A Dynamic Calibration Framework for the Event-frame Stereo Camera System

Rui Hu[1], Jürgen Kogler[2], Margrit Gelautz[2], Min Lin[1], and Yuanqing Xia[3]

***Abstract*—The fusion of event cameras and conventional frame cameras is a novel research field, and a stereo structure consisting of an event camera and a frame camera can incorporate the advantages of both. This paper develops a dynamic calibration framework for the event-frame stereo camera system. In this framework, the first step is to complete the initial detection on a circle-grid calibration pattern, and a sliding-window time matching method is proposed to match the event-frame pairs. Then, a refining method is devised for two cameras to get the accurate information of the pattern. Particularly, for the event camera, a patch-size motion compensation method with high computational efficiency is designed to achieve time synchronization for two cameras and fit circles in an image of warped events. Finally, the pose between two cameras is globally optimized by constructing a pose-landmark graph with two types of edges. The proposed calibration framework has the advantages of high real-time performance and easy deployment, and its effectiveness is verified by experiments based on self-recorded datasets. The code of this paper is released at: http://github.com/rayhu95/EFSC_calib.**

***Index Terms*—Calibration and Identification; Sensor Fusion; Event Cameras; Event-frame Stereo Camera System**

I. Introduction

EVENT cameras are novel bio-inspired silicon retina cameras which are attracting the attention of researchers over the past dozen years [1]. They have been applied in robotic localization [2]–[4], object recognition [5], [6], vehicle detection [7] and various computer vision fields. Each pixel of event cameras can independently detect brightness changes with high temporal resolution (μs) and then a stream of events is output asynchronously. This way is completely different from conventional frame cameras, which are the dominant cameras in the field of computer vision, capturing images at a fixed rate (e.g. 30 fps). These two types of cameras are complementary in different movement scenarios and data types. Event cameras have great potential at high-speed challenging scenarios, while having no response to stationary objects. In contrast, frame cameras may cause motion blur due to automatic exposure when fast movements occur inside their views [1], but work well in low speed or stationary scenarios. In terms of data types, the data acquired by frame cameras can hold rich image details that are not available with event cameras, while the event stream generated by event cameras can provide additional event information between frames.

By fusing the two types of cameras, their respective advantages can be fully utilized to enhance the adaptability to different motion scenarios and the informativeness of the data, thus better adapting to challenging applications, such as robust localization under fast motion [3], [4], and object detection in challenging environments [6]. The most commonly used structure is the event-frame stereo camera (EFSC) structure, which takes inspiration from traditional stereo vision structure [8], [9]. In this structure, an event camera and a frame camera are mounted on two viewpoints to observe the same scene with a baseline connected to each other. However, the calibration of such a system is still an open field, and this paper aims to address this problem.

Different from the traditional stereo calibration, the calibration of the EFSC system mainly faces the following two challenges. The first one is the fusion of different types of data from two cameras, especially in terms of time synchronization. The second one is the need for a dynamic calibration. The dynamic calibration here refers to matching each image with the event segment of the corresponding scene in the event stream by dynamically changing the relative positions of the EFSC system and the calibration pattern. It is worth mentioning that the event segment corresponding to an image is typically small and provides very limited information, so a larger-sized event segment is taken to aggregate information. To address these challenges, this paper utilizes the estimated motion parameters and the event segments to aggregate the information from the event stream into the timestamps of the frame-images. Further, the accurate information about the calibration pattern is extracted to construct a graph optimization for calibration. In this way, a novel dynamic calibration framework is developed for the EFSC system using a circle-grid pattern. The contributions of this paper are listed as follows.

- To the best of the authors' knowledge, it is the first easy-to-use open-source framework to perform the dynamic calibration of the EFSC system just by moving it in front of the circle-grid pattern.
- A sliding-window time matching method is designed to provide an initial match for the frame-images within the window and the corresponding event segments. This lays the foundation for dynamic calibration.
- A patch-size motion compensation method is devised to

Manuscript received: June, 3, 2024; Revised September, 18, 2024; Accepted October, 20, 2024.

This paper was recommended for publication by Editor Pascal Vasseur upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the National Natural Science Foundation of China under Grant No. 61836001. (Corresponding author: Yuanqing Xia)

[1]R. Hu and M. Lin are with School of Automation, Beijing Institute of Technology, Beijing 100081, China rayhu1007@gmail.com

[2]J. Kogler and M. Gelautz are with Vienna University of Technology, Vienna 1040, Austria juergen.kogler@tuwien.ac.at; margrit.gelautz@tuwien.ac.at

[3]Y. Xia is with Zhongyuan University of Technology, Zhengzhou 450007, Henan, China, and the School of Automation, Beijing Institute of Technology, Beijing 100081, China. xia_yuanqing@bit.edu.cn

Digital Object Identifier (DOI): see top of this page.

estimate the accurate pattern information for calibration in an efficient way. During this process, only the pattern-related events are preserved and aggregated to the timestamp of the frame-image to achieve time synchronization.
- A pose-landmark graph optimization problem is constructed to prevent the error propagation and estimate the pose between two cameras. The results on the designed experiments indicate the effectiveness of the proposed calibration framework.

The rest of the paper is organized as follows. Section II reviews the related work. Section III then presents the methodology of the proposed framework. Detailed experiments are demonstrated in Section IV. Section V concludes the paper.

II. Related Work

A. Methods for event monocular calibration

The current calibration methods [10]–[13] for event cameras are obtaining the camera parameter matrix using a blinking LED pattern or a blinking pattern on the computer screen, but require an additional blinking device. Instead, the calibration method in [14] uses neural-network-based image reconstruction for events, and performs calibration under the standard frame calibration framework. Nevertheless, the reconstructed images from events may introduce issues like lack of edge sharpness or artificial noise, which can impair the calibration process [15].

Further, there are other event monocular calibration methods that extract features of the circle-grid calibration pattern directly from the event stream. E-calib [15] proposes an efficient reweighted least squares method for feature extraction of the calibration pattern circles. The monocular calibration method in [16] introduces the concept of dynamic calibration for event cameras, and the idea of dynamic calibration inspires the work of this paper. It extracts the circle features from events, and estimates a continuous-time trajectory for the optimization. It is worth mentioning that the methods above only consider the monocular event camera calibration.

