizontal translation. Suppose now that the square changes its translational motion from the horizontal to the vertical direction. In this case, the events of horizontal edges will show up but the previously visible events for the vertical edges will disappear. This general phenomenon of "unstable event features" will create problems for applications like feature tracking. Lines which are in the direction of motion will not create events.

In the past decade, there have been many works trying to eliminate this problem through algorithms or post-processing approaches. They either associate events with previously maintained data [6], [9] or combine the event camera with a standard camera [10], [11]. The former techniques usually employ 2D/3D event maps or reconstructed intensity images to maintain more information, and minimize the reprojection error to optimize correspondence between events. However, these methods also suffer from noise, which is common when the event camera moves slowly or remains static. In the latter case, although the texture is much more stable by introducing standard cameras, it also leads to robustness issues due to the poor performance of standard cameras in high dynamic range, and dark scenarios and when there is motion blur.

All of the above methods try to solve the problem of unstable event features on the software side. The problem however is fundamentally introduced by sensor characteristics. Therefore, these works do not solve the problem at its root: indeed, there is no theory on how to complete or recover the missing information. To fundamentally address this problem, we need to physically introduce an additional motion to the event camera. This can only be done at the hardware level, and there are some previous works taking this perspective [12], [13] by physically adding random or unobserved motion (like vibration) to the event camera. Such approaches can mitigate the dependency of event generation on motion but they also introduce problems for the quality of the generated events because the unobserved motion cannot be compensated, which causes severe motion blur.

This report identifies and resolves these fundamental challenges by physically adding an active micro-motion, which is controllable, observable, and omni-directional. An overview of the proposed system is shown in Fig. 1. Specifically, our proposed approach is a hardware design combined with a software solution. Inspired by rotating wedge prism mechanism [14], the hardware is a rotating wedge prism in front of an event camera, with the rotation controlled by a servo actuator. In this design, we can track the rotating motion of the motor and analyze the refraction of the rotating wedge prism such that our introduced motion now becomes observable, details in II-A.

In the software solution, we first control the actuator to drive the rotation of the wedge prism. Then, we correspond the event stream with position feedback data by synchronizing the servo actuator with the timestamps of the event camera. After that, we perform motion compensation by warping the events with the rotational motion and restoring the quality of our event data. More details can be found in II-B and II-C.

The output of our system is a new **event stream**, where the motion trajectory induced by the wedge prism has been compensated for. The new event stream is useful for visual recognition tasks and for feature tracking.

The main components in this work can be summarized as:

† This work is finished during the first author's internship at Huzhou Institute of Zhejiang University

1 State Key Laboratory of Industrial Control Technology, Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, 310027, China. Email:{wangze0527, yzhou, cxu, fgaoaa}@zju.edu.cn.

2 Huzhou Institute of Zhejiang University, Huzhou, 313000, China.

3 Perception and Robotics Group, University of Maryland Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA. Email:{botao, ianchen, chahat, fer, jyaloimo}@umd.edu

