A Texture-enhanced Event Camera Using Rotating Wedge Prism

Botao He [3], Ze Wang [1,2], Yuan Zhou [1,2], Jingxi Chen [3], Chahat Deep Singh [3], and Fei Gao[1,2]

I. Introduction

The event cameras, a type of bio-inspired vision sensors that can capture only the intensity change information. In the general cases where the intensity distribution of the scene is approximately constant, the intensity change in the view is then due to the motion of objects or camera, for this reason, the event camera is generally used for the applications that require to capturing the motion information. Besides being used to capture motion information, event-camera also recently draws attention to research and applications due to its unique features in asynchronous low-latency response and high dynamic range.

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

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 [1], [2] or combine the event camera with a standard camera [3], [4]. The former usually use 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. The latter one, although the texture is much more stable by introducing the standard cameras, it also leads to many robustness issues due to the poor performance of standard camera in HDR, motion blur and dark scenarios. All the above methods try to solve the problem on the software aspect, however, the problem is fundamentally introduced by sensor characteristics instead of algorithm imperfection. Therefore, these works do not solve the problem at its root, the hardness of completing or recovering the missing information is theoretically unknown. To fundamentally solve this problem, we need to physically introduce additionally motion to the event camera. This can be only down on hardware-level, and there are some previous works on this perspective [5], [6] by physically adding random or unobserved motion (like vibration) to the event camera. This type of approach can mitigate the dependency of event generation on motion but also introduce the problem in the quality of generated events because such unobserved motion cannot be compensated, which causes severe motion blur.

This report identifies and resolves these fundamental challenges by physically adding a controllable, observable and omni-directional micro-motion. An overview of the proposed system is shown in Fig. 1. Specifically, our proposed approach is a hardware design as well as a software solution based on a rotating wedge prism mechanism. The hardware design contains a rotating wedge prism in front of an event camera. The rotation is 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 wedge prism rotating. Then, we correspond the event stream with position feedback data by synchronizing timestamps between the event camera and the servo actuator. After that, we do the motion compensation to remove the rotational motion and restore the quality of our event data. More details can be found in II-B and II-C.

The main contributions in this work can be summarized as:

- 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 boundary information in the environment.
- 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 perception algorithms.

† 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.

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@umd.edu, fgaoaa@zju.edu.cn.