B. Event-frame stereo calibration

Traditional stereo calibration for frame cameras is the process that first detects features from images and then determines the intrinsic and extrinsic parameters of two cameras [8]. Different from the traditional stereo calibration, the calibration of EFSC structure is a novel area of ongoing exploration. EF-calib [17] designs a new calibration pattern that combines circles and checkerboard crosspoints. It proposes a spatiotemporal calibration method that accounts for deformed circle features of events and generates a piece-wise trajectory to facilitate data alignment with the frame camera. Considering that crosspoints inside circles may introduce additional noise events, we follow the general convention of using a pure circle-grid pattern [15], [16] in our paper. During calibration, fewer events are triggered by the edges on the pattern circle that are parallel to the motion [1]. So this paper proposes to extract the edge points which trigger more events to perform the calibration. The idea of utilizing the edge information of the pattern circle comes from an edge-based stereo algorithm for disparity estimation in [18].

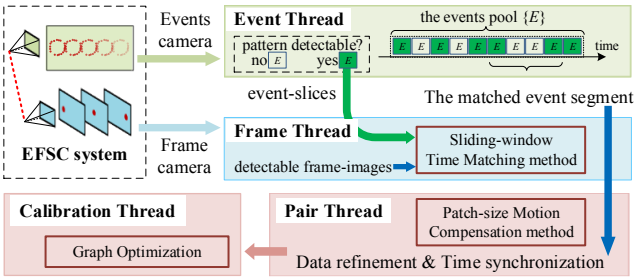

Fig. 1. The calibration diagram for the EFSC system.

C. Time synchronization

In the EFSC structure, each camera generates data at its own frequency aligned with an inner clock. Some cameras carry out time synchronization via hardware interfaces with synchronization pulses [14], but such interface may not be available for all the camera devices. In contrast, time synchronization using software is easier to implement, which is also adopted by the proposed framework. Specifically, instead of transmitting synchronization pulses, the time synchronization of this paper loads the data with the inner timestamps into the same software environment[1], and performs the data alignment of the event and frame cameras. We noted that the motion compensation methods [19]–[21] can warp a period of events into a reference timestamp to form a sharp image. By drawing on this method, this paper sets the reference time to the timestamp of the frame-image, so that both cameras observe the calibration pattern at the same time, thus achieving time synchronization of the EFSC structure.

III. Methodology

The calibration diagram of the EFSC system is shown in Fig. 1, which includes four threads to process data in parallel. First, the event thread receives the event stream in the form of packets E, and detects each event packet to identify the event-slice which contains the detectable calibration pattern. Meanwhile, the frame thread examines whether each incoming frame-image has a detectable pattern, and matches the detectable ones within the sliding-window to the event-slices. Each matched event-slice is then expanded into a larger-sized event segment. Next, the pair thread refines the data to get the accurate pattern circles and carries out the time synchronization for two cameras. Finally, the calibration thread constructs a pose-landmark graph and gets the calibration results based on the graph optimization.

A. Event stream

The event stream is generated asynchronously at different pixels with microsecond (μs) resolution. The k-th event in the event stream $\{e_k\}$ is denoted as $e_k = \{\mathbf{x}_k, p_k, t_k\}$, where $\mathbf{x}_k = (u_k, v_k)^\top$ represents the pixel position at the camera plane, the polarity $p_k = \{+1, -1\}$ indicates an increase or decrease in brightness, and t_k is the high-resolution timestamp.

[1]The booting timestamps of both cameras are recorded by the same software environment so that their inner timestamps are converted to the same time axis.

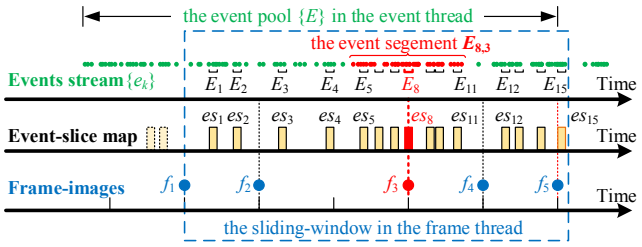

Fig. 2. The sliding-window time matching method. The current sliding-window contains 5 frame-images and 15 event-slices, and the frame-image f_3 is matched at the current round.

The event camera naturally responds to edge information in the scenarios, thus the events generated by the edges constitute the main component of the event stream. Therefore, the non-pattern edge-triggered events and wrongly-triggered noisy events should be filtered out before calibration.

B. Detecting the calibration pattern

The calibration framework starts by detecting the pattern information from the event stream and the frame-images.

1) Pattern detection for the event stream: The information of one event is limited, so the packets of events $E \in \{e_k\}$ are exploited to detect the pattern information. The size of E is either a fixed number or a fixed time duration, and this paper uses the fixed time duration of $5\,\mathrm{ms}$[2]. The event thread identifies each event packet E containing the circle-grid pattern as a detectable one E_i, and stores the estimated circles from E_i into the i-th event-slice es_i.

The *Soft Feature Extraction* mechanism proposed in [16] is employed to estimate the center position C_i^m and the radius R_i^m of the m-th pattern circle from E_i. In this mechanism, E_i is first divided into positive and negative clusters based on the polarities p_k of events. Then, for each positive cluster, it finds its nearest negative cluster so that the two matched clusters belong to the same circle. Thus C_i^m and R_i^m for each circle can be estimated according to the line between two cluster centers. This mechanism can get the initial pattern information in real time, and ensure that the estimated C_i^m is surrounded by those events belonging to the m-th circle.

For these estimated circles, a mature method *findCirclesGrid* implemented in OpenCV-lib [9] is used to determine if their arrangement conforms to the calibration pattern. If so, the estimated C_i^m and R_i^m, as well as E_i, are stored into es_i, which in turn is pushed into the event-slice map $\{es_i\}_{i=0,1,\ldots,n_{es}}$ and fed into the frame thread. Note that n_{es} is the size of the event-slice map, and the middle timestamp of the E_i's time duration is adopted as the order of map storage just for convenience. In addition, all the recent event packets E will be stored into the event pool $\{E\} \in \{e_k\}$ with a fixed capacity, regardless of whether the packet is detectable or not.