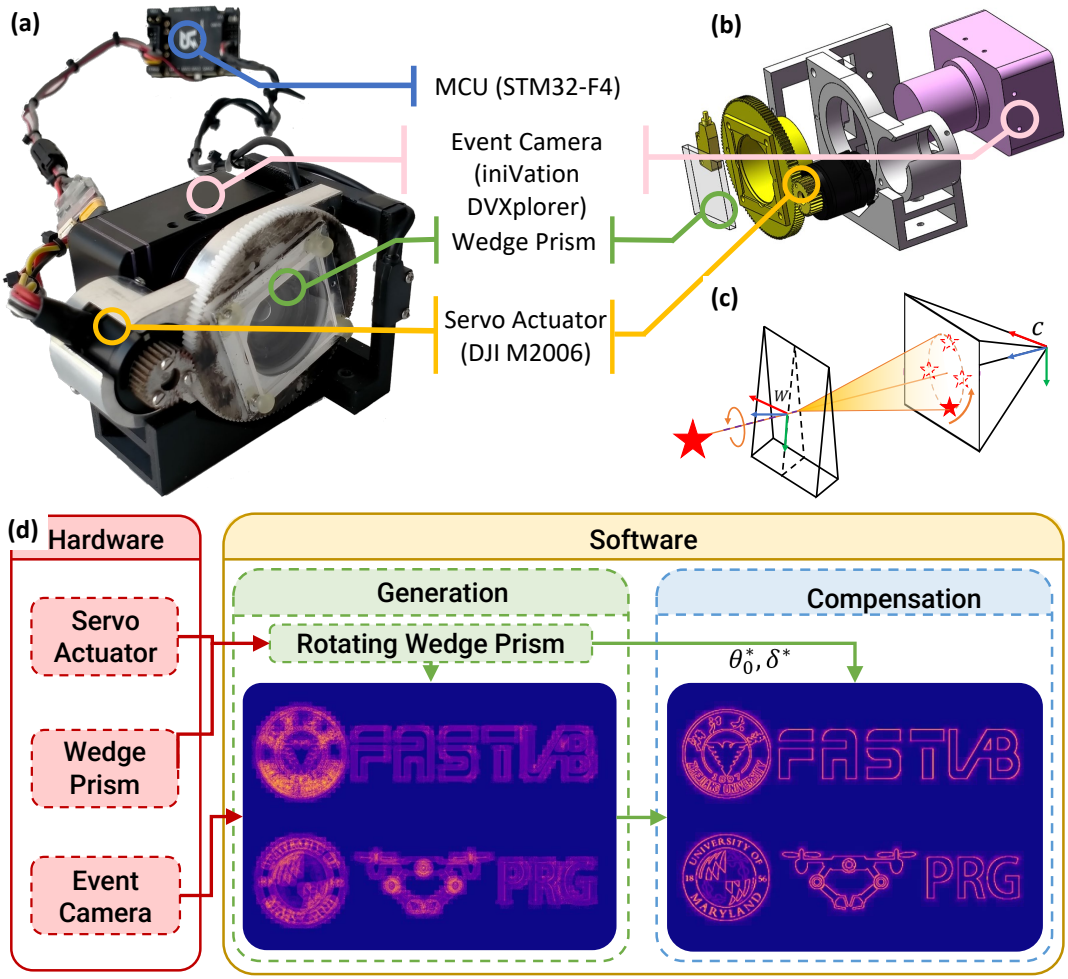

Fig. 1. (a): Hardware components of our texture-enhanced event camera. (b): Hardware Design. (c): Graphical demonstration of the micro-motion generating process. (d): System overview.

- A novel event-based hardware solution utilizing a rotating wedge prism to account for the fundamental motion dependency problem in event-based vision.
- A motion generation method that can generate controllable, observable, and omni-directional motion, which can actively maintain all environmental boundary information.
- A motion compensation method that can compensate the actively introduced motion and output a texture-enhanced event stream that is compatible with existing event-based algorithms.

II. System Design

In this section, we present the design of our texture-enhanced event camera. The system description is divided into two parts: the hardware design, and the software solution, as shown in Fig. 1(d). For the hardware design, we demonstrate the mechanical structure of the proposed system. In the software part, we introduce the rotation generation process and then demonstrate the calibration and compensation procedure with some experimental results.

A. Hardware Design

The hardware design, illustrated in Fig. 1(a) and Fig. 1(b), consists of three parts: the optical deflector module, the actuator module, and the camera module with the micro computing unit (MCU). The optical deflector module, marked as green in Fig. 1(a) and 1(b), is a wedge prism that deflects the incoming light to a fixed angle along x_w, where x_w, y_w, z_w denote the corresponding axis of the wedge-prism frame $W \in \mathbb{S}^2$ (shown in Fig. 1(c)). The actuator module, marked as yellow, is composed of a servo actuator that can provide absolute position feedback and a transmission mechanism. It is used to drive the optical deflector module to rotate along the z_c, where x_c, y_c, z_c denote the corresponding axis of the camera frame $C \in \mathbb{S}^2$ (shown in Fig. 1(c)). The camera module, marked as pink, is a standard event camera that supports time synchronization with external sensors. The MCU, marked as blue, accounts for controlling the rotating speed, receiving position feedback and synchronizing timestamps between the event camera and the actuator's encoder.

