

Article

Real-World Spatial Synchronization of Event-CMOS Cameras through Deep Learning: A Novel CNN-DGCNN Approach

Dor Mizrahi ^{1,2,*}, Ilan Laufer ¹ and Inon Zuckerman ¹

¹ Department of Industrial Engineering and Management, Ariel University, Ariel 40700, Israel; ilanl@ariel.ac.il (I.L.); inonzu@ariel.ac.il (I.Z.)

² Applied Physics Division, Soreq NRC, Yavne 81800, Israel

* Correspondence: dor.mizrahi1@msmail.ariel.ac.il

Abstract: This paper presents a new deep-learning architecture designed to enhance the spatial synchronization between CMOS and event cameras by harnessing their complementary characteristics. While CMOS cameras produce high-quality imagery, they struggle in rapidly changing environments—a limitation that event cameras overcome due to their superior temporal resolution and motion clarity. However, effective integration of these two technologies relies on achieving precise spatial alignment, a challenge unaddressed by current algorithms. Our architecture leverages a dynamic graph convolutional neural network (DGCNN) to process event data directly, improving synchronization accuracy. We found that synchronization precision strongly correlates with the spatial concentration and density of events, with denser distributions yielding better alignment results. Our empirical results demonstrate that areas with denser event clusters enhance calibration accuracy, with calibration errors increasing in more uniformly distributed event scenarios. This research pioneers scene-based synchronization between CMOS and event cameras, paving the way for advancements in mixed-modality visual systems. The implications are significant for applications requiring detailed visual and temporal information, setting new directions for the future of visual perception technologies.

Keywords: dynamic graph convolutional neural networks (DGCNN); event-based sensing; sensor fusion; image alignment

Citation: Mizrahi, D.; Laufer, I.; Zuckerman, I. Real-World Spatial Synchronization of Event-CMOS Cameras Through Deep Learning: A Novel CNN-DGCNN Approach. *Sensors* **2024**, *24*, 4050. <https://doi.org/10.3390/s24134050>

Academic Editor: James Covington

Received: 13 May 2024

Revised: 12 June 2024

Accepted: 20 June 2024

Published: 21 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the rapidly evolving field of imaging technology, integrating event-based cameras with traditional CMOS (complementary metal–oxide–semiconductor) sensors represents a significant research frontier with vast application potential. Traditional CMOS sensors have been the backbone of digital imaging for decades, providing high-quality, frame-based captures. However, they face challenges such as substantial power requirements, limited dynamic range, and susceptibility to motion blur in rapid-moving scenarios [1,2]. Conversely, event cameras, a relatively novel innovation, detect changes in light intensity at each pixel independently and asynchronously, only recording changes. This operation leads to lower power consumption, a much higher dynamic range, and the capability to capture fast-moving objects without blur [2].

Despite these advantages, previous studies on event cameras have encountered significant challenges. Traditional methods often aggregate events into grid-based representations for processing with standard vision pipelines, which can lead to inefficiencies and loss of information [3]. Additionally, there is a notable lack of a general framework to convert event streams into grid-based representations that can be learned end-to-end with task-specific networks [4]. Moreover, while event cameras have shown effectiveness for specific applications such as machine fault diagnosis and driving behavior

characterization, these studies have highlighted difficulties in handling environmental noise and achieving high precision in real-world conditions [5].

The output from event cameras comprises a stream of “events” instead of conventional images. Each event represents a change in intensity at a pixel level, occurring independently of a frame structure. This output format can be challenging to integrate into many applications that traditionally rely on continuous frame-based video data, making it difficult to apply standard video processing techniques [3,6]. The potential of combining these two technologies lies in leveraging the continuous, high-resolution imagery of CMOS cameras with the high temporal resolution and motion sensitivity of event cameras. This hybrid approach could significantly enhance fields that require detailed, dynamic visual information, such as robotics [7–9]. While CMOS cameras provide essential high-resolution images for navigation and spotting obstacles, they can falter under rapidly changing light conditions or during high-speed situations. Event cameras, known for their quick response times and minimal delay, excel at capturing sudden changes in a scene, thus reducing motion blur.

Recent advancements in sensor fusion and visual-inertial odometry (VIO) have significantly enhanced the capabilities of autonomous vehicles. A notable approach integrates data from traditional CMOS cameras (frames) and event cameras (events) along with inertial measurements, creating a robust adaptive system. This system excels in complex lighting and high-speed conditions, significantly improving object detection and tracking accuracy. Central to this system is an 8-degrees-of-freedom (DOF) warping model that aligns the different data types by creating brightness increment patches, which help to minimize the differences between these outputs. This process allows for robust feature tracking by adaptively updating measurements based on the quality of tracked features, thus ensuring superior pose estimation accuracy compared to existing event-based VIO algorithms [10].