2) The sliding-window time matching method: Every time a frame-image arrives in the frame thread, the aforementioned method *findCirclesGrid* [9] is utilized to determine whether the frame-image has a detectable pattern. The detectable ones are labeled as f_j and inserted into the sliding-window. Then, the sliding-window time matching method is used to find the matched event-frame pairs within the window. Such an event-frame pair contains the matched event-slice es_i and frame-image f_j with an event segment $E_{i,j}$. As the example shown in Fig. 2, the matched pair of event-slice es_8 and frame-image f_3 is chosen at the current round, and $E_{8,3}$ is the fetched event segment. Details of this method are as follows.

Firstly, the smallest time difference $\Delta t_{\min}$ between the event-slice map and the frame-images needs to be found. For each frame-image f_j inside the sliding-window, the closest event-slice can be easily identified in the current available map $\{es_i\}$ which is organized by time. The time difference between each f_j and its closest event-slice is denoted as Δt_j. By comparing these Δt_j for all the frame-images within the window, the smallest value $\Delta t_{\min}$ can be found. Thus, the matched event-slice and frame-image corresponding to $\Delta t_{\min}$ are found at the current round. Besides, an event segment $E_{i,j}$, which has a larger size $N_{E_{i,j}}$ than E_i, is fetched from the event pool $\{E\}$ for the purpose of refining data in Section III-C.

Secondly, the above event-frame pair is sent to the pair thread in Fig. 1 if the following conditions are satisfied: (i) $\Delta t_{\min}$ is less than a prescribed threshold (theoretically half of the frame camera frequency, such as $0.5/30\,\mathrm{fps} = 16.7\,\mathrm{ms}$) and (ii) the matched event-slice is not the newest one in the map. The purpose for the latter is to wait for the event thread to store the new detectable event packet E_i into $\{E\}$, so that $E_{i,j}$ can fetch more complete information. For example (see Fig. 2), es_{15} and f_5 are the matched event-frame pair at current round but E_{15} has not been stored into $\{E\}$. In this way, if es_{15} and f_5 are re-matched in the next round, the event segment $E_{15,5}$ can fetch more event packets as the window slides.

Finally, the window slides along the timeline. Each time a new f_j is inserted, the sliding-window is updated by removing the oldest one if no matched event-frame pair is sent to the pair thread. Otherwise, the matched pair will be removed. In this way, the size of the sliding-window is determined by the fixed number of frame-images. Besides, the oldest event-slices that exceed the window are deleted from the map $\{es_i\}$.

C. Data refinement for two cameras

The pair thread in Fig. 1 involves a refining method to obtain the accurate pattern information based on the received event-frame pairs. Inspired by the works in [19]–[21], a patch-size motion compensation method is proposed for events, which is illustrated in Fig. 3. It solves the time synchronization problem and fits the accurate circles in an image of warped events (IWE) in real time. Additionally, the edge detection method is optional to refine the pattern circles for the frame-image.

1) Forming the IWE: Due to the characteristics of the event camera, the events triggered by a moving edge show a trajectory on the pixel plane (see Fig. 3(a)). For a short time duration, the original motion compensation method in [19] has the ability to aggregate those events to a reference time t_{ref} along the motion direction, and form an IWE (see the lower

[2]Through experimental testing, $5\,\mathrm{ms}$ is the most appropriate time duration that avoids the event packets containing too little information and allows for sufficient event-slices in the subsequent matching.

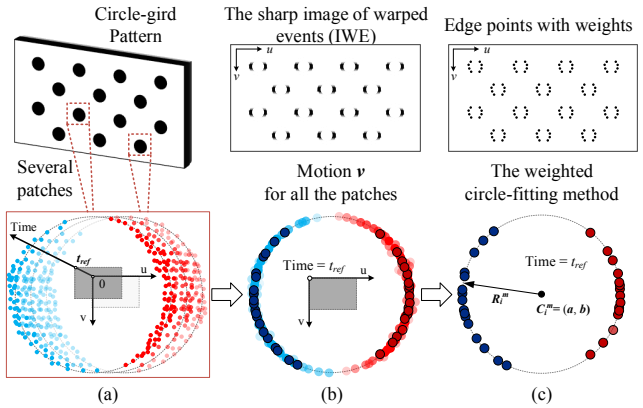

Fig. 3. Patch-size motion compensation method for events. (a) Divide the circle-grid pattern in the event segment $E_{i,j}$ into patches, each with events belonging to the same identified circle; (b) Estimate the motion $\mathbf{v}$ using the events of several patches and apply it to all the patches to form the IWE; (c) Select pixels with high weights in the IWE as edge points.

panel of Fig. 3(b)). Note that t_{ref} can be any time inside this time duration.

Following the local-flow-constancy hypothesis in [22], the two-dimensional motion $\mathbf{v}$ for the related event segment $E_{i,j}$ is assumed to be constant since its time duration is in the range of milliseconds. Firstly, each event e_k is warped into a reference time t_{ref} according to

$$\mathbf{x}'_k = \mathbf{x}_k - (t_k - t_{ref})\mathbf{v}, \tag{1}$$

where $\mathbf{x}'_k = (u'_k, v'_k)^\top$ is the position of the warped event.