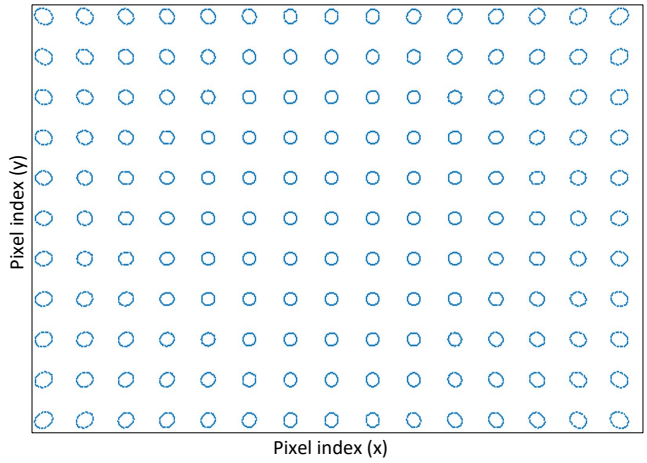

Fig. 2. The trajectories in the image plane, which are induced by the changing incoming light.

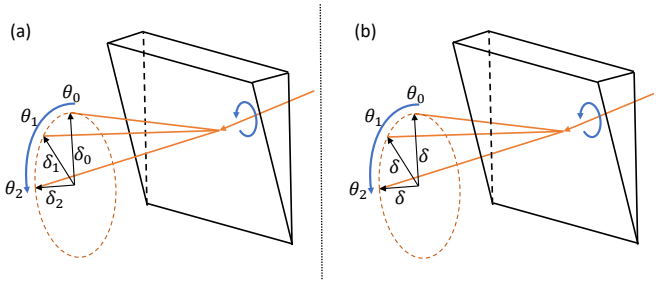

Fig. 3. Illustration of the fitting process in the calibration procedure. (a): The trajectory induced by deflecting an incoming light ray over time looks approximately like a circle in $\mathbb{S}^2$ with a slightly varying distance δ_i between the original and deflected ray. (b): A circle in $\mathbb{S}^2$ is fit to the trajectory, where each point has the same distance δ from the original direction.

B. Micro-motion Generation

In the proposed system to generate additional events on contours parallel to the motion of the camera and on the edges of the static background, we utilize the working principle of the wedge-prism deflector[15]. It actively adjusts the direction of the incoming light, as illustrated in Fig. 1(c). At the beginning of the procedure, the wedge prism has a certain orientation and deflects the incoming light at fixed angle δ_0 as described in Section. II-A. Then during operation, the actuator module drives the optical deflector module to rotate along the z_c to make the incoming light constantly change its deflection. This way, the incoming light continually generates events because it creates a motion on the image plane with a circle-like trajectory, as shown in Fig. 1(c). As a result, there appears to be continuously changing 2d rotational motion induced in the image plane. The induced motion patterns are circle-like trajectories. Because the image motion is in all directions in the image plane, the output event stream contains all the information of boundaries in the scene, as shown in Fig. 4 and Software-Generation part in Fig. 1(d).

C. Micro-motion Calibration and Compensation

By rotating the wedge prism, a saccade-like motion is generated on the image plane, resulting in object boundaries

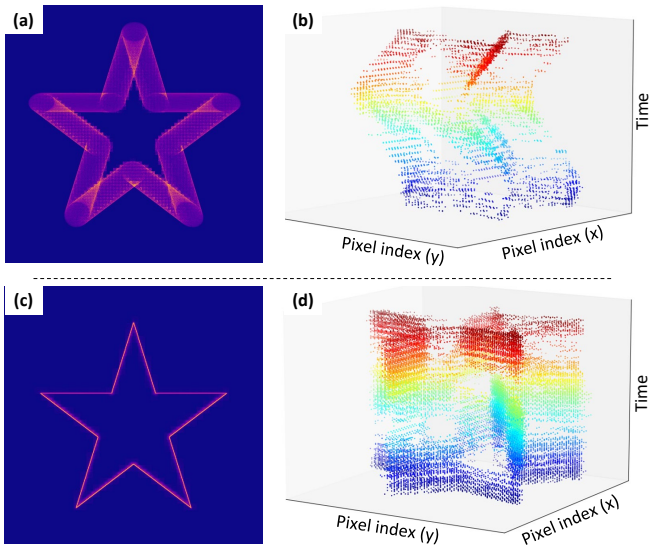

Fig. 4. Illustration of the micro-motion compensation. (a) and (b): 2-D edge map and 3-D event stream before compensation. (c) and (d): 2-D edge map and 3-D event stream after compensation.