Nonetheless, achieving precise spatial synchronization between CMOS and event camera outputs remains a significant challenge. This difficulty primarily arises from the fundamental differences in their data acquisition modes. CMOS cameras capture uniform, time-synchronized frames, while event cameras record asynchronous, pixel-level changes triggered by variations in scene lighting. The asynchronous nature of event data does not align neatly with the synchronous frame rate of CMOS cameras, complicating real-time data fusion. Even with sophisticated models like the 8-DOF warping model, temporal alignment requires matching these sporadic, event-driven data with the uniform timestamps of frame data, necessitating complex interpolation and prediction methods that are yet to be perfected. These challenges necessitate advanced synchronization techniques beyond traditional calibration methods, which often fall short in dynamic, real-world conditions where both types of cameras must operate in unison [11]. Previous attempts to transform event camera data into pseudo-images for applying conventional calibration techniques have proven complex and often imprecise, particularly in handling real-time, dynamically changing environments [11,12].

Our work aims to bridge this gap by directly synchronizing the output of event cameras with that of CMOS cameras, bypassing the intermediate transformation steps. We propose a direct method for aligning these disparate data streams, enhancing the accuracy and efficiency of the process. Our approach utilizes deep learning models to interpret and synchronize the raw outputs from both camera types, focusing on real-time, real-world applicability [4,13]. We introduce innovative neural network architectures that are specifically designed to handle the unique characteristics of each camera’s output, facilitating a seamless integration of these technologies. Additionally, the integration of dynamic vision sensors into unmanned aerial vehicles optimizes the estimation of dynamic interactions in real time, demonstrating the effectiveness of event cameras in scenarios that require rapid and precise adjustments [3,14].

Our findings show that this direct synchronization method significantly improves the accuracy and efficiency of data integration from both camera types, overcoming the

limitations of previous approaches. The proposed deep learning architectures enable end-to-end learning and adaptive processing, leading to enhanced feature extraction and event interpretation. Experimental results have demonstrated high fault diagnosis accuracies in machinery monitoring, comparable to traditional accelerometer data, while also being effective in driving behavior characterization and autonomous navigation tasks. These advancements suggest that the method can be applied in diverse real-world scenarios, potentially offering reliable performance in dynamic environments [5,13].

2. Materials and Methods

2.1. Workflow Overview

The flowchart (see Figure 1) outlines the key steps involved in the spatial synchronization of CMOS and event cameras using the CNN-DGCNN model. This workflow covers the entire process from data acquisition to model evaluation.

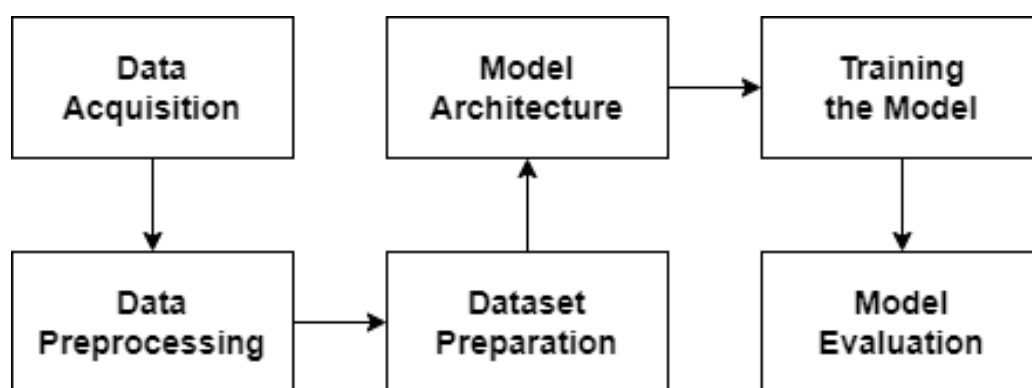


Figure 1. Flowchart—workflow overview.

The process begins with data acquisition, where we collect high-resolution video data from CMOS cameras (480 p and 1080 p) and corresponding event data from event cameras. Next, during the data preprocessing step, we segment CMOS video frames into 100×100 windows and label these segments with precise shifts. Following this, in the dataset preparation phase, we create training and testing datasets from the segmented and labeled data, with the training set containing 3 million observations and the testing set comprising 600,000 observations.

In the model architecture stage, we define a convolutional neural network (CNN) to extract features from CMOS data and a dynamic graph convolutional neural network (DGCNN) to process event data. These outputs are then integrated using a multi-layer perceptron (MLP). During the model training phase, we initialize the model parameters and train the model using the Adam optimizer, applying early stopping and a learning rate scheduler to enhance performance and prevent overfitting. Finally, in the model evaluation step, we assess the model's performance on the test set by computing the calibration error and analyzing the results to ensure accuracy and reliability.