Secondly, these warped events are accumulated at every pixel on the matrix $\mathbf{H}(\mathbf{v})$, which becomes the IWE if $\mathbf{v}$ is chosen properly. Its element $H(\mathbf{x}^*, \mathbf{v})$ is given as

$$\begin{aligned} H(\mathbf{x}^*, \mathbf{v}) &= \sum_{e_k \in E_{i,j}} d_p(\mathbf{x}^*, \mathbf{x}'_k), \\ d_p(\mathbf{x}^*, \mathbf{x}'_k) &= \begin{cases} \delta(\mathbf{x}^* - \text{round}(\mathbf{x}'_k)), & \textbf{Mode 1}, \\ (1-|u^*-u'_k|)(1-|v^*-v'_k|), & \textbf{Mode 2}, \end{cases} \end{aligned} \tag{2}$$

where $\delta(\cdot)$ is Dirac delta function, $\mathbf{x}^* = (u^*, v^*)^\top$ is the pixel of $\mathbf{H}(\mathbf{v})$, and $H(\mathbf{x}^*, \mathbf{v})$ sums all the events which are warped to $\mathbf{x}^*$. Typically, the pixel position of the warped event $\mathbf{x}'_k$ obtained from Eq. (1) is non-integer and thereby won't fall exactly into $\mathbf{x}^*$. So two modes are devised to handle it: Mode 1 is rounding $\mathbf{x}'_k$ to the nearest pixel; Mode 2 is to interpolate $\mathbf{x}'_k$ to the four surrounding pixels $\mathbf{x}^*$ according to the distance, which is adopted for the following experiments in our paper.

Thirdly, the proposed framework sets t_{ref} as the timestamp of the matched frame-image to achieve time synchronization. Hence, the IWE formed at t_{ref} observes the same pattern simultaneously as the frame-image.

2) Dividing events into patches: Instead of utilizing all the events in the event segment $E_{i,j}$ as in the original motion compensation method [19], our method is performed in the form of patches. Specifically, the event segment $E_{i,j}$ is divided into event patches, each of which corresponds to an individual identified circle. This division is realized by means of the Kd-tree [23] which stores all the pixel positions of events in $E_{i,j}$. The Kd-tree searches for the events belonging to a single patch based on the C_i^m and R_i^m that have been previously determined for each circle in es_i, and its search range is centered on C_i^m with a radius of $2 \cdot R_i^m$. In addition, the maximum number of events in a patch is a constant N_p, so that the computational complexity has an upper bound related to N_p. During this process, the events that are assigned to the pattern circles are preserved, while the wrongly-triggered noisy events outside the circles and the non-pattern events are filtered out.

3) The optimization problem for events: The motion $\mathbf{v}$ is estimated by an optimization problem and applied to all patches to form the IWE, as shown in the upper panel of Fig. 3(b). Considering that all circles have the same motion behavior within a given short time duration, utilizing part of them is sufficient to estimate the motion $\mathbf{v}$ of all circles. Although one patch may be enough, it is recommended that at least two patches be used to enhance the estimation accuracy.

To choose a proper motion $\mathbf{v}$, the patch-size motion compensation method constructs the optimization problem based on the analysis in [21]. The cost function $C(\mathbf{v})$ adopts the squared Frobenius norm $\|\cdot\|_F^2$ for $\mathbf{H}(\mathbf{v})$, and the candidate motion is optimized by maximizing the cost function

$$\max_{\mathbf{v}} C(\mathbf{v}) = \|\mathbf{H}(\mathbf{v})\|_F^2. \tag{3}$$

Its Jacobian matrix is given by

$$\frac{\mathrm{d}C(\mathbf{v})}{\mathrm{d}\mathbf{v}} = \sum_{\mathbf{x}^*} 2H(\mathbf{x}^*, \mathbf{v}) \cdot \nabla H(\mathbf{x}^*, \mathbf{v}), \tag{4}$$

where $\nabla H(\mathbf{x}^*, \mathbf{v})$ is the gradient of $H(\mathbf{x}^*, \mathbf{v})$ with respect to $\mathbf{x}^*$ using the forward difference operator.

4) Fitting pattern circles for events: The accurate circles are fitted by selecting edge points in the IWE. Since each pixel of the IWE records the number of accumulated events triggered by the moving edge within $E_{i,j}$, the edge points can be selected according to the weights of the pixels from the highest to the average, as in Fig. 3(c). Hence, the circle-fitting algorithm in [24] can be utilized to fit the pattern circles in combination with the weights of these edge points. For the l-th $(l = 1, 2, ..., n_l)$ edge point of the m-th circle, the weight at its pixel position $(u_l, v_l)^\top$ is denoted as $w_l = H((u_l, v_l)^\top, v)$, representing the pixel value in the IWE. Then the refined center position $\tilde{C}_i^m$ and radius $\tilde{R}_i^m$ are given by

$$\tilde{C}_i^m = (a, b), \ \tilde{R}_i^m = \sqrt{a^2 + b^2 + c}, \tag{5}$$

where a, b, c are obtained by

$$\begin{aligned} Q \begin{bmatrix} a & b & c \end{bmatrix}^\top &= \begin{bmatrix} \sum_{l=1}^{n_l} w_l(u_l^2 + v_l^2) \\ \sum_{l=1}^{n_l} w_l(u_l^3 + u_l v_l^2) \\ \sum_{l=1}^{n_l} w_l(u_l^2 v_l + v_l^3) \end{bmatrix}, \\ Q &= \begin{bmatrix} 2\sum_{l=1}^{n_l} w_l u_l & 2\sum_{l=1}^{n_l} w_l v_l & \sum_{l=1}^{n_l} w_l \\ 2\sum_{l=1}^{n_l} w_l u_l^2 & 2\sum_{l=1}^{n_l} w_l u_l v_l & \sum_{l=1}^{n_l} w_l u_l \\ 2\sum_{l=1}^{n_l} w_l u_l v_l & 2\sum_{l=1}^{n_l} w_l v_l^2 & \sum_{l=1}^{n_l} w_l v_l \end{bmatrix}. \end{aligned} \tag{6}$$

Here, we refer to Eqs. (5)-(6) as the weighted circle-fitting method. It renders the estimation biased towards the edges which trigger more events. This is consistent with the fact that event cameras are sensitive to edge information. Furthermore, noisy events have less impact on the estimation results.

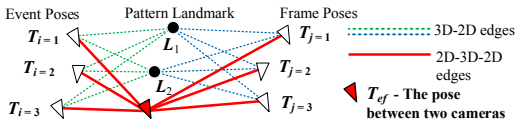

Fig. 4. An illustration example of pose-landmark graph optimization.