stably maintained in the event stream. However, in the event slices (i.e. the images created by binning the events over a small time interval), the boundaries are blurry due to the self-motion of the wedge prism. To get sharp edges, the events triggered by the same incoming light ray direction should be moved to the same pixel. To achieve this, we first need to calibrate that is find the rotation of the wedge at the beginning of the recording, and then compensate for the spatial displacement of the events due to the wedge motion.

Fig.2 shows that the trajectory of each incoming light is like an ellipse on the image plane. By projecting each event to the camera frame $C \in \mathbb{S}^2$ using camera intrinsic matrix K, the ellipse-like trajectory on the image plane can be transformed to circle-like trajectory in C, as shown in Fig. 3(a). As shown with examples in **Appendix. A** the error due to this approximation is not significant.

In a calibration procedure, we estimate the two parameters: δ, the radius of the circle, and θ_0, the offset between the angle of the wedge prism at the start and a fiducial position of the servo actuator. We estimate these angles in an iterative procedure by aligning events collected over two seconds using a sharpness measure [16]. As the initial value, we use the refraction angle of parallel light by the corresponding wedge-prism for δ, and the zero-position of the servo actuator for θ_0.

In the current implementation, we use a simple measure for alignment: we minimize the number of pixels, which have a value above a threshold. In more detail, we transfer the events from the spatial-temporal domain (x, y, t) to the (x, y, θ) domain by synchronizing the events' timestamp with the wedge prism's angular position. Then, we warp all events back to θ_0 to compensate for the rotation. The warping function is described as $\Pi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, which warps each event's position on the image plane as $\Pi(x, y, \theta) : (x, y, \theta + \theta_0) \rightarrow (x', y', \theta_0)$. The warping function can be

represented as:

$$e'_i = \Pi\{(x, y, \theta + \theta_0)\} = (x', y', \theta_0). \quad (1)$$

From the warped events $E' = e'_i (i = 1, 2...)$, we construct the event-count image γ, with each pixel encoding the number of events mapped to it in E':

$$\gamma_{i,j} = \sum_{e_i \in E'} \begin{cases} 1, (x, y) = (i, j); \\ 0, else. \end{cases} . \quad (2)$$

Now, we normalize γ to (0, 1) and construct a new normalized event-count image Γ:

$$\Gamma = Normalize(\gamma) \in (0, 1). \quad (3)$$

Then, the cost J is represented by thresholding Γ:

$$J = \sum_{(i,j) \in \Gamma} \begin{cases} 1, \Gamma_{i,j} \geq \tau_{threshold}, \\ 0, else. \end{cases} , \quad (4)$$

with $\tau_{threshold}$ being a specified threshold. The optimization problem can be expressed as:

$$\min_{\delta, \theta_0} J \quad (5)$$

Because the hardware setup is fixed and known, the optimal value δ^* and θ_0^* will not differ much from the initial guess δ and θ_0, which means it does not take much computational cost to find the optimal solution pair by brute force search in specified solution space.

After calibrating the rotation parameters, we can compensate for the microsaccade using the same warping function Π in real-time. The experimental results are shown in Fig. 4. As a result, the compensation algorithm outputs a new compensated **event stream** for further processing, like feature detection.

III. Experiment and Evaluation

A. Evaluation for Feature Detection Application

This experiment is designed to demonstrate the superiority of the proposed system in the feature detection task, which is one of the most representative tasks in low-level vision, and also a basic building block for various robotics applications.

In the experiment, we use the eFAST[17] method to detect corner features directly in the event stream. The refraction angle of the wedge prism is set to 0.5 degrees, because when the refraction angle becomes larger, the number of events per one rotation period also becomes larger, and the difficulty for compensation also increases since the error is also amplified. Choosing 0.5 degrees is a good balance between event density and compensation performance. The rotation speed is set to 12.5 hz, which is also a good balance between system performance and accuracy, because a higher speed generates more environmental information, while introducing more compensation errors due to time synchronization.