2.2. Event Camera Data Format

Event cameras are specialized sensors that capture local variations in luminance over time. Differing from conventional cameras, which use shutters to take pictures, event cameras consist of pixels that function autonomously and asynchronously. These pixels activate only in response to changes in light intensity, otherwise remaining passive. This mechanism allows for a highly efficient and sparse representation of video sequences which is particularly suited to natural scenes. In essence, when any pixel detects a variation in brightness, it generates an event. This information is aggregated as a sequence of events:

$$\{e_i\}_{i=1}^N = \{(x_i, y_i, t_i, p_i)\}_{i=1}^N \quad (1)$$

In Equation (1), (x_i, y_i) represents the spatial coordinates of the event, t_i denotes the event's timing (with microsecond precision), and p_i signifies the intensity change at the pixel. Consequently, the output from an event camera is a streamlined series of events, structured as follows, with the objective being to apply vision-based downstream processing to these data, specifically for 2D regression tasks.

2.3. Dataset Creation

To facilitate the training of our proposed models, a comprehensive and accurately labeled dataset was essential. This dataset needed to encapsulate both images from the CMOS camera and the corresponding events captured by the event camera, alongside the precise movements on each axis to ensure effective spatial synchronization.

2.3.1. Data Sources

For our study, we selected three datasets from the DAVIS 240C Dataset [12,13], valued for its unique integration of a conventional CMOS camera with an event-based sensor. This integration offers dual perspectives on visual data capture, as demonstrated in Figures 2 and 3.

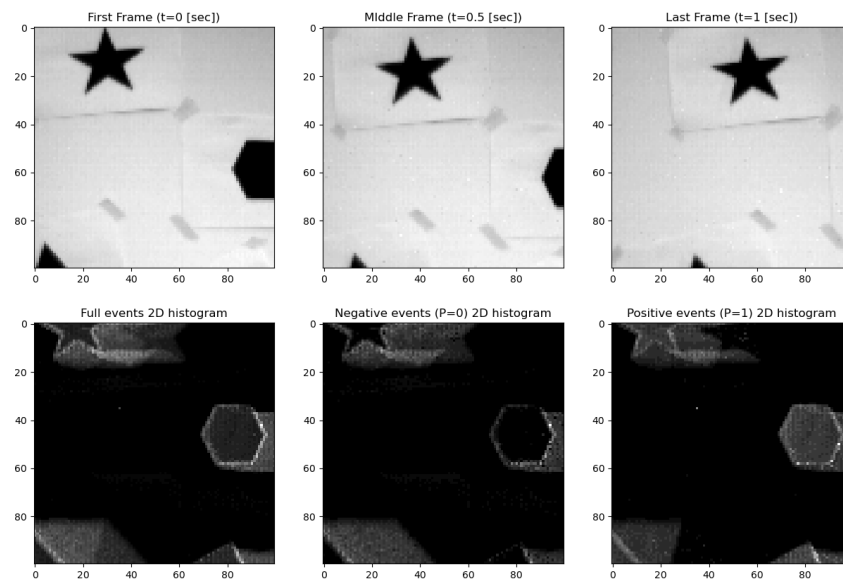


Figure 2. Video frame and corresponding 2D event histogram maps (“shapes_rotation”).

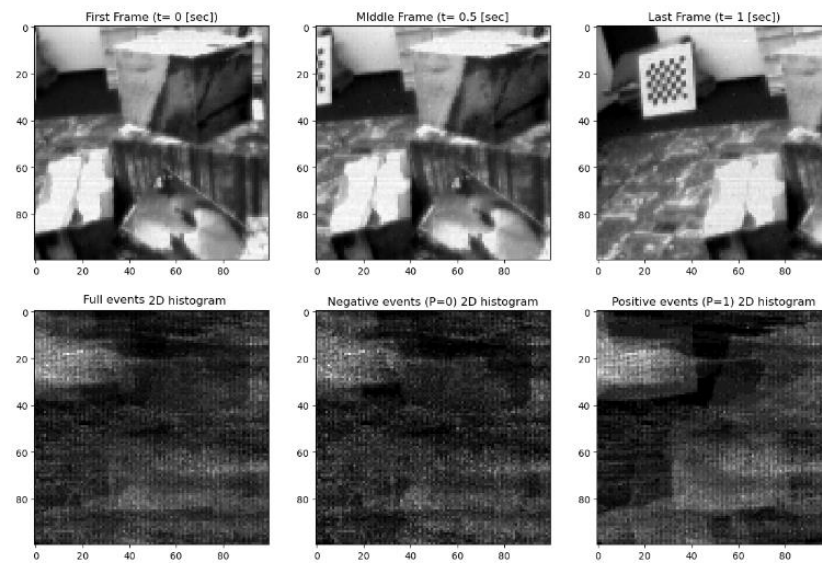


Figure 3. Video frame and corresponding 2D event histogram maps (“boxes_rotation”).