5) Frame-image refinement based on Sobel edge: The previously mentioned method *findCirclesGrid* is quite reliable for the pattern detection of frame-images. However, there are a few cases, such as under the low-light conditions, that may cause the center positions of some circles being slightly shifted. Therefore, it is recommended to rectify the circle centers in the frame-image by means of edge detection, employing the same Sobel detector as in [18]. The circle-grid pattern is first chopped from the frame-image based on the results generated by *findCirclesGrid*, and then the chopped image is converted into a gradient image using the 3×3 Sobel operator [9]. In this way, the circles are fitted again using the circle-fitting algorithm [24] when enough pixels are collected from the above gradient image.

D. Calibration for EFSC system based on graph optimization

The above methods are designed to obtain the accurate information about the circle-grid pattern, which is utilized by the graph optimization for the calibration. The traditional stereo calibration is to calibrate the intrinsic parameters of each camera first, and then obtain the extrinsic parameters between two cameras. In terms of the intrinsic parameters, they are available for most camera devices. Beside, the intrinsic matrix can be easily obtained via the Zhang's method [25] based on the accurate pattern information acquired by the above methods. Therefore, this section focuses on the extrinsic calibration, and the pose-landmark graph based on the g2o optimization [26] is then constructed as follows to estimate the pose between two cameras $\mathbf{T}_{ef}$.

Firstly, the 2D-3D edges of the pose-landmark graph are constructed in the type of *BaseUnaryEdge* [26] (as illustrated by the dashed lines in Fig. 4). In the pose-landmark graph, the circle centers on the calibration pattern are the 3D landmarks $\{\mathbf{L}_*\}$, the frame poses $\{\mathbf{T}_j\}$ and the event poses $\{\mathbf{T}_i\}$ are pose vertices, and the edges connect each pose vertex with each landmark. Further, for each pose $\mathbf{T}_{i/j} = [\mathbf{R}\ \mathbf{t}; \mathbf{0}_{1\times3}\ 1]$, the measurement for the m-th 3D landmark $\mathbf{L}_m$ is the 2D refined center $\tilde{C}^m$ on the pixel plane. Specifically, the projection error of a 2D-3D edge is

$$\begin{aligned} P^m_{2D-3D} &= \mathbf{R}\cdot\mathbf{L}_m + \mathbf{t} - \pi_c^{-1}(\tilde{C}^m)\cdot d^m, \\ d^m &= \frac{-row(\mathbf{t},3)}{row(\mathbf{R},3)\cdot\pi_c^{-1}(\tilde{C}^m)}, \end{aligned} \tag{7}$$

where d^m represents the depth whose calculation is identical to that in [16], $\pi_c^{-1}(\cdot)$ is the back-projection from the pixel plane to the normalized camera plane, and $row(*,3)$ is the third row of the matrix $*$. The 2D-3D edges aim to limit the error propagation for camera poses and reach the global optimal.

Secondly, the 2D-3D-2D edges of this graph are constructed in the type of *BaseBinaryEdge* [26] (as illustrated by the solid red lines in Fig. 4). In this graph, the pose between two cameras $\mathbf{T}_{ef}$ is a vertex, the camera poses $\mathbf{T}_{i/j}$ are other vertices, and the edges connect $\mathbf{T}_{ef}$ with each $\mathbf{T}_{i/j}$. Further, the projection has two directions, and a projection example of the direction $es_i(\mathbf{T}_i) \xrightarrow{T_{ef}} f_j(\mathbf{T}_j)$ is given as follows. For an event-frame pair, the 2D refined center $\tilde{C}^m_i$ for the event-slice es_j is projected to the estimated 3D point $\hat{\mathbf{L}}_m = \pi_{ci}^{-1}(\tilde{C}^m_i) * d^m_i$ under the coordinate of the event camera. Here, d^m_i is the depth corresponding to the event-slice and can be obtained similarly by following the calculation of d^m in Eq. (7). Then $\hat{\mathbf{L}}_m$ is transformed into the coordinate of the frame camera through $\mathbf{T}_{ef}$, and is subsequently projected into the 2D pixel plane at f_j. Further, the projection error for a 2D-3D-2D edge in this direction is

$$P^m_{2D-3D-2D} = \pi_{cj}\left(\frac{\mathbf{T}_{ef}\cdot\hat{\mathbf{L}}_m}{d^m_j}\right) - \tilde{C}^m_j, \tag{8}$$

where d^m_j is the depth corresponding to the frame-image and can be obtained similarly by following a similar calculation in Eq. (7). The 2D-3D-2D edges in the other direction are constructed similarly. Note that the estimated $\hat{\mathbf{L}}_*$ is not the given 3D landmark $\mathbf{L}_*$ on the pattern but the 3D point observed by two cameras at the same time.

Finally, the two types of edges are fed into the same graph optimization problem and optimized using the g2o graph optimization with Huber robust kernel function. Moreover, the initialization of each pose vertex $\mathbf{T}_{i/j}$ in this graph optimization problem is acquired following the PnP method [27] within a RANSAC scheme. By minimizing the projection errors given by Eqs. (7) and (8), the graph optimization yields poses of each camera as well as the pose of two cameras $\mathbf{T}_{ef}$.

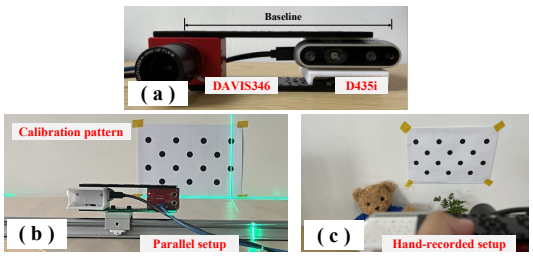

Fig. 5. Two testing setups for calibration. (a) The prototype of the EFSC structure; (b) The parallel testing setup; (c) Hand-recorded testing setup in a variety of perspectives and positions.

IV. Experiments

A. Experimental setup

1) Calibration pattern: In the proposed calibration framework, the preparation of the calibration pattern is very simple: print out the circle-grid pattern with specific parameters on a blank sheet of paper. In our work, a calibration pattern consisting of 14 circles is adopted (see Fig. 5). The parameters of the circles are $17.5\,\mathrm{mm}$ in diameter and $55\,\mathrm{mm}$ between centers. Besides, the $Z = 0$ plane of the world coordinates $X-Y-Z$ is defined as the pattern plane, and the coordinate origin is located at the center of the first pattern circle.