The result is illustrated in Fig. 5, in which the camera is moving along its X-axis. Fig. 5(a) shows that from the standard event camera output, we cannot detect the corner features robustly. The reason is that there is not sufficient information for corner detection as there is barely any motion along the Y-axis, and thus hardly any events are created on the horizontal edges. Fig. 5(b) shows the out of the proposed system, which in this case can detect all corner features accurately and robustly because of the introduced micro-motions and the event compensation.

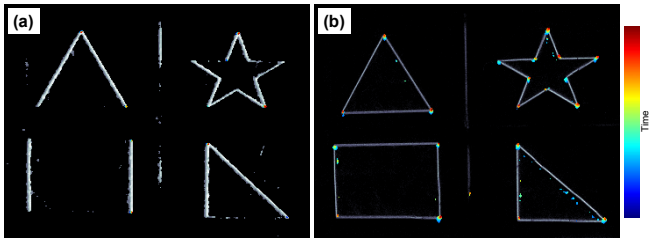

Fig. 5. Experimental result of feature detection application. The camera is moving along its X-axis. (a): Feature detection without the proposed texture-enhanced event camera system. (b):Feature detection with the proposed system.

References

[1] G. Gallego, J. E. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, "Event-based, 6-dof camera tracking from photometric depth maps," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 10, pp. 2402–2412, 2018.

[2] C. Ye, A. Mitrokhin, C. Fermüller, J. A. Yorke, and Y. Aloimonos, "Unsupervised learning of dense optical flow, depth and egomotion with event-based sensors," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5831–5838.

[3] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos, "Event-based moving object detection and tracking," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[4] A. Mitrokhin, C. Ye, C. Fermüller, Y. Aloimonos, and T. Delbruck, "Ev-imo: Motion segmentation dataset and learning pipeline for event cameras," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6105–6112.

[5] J. H. Lee, T. Delbruck, M. Pfeiffer, P. K. J. Park, C.-W. Shin, H. Ryu, and B. C. Kang, "Real-time gesture interface based on event-driven processing from stereo silicon retinas," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 12, pp. 2250–2263, 2014.

[6] Y. Zhou, G. Gallego, and S. Shen, "Event-based stereo visual odometry," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1433–1450, 2021.

[7] G. Cohen, S. Afshar, B. Morreale, T. Bessell, A. Wabnitz, M. Rutten, and A. van Schaik, "Event-based sensing for space situational awareness," *The Journal of the Astronautical Sciences*, vol. 66, no. 2, pp. 125–141, 2019.

[8] Z. Ni, C. Pacoret, R. Benosman, S. Ieng, and S. RÉGNIER*, "Asynchronous event-based high speed vision for microparticle tracking," *Journal of microscopy*, vol. 245, no. 3, pp. 236–244, 2012.

[9] 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*, vol. 2, no. 2, pp. 593–600, 2016.

[10] 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*, vol. 3, no. 2, pp. 994–1001, 2018.

[11] J. Hidalgo-Carrió, G. Gallego, and D. Scaramuzza, "Event-aided direct sparse odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5781–5790.

[12] A. Mishra, R. Ghosh, J. C. Principe, N. V. Thakor, and S. L. Kukreja, "A saccade based framework for real-time motion segmentation using event based vision sensors," *Frontiers in Neuroscience*, vol. 11, 2017. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2017.00083

[13] A. Yousefzadeh, G. Orchard, T. Serrano-Gotarredona, and B. Linares-Barranco, "Active perception with dynamic vision sensors. minimum

saccades with optimum recognition," *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 4, pp. 927–939, 2018.

[14] K. Tyszka, M. Dobosz, and T. Bilaszewski, "Double wedge prism based beam deflector for precise laser beam steering," *Review of Scientific Instruments*, vol. 89, no. 2, p. 025113, 2018.

[15] D. Senderakova and A. Strba, "Analysis of a wedge prism to perform small-angle beam deviation," in *Photonics, Devices, and Systems II*, vol. 5036. SPIE, 2003, pp. 148–151.