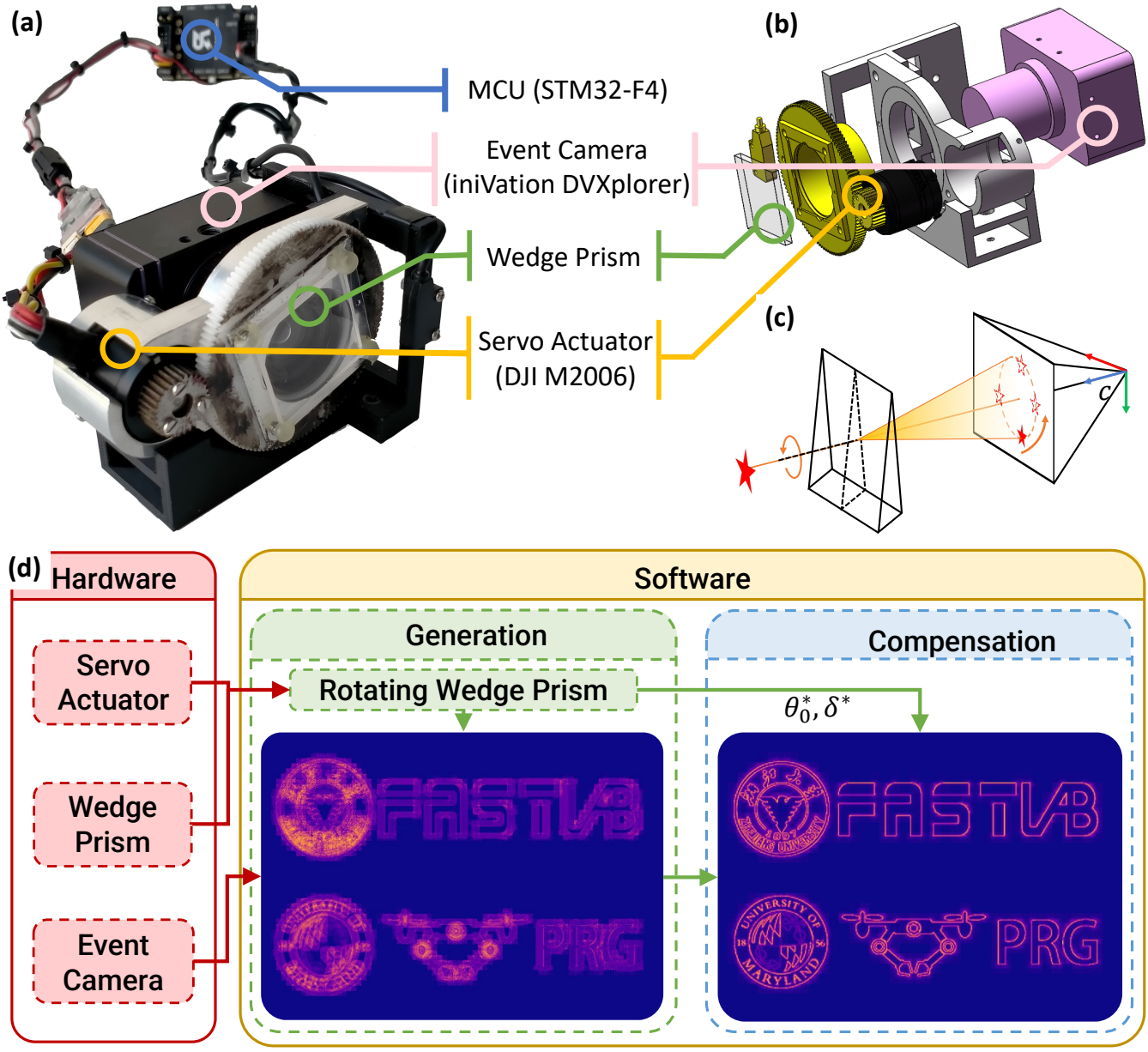

Fig. 1. (a): Hardware components of our texture-enhanced event camera. (b): Hardware Design. (c): Graphical demonstration of the micro-motion generating process. (d): System overview.

II. System Design

In this section, we present the design of our texture-enhanced event camera. The system is separated into two parts: Hardware design and software solution, as is 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 is illustrated in Fig. 1(a) and Fig. 1(b), which is consist of three main parts: optical deflector module, actuator module, camera module and micro computing unit(MCU). The optical deflector module, marked as green in Fig. 1(a) and 1(b), is a wedge prism that can deflect the incoming light to a fixed angle along x_w. 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. The camera module, marked as pink, is a standard event camera that supports time synchronization with external sensors. The MCU, marked as blue, is account for controlling the rotating speed, receiving position feedback and synchronizing timestamps between the event camera and the actuator's encoder.

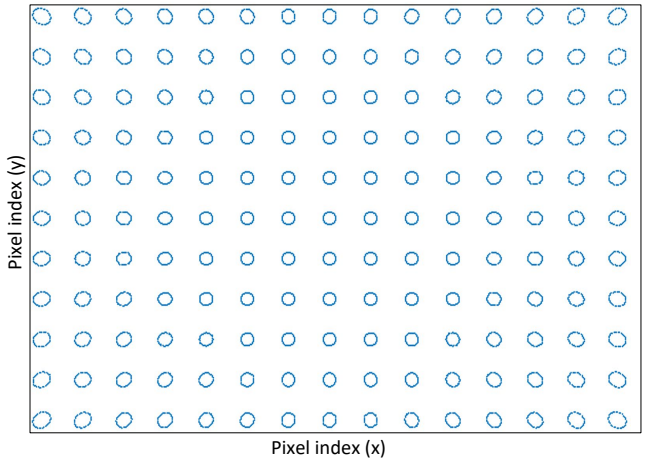

Fig. 2. The trajectory of different incoming lights on the image plane.

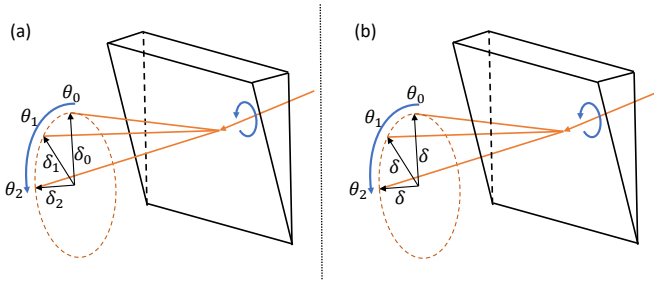

Fig. 3. Illustration of the fitting process in calibration procedure. (a): The trajectory of one incoming light, it looks like a circle in $\mathbb{S}^2$ but not a circle because each point on it has a corresponding distance δ_i from its original direction. (b): A circle in $\mathbb{S}^2$ is used to fit the trajectory, where each point on it has same distance δ from the original direction.