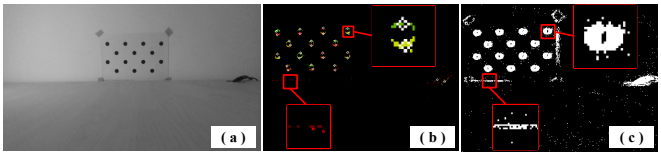

Fig. 6. The matched pair from dataset $D_{ef} = 60\,\text{cm}$. (a) Detectable frame-image; (b) Clustering centers in the event-slice; (c) Binary image of the fetched event segment $E_{i,j}$.

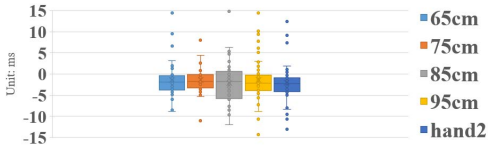

Fig. 7. The box-plot of the minimal time differences $\Delta t_{\min} = t_{es} - t_f$ for all the matched event-frame pairs in each dataset.

2) Prototype of the EFSC system: The prototype of the EFSC system contains a classical DAVIS346 camera with 346×260 pixel resolution [28] and an 848×480 color camera module of Intel RealSense D435i. To build the EFSC structure, a connector is needed for the D435i, which is made by a 3D printer. Following the traditional stereo setup, the pixel planes of the two cameras are mounted in parallel, hence the rotation matrix between them is assumed to be an identity matrix.

3) Testing setups: All the datasets are recorded by the Robot Operating System (ROS), and the code (in C++ language) is implemented on a laptop with the 11th Gen Intel(R) Core(TM) i7-11800H CPU and Ubuntu 18.04 operating system. There are two testing setups, as shown in Fig. 5. (i) Parallel: To facilitate the evaluation of distance, the prototype of the EFSC system is mounted on a straight slide rail, which is parallel to the calibration pattern. In this setup, the translation between the two cameras is $(X, Y, Z) = (16.8\,\text{cm},\ 0.5\,\text{cm},\ 3.0\,\text{cm})$. (ii) Hand-recorded: The datasets are recorded by manually moving the prototype in a variety of perspectives and positions. These three datasets are recorded at three different baselines of the EFSC structure, that is, $X = 12.2\,\text{cm}$ (*hand1*), $16.8\,\text{cm}$ (*hand2*) and $20.8\,\text{cm}$ (*hand3*).

4) Parameters of the designed methods: The number of frame-images in the sliding-window is chosen as 5 according to the frame-rate and computational power. The size of the event segments fetched from the event pool is $N_{E_{i,j}} = 100\,\text{ms}$, and the maximum number of selected events for each patch is $N_p = 1000$. For the graph optimization, the threshold parameter of the Huber robust kernel function is 0.3 for all the 2D-3D edges, and 0.5 for the 2D-3D-2D edges.

5) Ground truth: The DAVIS346 has an APS module that additionally provides frames (not all event cameras have this function). Here, we consider the calibration results using OpenCV-lib [9] for the APS frames and D435i frames as ground truth (GT), and compare them to our calibration framework that uses events directly. In this case, the pattern circles of the APS frames are extracted with subpixel precision.

B. Results of each part of the proposed framework

The datasets with the same baseline $X = 16.8\,\text{cm}$ are chosen to test each part of the proposed framework, including all the "parallel" datasets at different distances D_{ef} and the dataset "*hand2*". The details of these datasets are listed in TABLE I, and each dataset contains the time duration of events T_e and the number of the detectable frame-images n_f.

1) Results of the sliding-window time matching method: The sliding-window time matching method designed in Section III-B provides an initial match for the two types of data.

TABLE I
THE TESTING RESULTS AND SOME EVALUATIONS (UNIT: cm)

	D_{ef}	60 cm	70 cm	80 cm	*hand2*
Dataset details	T_e	2.71 s	1.53 s	2.90 s	6.73 s
	n_f	62	44	51	117
	n_{es}	293	255	374	604
Pairs	n_{ef}	53	42	48	95
Depth	d_e	57.08	67.28	76.78	47.7–69.7
	d_f	60.60	70.67	79.57	50.2–72.6
q_{ef}	q_i	0.0019	0.0006	-0.0000	0.0003
	q_j	0.0075	0.0073	0.0060	0.0039
	q_k	0.0008	-0.0005	0.0013	-0.0027
	q_w	0.9999	0.9999	0.9999	0.99998
t_{ef}	t_x	15.946	15.979	15.989	15.945
	t_y	0.5947	0.4736	0.3987	0.4979
	t_z	3.2086	3.1129	2.7706	3.0624
Graph optimization	E_e	1.2181	1.1447	1.2465	0.8371
	E_f	0.6678	0.6719	0.6133	0.7965
	E'_e	0.3332	0.3799	0.1827	0.3195
	E'_f	0.6474	0.6228	0.2151	0.3069
E_T		0.5434	0.4915	0.7448	0.9916

[1] E_e and E_f are the average PnP projection errors, respectively.
[2] d_f and d_e are the average depths estimated at different positions for the frame and event cameras, respectively (the theoretical difference between the two is 3 cm), while D_{ef} is the distance between the frame camera and the pattern plane.
[3] The rotation between two cameras is presented in the form of a quaternion $q_{ef} = (q_i, q_j, q_k, q_w)$ and the translation is $t_{ef} = (t_x, t_y, t_z)$. E_T is the average projection error from event camera to frame camera.

As shown in the event-frame pair in Fig. 6, a detectable frame-image and an event-slice are matched, and the event segment $E_{i,j}$ (displayed as a binary image in Fig. 6(c)) with the larger size of events than the event-slice is fetched. It can be observed from TABLE I that the number of matched event-frame pairs n_{ef} is rather close to n_f. In other words, this method provides each frame-image with a corresponding event segment that is larger than the matched event-slice.