[16] 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*, 2018, pp. 3867–3876.

[17] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection," 2017.

APPENDIX

A. Fitting error analysis of micro-motion calibration

In our approximation of the light refraction for the rotation wedge-prism, we approximate trajectory with a circle of radius δ, instead of using different δ_i for different rotation angles θ_i. Fig. 6 shows the results of a simulation for a wedge prism with a refraction angle of 0.5 degrees. Simulating 640 by 480 event count image due to a 90-degree field of view, the maximum error in this approximation is 0.09 degrees, which is less than 2 pixels for the DVXplorer camera. This approximation error is small enough that we can safely ignore it for real-world robotics application scenarios.

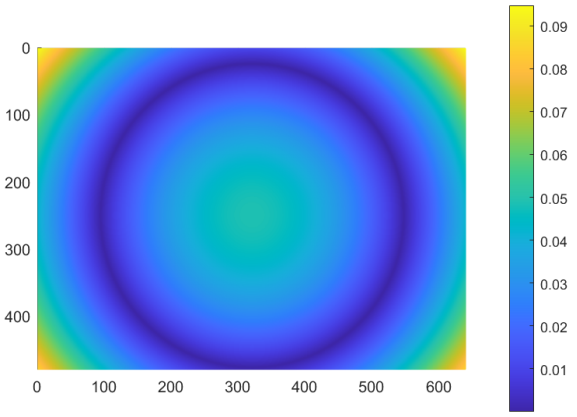

Fig. 6. Illustration of the error introduced by calibration and compensation.

B. Discussion of the effect of micro-motions on moving objects

Event cameras are designed to capture motion information and are most interesting for applications when the camera or objects are moving. For these scenarios, the motion at each pixel in the image plane is a combination of our introduced micro-motion and the 3D scene motion.

In the following Fig. 7, we qualitatively show the combined effect of the object(a ball) motion and the micro-motion. As we can see, the trajectory is circular when the translation speed is 0 (static), because the motion is pure rotational micro-motion. As the translational speed increases, the trajectory goes from a circle to a sinusoidal path and finally to an almost straight line, which means the proportion of object motion in the overall motion gradually rises to the dominant position. Because the effect of micro-motion decrease with the object moving faster, the micro-motion is more effective for scenarios involving a low or medium speed ratio between the scene motion and our introduced micro-motion. This is the case for most robotic applications because our micro-motion can be around 20hz, and this is much higher than the ego-motion of objects in most robotic applications.

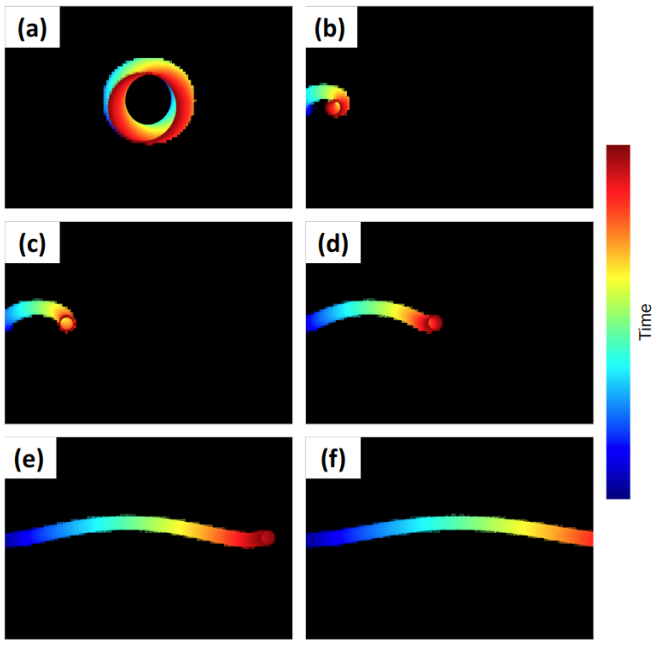

Fig. 7. Illustration of the coupling effect of introduced micro-motion and the ego-motion on the trajectory of a solid circle. The micro-motion is 1 degree per frame. (a): the ball is static. (b), (c), (d), (e): The ball is translating with speed 1, 2, 4, 8, 10 degree per frame.