B. Micro-motion Generation

One fundamental property of the event camera is that it only responds to motion (assuming constant illumination), therefore, if we want it to see the whole static background, we should move the camera in various directions. In the proposed system, instead of moving the whole camera, which is energy-consuming, we utilize the working principle of the wedge-prism deflector[7] to actively adjust the direction of the incoming light, as is demonstrated in Fig. 1(c). Firstly, the wedge prism is utilized to refract the incoming light for a fixed angle as described in Section. II-A. Then, the actuator module drives the optical deflector module to rotate along the z_c to make the incoming light constantly change its motion pattern. In this way, the incoming light can continually generate events because it is in continuous motion on the image plane with a circle-like trajectory, as demonstrated in Fig. 1(c). As a result, the whole image plane looks like rotating along its x and y axis. Because the circle-like trajectory contains motion in all directions, the output event stream contains all boundary information of the environment, as shown in Fig. 4 and Software-Generation part in Fig. 1(d).

C. Micro-motion Calibration and Compensation

After generating the microsaccade, the dependency of event generation on the motion is mitigated and the boundary of objects are stable across the event stream and event frame.

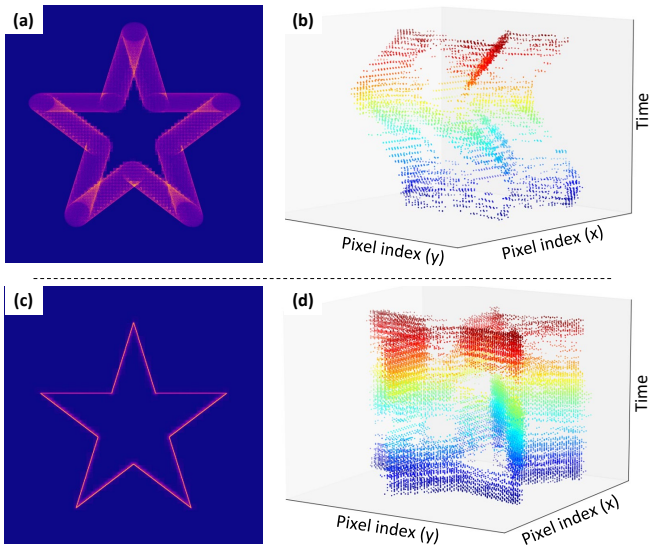

Fig. 4. Illustration of the micro-motion compensation. (a) and (b): 2-D edge map and 3-D event stream before compensation. (c) and (d): 2-D edge map and 3-D event stream after compensation.

the environmental boundaries completely appear in the event stream and event frame. However, the boundary is blurry due to the self-motion of the wedge-prism. To get sharp edges, the events triggered by the same incoming light should be compensated to the same pixel. To achieve that, we should first calibrate the rotation parameters and then compensate for the introduced motion to the events.

The Fig.2 shows that the trajectory of each incoming light is like an ellipse on the image plane, if we project them to the camera frame $C \in \mathbb{S}^2$, the trajectory looks like a circle, as shown in Fig. 3(a). Intuitively, we can use a circle in $\mathbb{S}^2$ with radius δ to fit the trajectory, as demonstrated in Fig. 3(b), and the fitting error is ignore-able as analyzed in **Appendix. A**. Based on this, the calibration procedure is designed to solve an optimization problem with two parameters: δ and θ_0, where θ_0 represents the offset between the starting angle of the wedge prism and the initial position of the servo actuator. The first step is to assign the initial values for δ and θ_0, noted as δ^0 and θ_0^0. The choice of initial values is based on the hardware setup, δ^0 is set as the refraction angle of parallel light by the corresponding wedge-prism and θ_0^0 is determined by the zero-position of the servo actuator. Secondly, we collect a batch of events $E = e_i (i = 1, 2...)$ and actuator position feedback data over a period of time t, which is set to $t = 2s$ in the experiment as we found that this can achieve a good balance between calibration accuracy and computational cost. Next, we transfer the events from the spatial-temporal domain (x, y, t) to (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 the rotation. The warping function is described as $\Pi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, which warps the event's position on image plane as $\Pi(x, y, \theta) : (x, y, \theta + \theta_0) \rightarrow (x', y', \theta_0)$. The