In addition, each event-frame pair has a minimal time difference $\Delta t_{\min}$ and the box-plot of these $\Delta t_{\min}$ is illustrated in Fig. 7. Obviously, most $\Delta t_{\min}$ are within $(0, -3\,\text{ms})$, which indicates that the event-slices enter the sliding-window slightly slower than the frame-images. This is primarily caused by the fact that the sliding-window exits in the frame thread and waits for the latest event packet to be detected as an event-slice. This in turn confirms the rationality of the designed (ii) matching condition in Section III-B-2). Besides, the sliding-window time matching method can run in real time ($0.15\,\text{ms}$ on average in TABLE II).

2) Results of the patch-size motion compensation method for the event-slices: Following the methods presented in Section III-C-1) to 4), the accurate circles are fitted on the IWE, and the time synchronization is carried out by warping

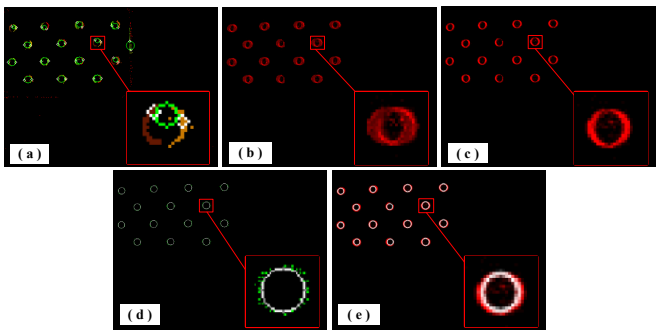

Fig. 8. The patch-size motion compensation method for dataset $D_{ef} = 60\,\text{cm}$. For clarity, events are represented in red. (a) Event-slice before data refinement; (b) Accumulated grayscale image of all patches obtained from the event segment $E_{i,j}$, whose pixels indicate the number of events; (c) IWE after applying the patch-size motion compensation method; (d) The weighted circle-fitting method for the edge points; (e) The fitted circles re-drawn on the IWE.

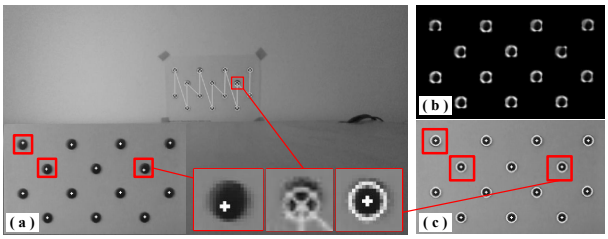

Fig. 9. The data refinement for the frame-image. (a) The frame-image with detectable calibration pattern; (b) Sobel edge detection on the pattern area; (c) The results of the fitted circles.

the events in the event segment to the timestamp of the matched frame-image. An example of refining the event-slice is shown in Fig. 8. First, the method in Section III-B-1) (see Fig. 8(a)) only provides the initial estimation of pattern circles for the event-slice. Then, the fetched event segment $E_{i,j}$ is divided into patches, and only the pattern-related events are preserved. A grayscale image that accumulates all the patches is visualized in Fig. 8(b). It can be observed that the non-pattern events and wrongly-triggered noisy events outside the circles are filtered out, compared with the binary image formed by the entire $E_{i,j}$ in Fig. 6(c). Next, the motion $\mathbf{v}$ is estimated through events of several patches using Eqs. (1)-(4), and applied to all the patches to form the IWE. Fig. 8(c) shows a sharp image (IWE), which aggregates the events triggered along the motion direction in Fig. 8(b) to the reference time t_{ref}. Finally, edge points are selected based on the high weights in IWE, and the accurate circles are obtained by those edge points using the weighted circle-fitting method (see Fig. 8(d)). The fitted circles are drawn on the IWE again to show the accurate result, which is shown in Fig. 8(e).

3) Results of the data refinement for the frame-images: When the estimated circles are shifted from their actual positions (see Fig. 9(a)), the refined circles in Fig. 9(c) are fitted by selecting edge points on the gradient image in Fig. 9(b) using the Sobel operator. TABLE II shows that the average running time of this method is $6\,\text{ms}$ if it is applied for every round in the pair thread.

4) The running time: The running time at different sizes $N_{E_{i,j}}$ for $E_{i,j}$ is listed in TABLE II. Obviously, the patch-size motion compensation method has less running time since the original motion compensation method in [19] uses all the events to estimate the motion. Besides, the running time of the original one is linearly increasing as $N_{E_{i,j}}$. In contrast, the patch-size motion compensation method has a maximum running time, because the size for each patch is fixed at the maximum value $N_p = 1000$ even though $N_{E_{i,j}}$ increases.

TABLE II
RUNNING TIME FOR THE DATASET $D_{ef} = 80\,\text{cm}$ AT DIFFERENT SIZES OF EVENT SEGMENT $N_{E_{i,j}}$ (UNIT: ms)

$N_{E_{i,j}}$		50 ms	100 ms	150 ms
Event numbers	avg.	16531	27115	38922
Time matching	avg.	0.145	0.148	0.146
	max	0.210	0.223	0.209
Patch-size MC	avg.	2.015	2.808	3.089
	max	2.623	4.884	4.814
Original MC	avg.	3.160	4.799	5.722
	max	5.243	7.589	11.556
Sobel-edge (frame)	avg.	6.078	6.127	6.008
	max	7.484	7.509	6.736

[1] MC is short for the motion compensation method.

5) Results of the graph optimization: A pose-landmark graph (described in Section III-D) is constructed after collecting the refined data, and the results are shown in TABLE I.

Benefiting from the design in Eq. (7), the error propagation of the camera poses $\mathbf{T}_{i/j}$ is limited since the average projection errors E'_f and E'_e (after optimization) are less than E_f and E_e (before optimization), respectively. In addition, the translation t_{ef} gives an accurate estimation as the average projection error E_T is low, and the estimated rotation q_{ef} is almost the unit quaternion $(0, 0, 0, 1)$. It is worth noting that the dataset *hand2* also shows the same calibration results when it is recorded at different calibration distances in a variety of perspectives and positions. Thus, the graph optimization can reach high numerical accuracy and globally minimize the errors for all the pose vertices.

C. Intrinsic calibration

In our paper, the event camera follows the same perspective projection model as the frame camera. Specifically, the intrinsic parameters used for each camera are modeled as pinhole cameras with distortion parameters.