warping function can be represented as:

$$e_i' = \Pi\{(x, y, \theta + \theta_0)\} = K \cdot g'^{-1}(\mathbf{v_1}, \theta + \theta_0) \cdot K^{-1} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = (x', y', \theta_0). \tag{1}$$

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

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

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

$$\Gamma = Normalize(\gamma) \in (0, 1). \tag{3}$$

Then, the cost J is represented by thresholding the Γ:

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

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

$$\min_{\delta, \theta_0} J \tag{5}$$

Because the hardware setup is fixed and known, the optimal value δ^* and θ_0^* will not differ much from the initial guess δ^0 and θ_0^0, which means it does not take much computational cost to densely sample different solution pairs in specified solution space to find the optimal solution pair.

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.

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 basic building blocks for various robotics applications.

In the experiment, we use the eFAST[8] method to detect corner features directly in event stream. The refraction angle of the wedge prism is set as 0.5 degree, because when the refraction angle become larger, the amount of events per one rotation period also be larger, while the difficulty for compensation also increase since the error is also amplified. The 0.5 degree is a good balance between event density and compensation performance. The rotation speed is set as 12.5 hz, it is also a good balance between system performance and accuracy. Because higher speed generates more environmantal information, while introducing more compensation error due to time synchronization.

The result is illustrated in Fig. 5, in which the camera is moving along its Y-axis. The Fig. 5(a) shows standard event camera cannot detect the corner features robustly in this circumstance, because the information is incomplete for corner detection as there is barely no motion along X-axis, which does not trigger any event belongs to horizontal edges. In Fig. 5(b), the proposed system can always detect corner features accurately and robustly because of the introduced micro-motion and its compensation.

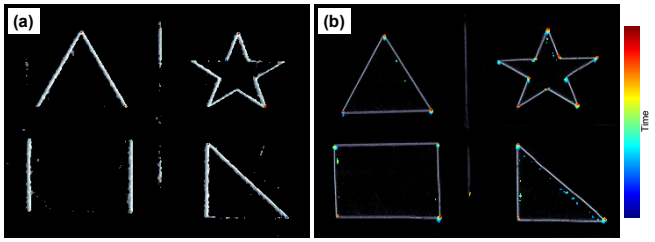

Fig. 5. Experimental result of feature detection application. The camera is moving along its Y-axis. (a): Feature detection without the proposed texture-enhanced event camera system. (b):Feature detection with the proposed system.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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

[6] 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.

[7] 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.

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

APPENDIX

A. *Error analysis of approximation for rotating wedge lens refraction*

In our approximation of light refraction for rotation wedge lens, we approximate the radius of the circular trajectory (approximating with a circle), noted as δ, instead of using different δ_i for different rotation angle θ_i. Based on the error analysis in Fig. 6, for a wedge prism with a 0.5 degree refraction angle, for any point on a 640 by 480 event count image with 90 degree field of view, the maximum error in this approximation is 0.09 degree, which is less than 2 pixels for DVXplorer camera. This approximation error is small enough that we can safely ignore it for real-world robotics application scenarios.

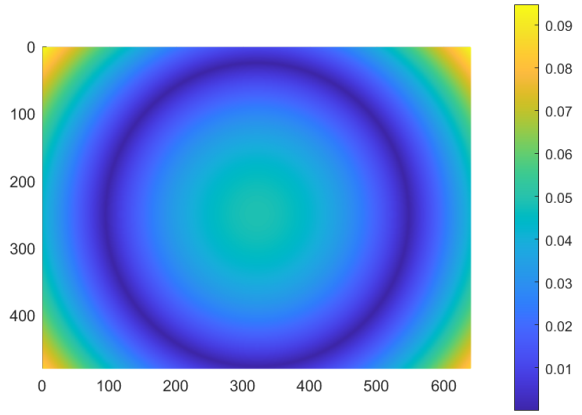

Fig. 6. Illustration of the error introduced by calibration and compensation.