Our calibration framework is compared with two state-of-the-art event intrinsic calibration methods using the same pattern [15], [16]. The reprojection errors [8] for each method are presented in TABLE III. It can be seen that our calibration framework performs well on all three datasets and can ensure the accuracy of intrinsic calibration.

D. Extrinsic calibration and ablation study

To further analyze the performance of each part of the proposed calibration framework, an ablation study is provided in the extrinsic calibration experiments. The projection error from the event camera to the frame camera is used as the evaluation metric of the calibration results (see TABLE IV). Comparing

TABLE III
COMPARISON RESULTS OF THE INTRINSIC CALIBRATION USING THE REPROJECTION ERROR (UNIT: PIXEL)

	hand1	*hand2*	*hand3*
Dynamic calibration [16]	0.3961	0.2683	0.4283
E-calib [15]	**0.1734**	0.1562	0.1737
Ours	0.2092	**0.1539**	**0.1577**
Frame-based	0.2422	0.1930	0.2042

TABLE IV
ABLATION STUDY ON THE EXTRINSIC CALIBRATION RESULTS

Sobel detection	Patch-size MC	Graph optimization	*hand1*	*hand2*	*hand3*
✓		✓	1.8113	2.0151	1.5908
✓	✓		1.1660	1.1180	2.4593
	✓	✓	0.7734	1.1549	0.9435
✓	✓	✓	**0.6265**	0.9761	**0.8874**
Frame-based (GT)			0.6930	**0.8492**	1.2407

[1] The results are averaged over 10 experiment repetitions, with standard deviations in the order of 10^{-2}.

with the frame-based stereo calibration results, it can be observed that not only our proposed calibration framework has high-precision extrinsic calibration performance, but also each part of the framework contributes to the improvement of the calibration results.

V. CONCLUSION

This paper proposes an easy-to-use dynamic calibration framework for the event-frame stereo camera system. In this framework, an initial match of event streams and the frame-images is provided by the sliding-window time matching method, then the patch-size motion compensation method achieves time synchronization and refines the accurate pattern information. The pose-landmark graph optimization gets the accurate calibration results with limited error propagation, and hand-recorded experimental results show the effectiveness of the proposed calibration framework.

To further enhance the performance of our framework, the future work may include designing mechanisms for the event collapse problem in challenging environments, e.g., heavily-noisy scenes or overload events, or exploring the performance when the pixel planes of two cameras are not parallel.

REFERENCES

[1] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis *et al.*, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2020.

[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] S. Zhu, Z. Tang, M. Yang, E. Learned-Miller, and D. Kim, "Event camera-based visual odometry for dynamic motion tracking of a legged robot using adaptive time surface," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 3475–3482.

[4] 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.

[5] Y. Deng, Y. Li, and H. Chen, "AMAE: Adaptive motion-agnostic encoder for event-based object classification," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4596–4603, 2020.

[6] Z. Zhou, Z. Wu, R. Boutteau, F. Yang, C. Demonceaux, and D. Ginhac, "Rgb-event fusion for moving object detection in autonomous driving," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7808–7815.

[7] J. Li, S. Dong, Z. Yu, Y. Tian, and T. Huang, "Event-based vision enhanced: A joint detection framework in autonomous driving," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019, pp. 1396–1401.

[8] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[9] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., 2008.

[10] RPG-UZH, "Calibration using blinking LEDs or computer screens," https://github.com/uzh-rpg/rpg_dvs_ros/tree/master/dvs_calibration.

[11] VLOGroup at TU Graz, "Automatic camera calibration for DVS cameras," https://github.com/VLOGroup/dvs-calibration.

[12] G. Orchard, "DVS calibration using the caltech camera calibration toolbox," https://github.com/gorchard/DVScalibration.

[13] J. Kogler, F. Eibensteiner, M. Humenberger, M. Gelautz, and J. Scharinger, "Ground truth evaluation for event-based silicon retina stereo data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2013, pp. 649–656.

[14] M. Muglikar, M. Gehrig, D. Gehrig, and D. Scaramuzza, "How to calibrate your event camera," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 1403–1409.

[15] M. Salah, A. Ayyad, M. Humais, D. Gehrig, A. Abusafieh, L. Seneviratne, D. Scaramuzza, and Y. Zweiri, "E-calib: A fast, robust and accurate calibration toolbox for event cameras," *IEEE Transactions on Image Processing*, vol. 33, pp. 3977–3990, 2024.

[16] K. Huang, Y. Wang, and L. Kneip, "Dynamic event camera calibration," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7021–7028.

[17] S. Wang, Z. Xin, Y. Hu, D. Li, M. Zhu, and J. Yu, "EF-Calib: Spatiotemporal calibration of event-and frame-based cameras using continuous-time trajectories," *arXiv preprint arXiv:2405.17278*, 2024.

[18] Z. Wang, L. Pan, Y. Ng, Z. Zhuang, and R. Mahony, "Stereo hybrid event-frame (SHEF) cameras for 3D perception," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 9758–9764.

[19] 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 (CVPR)*, 2018, pp. 3867–3876.

[20] G. Gallego and D. Scaramuzza, "Accurate angular velocity estimation with an event camera," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 632–639, 2017.

[21] H. Kim and H. J. Kim, "Real-time rotational motion estimation with contrast maximization over globally aligned events," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 6016–6023, 2021.

[22] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 2, 1981, pp. 674–679.

[23] J. L. Blanco and P. K. Rai, "nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees," https://github.com/jlblancoc/nanoflann, 2014.

[24] I. Kåsa, "A circle fitting procedure and its error analysis," *IEEE Transactions on Instrumentation and Measurement*, no. 1, pp. 8–14, 1976.

[25] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[26] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3607–3613.

[27] T. Collins and A. Bartoli, "Infinitesimal plane-based pose estimation," *International Journal of Computer Vision*, vol. 109, no. 3, pp. 252–286, 2014.

[28] C. Brandli, L. Muller, and T. Delbruck, "Real-time, high-speed video decompression using a frame-and event-based DAVIS sensor," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2014, pp. 686–689.