The “shapes rotation” dataset [15], visualized in Figure 2, presents simple geometric shapes against a wall. This dataset provided a controlled environment in which to test the basic synchronization capabilities of our models. Figure 2 illustrates this setup with three distinct frames: the first frame ($t = 0$ s) shows a single stationary shape; the middle frame ($t = 0.5$ s) captures the shape mid-rotation; and the last frame ($t = 1$ s) depicts the rotation’s conclusion, with a change in orientation of the shapes. Below the video frames, corresponding 2D event histogram maps split into full events, negative events ($P = 0$), and positive events ($P = 1$). These histograms capture the pixel-wise changes in intensity due to movement, with the negative and positive events representing decreases and increases in intensity, respectively.

“Boxes rotation” [15,16], as seen in Figure 3, is set within a complex, texture-rich environment to provide a multifaceted pattern of movements, challenging our models under diverse textural conditions. In Figure 2, the first frame ($t = 0$ s) shows the initial static scene. The middle frame ($t = 0.5$ s) captures slight movement, primarily due to the camera’s motion. The last frame ($t = 1$ s) includes more noticeable movement, again primarily from the camera’s perspective. This series of frames illustrates the scene as observed by the camera over a one-second interval. The corresponding 2D event histogram maps below each frame represent the full events, negative events ($P = 0$), and positive events ($P = 1$), highlighting the pixel-wise changes in intensity due to the movement. To produce a tagged dataset with precisely labeled shifts, we reduced and cropped the CMOS segments to smaller windows of 100×100 pixels. This reduction was necessary to align the CMOS data accurately with the event camera data and to manage computational resources effectively. Figures 2 and 3 reflect these 100×100 segments from higher-resolution movies, providing a proof of concept within the study’s scope. Thus, this context also applies to Figure 3.

The “outdoors walking” dataset records the conditions of a sunny urban environment, including changes in natural light and the active nature of outdoor scenes. Although this dataset is not shown here (refer to Figures 2 and 3 for impressions of similar datasets), it adds value to the previous two by offering a real-world setting with uncontrolled lighting and movement. This helps us to test the effectiveness of our synchronization methods under less predictable conditions. These datasets encompass a broad spectrum of visual scenarios, from the simplicity and control of rotating shapes to the complexity of textured motion and the unpredictability of outdoor settings. This strategic curation of datasets was pivotal in developing our synchronization approaches, ensuring that they are effective and resilient in various real-life applications. The datasets provide a solid foundation for training and evaluating our models, confirming their effectiveness for practical use.

2.3.2. Data Segmentation

We segmented the video data into one-second intervals, each comprising 22 frames, adhering to the dataset's frame rate of 22 FPS. Given the original frame dimensions of 240×180 pixels, we further processed these frames into sub-frames or windows of 100×100 pixels. This segmentation was achieved by sliding the window across the frame in 20-pixel increments, ensuring thorough coverage and variety in the captured data.

2.3.3. Labeling and Variations

In our dataset, each window extracted from the video data underwent a systematic frame shifting process relative to the event image. This process, as elucidated in Figure 4, involved shifting the CMOS camera frame with respect to the event camera frame across a defined range along both the X and Y axes. Figure 4 provides a visual representation of the four primary shift scenarios implemented to produce our labeled dataset. Each illustration demonstrates a CMOS camera frame (dotted line) and an event camera frame (solid line) with specific shifts along the X and Y axes:

1. Top left illustration: Here, the CMOS camera frame is shifted to the left ($\Delta x < 0$) and upwards ($\Delta y < 0$) relative to the event camera frame, depicting a negative shift along both axes.
2. Top right illustration: The CMOS camera frame is shifted to the right ($\Delta x > 0$) and upwards ($\Delta y < 0$) with respect to the event camera frame, illustrating a positive shift along the X axis and a negative shift along the Y axis.
3. Bottom left illustration: This scenario shows the CMOS camera frame shifted to the left ($\Delta x < 0$) and downwards ($\Delta y > 0$) in comparison to the event camera frame, indicating a negative shift along the X axis and a positive shift along the Y axis.
4. Bottom right illustration: The CMOS camera frame is shifted to the right ($\Delta x > 0$) and downwards ($\Delta y > 0$) relative to the event camera frame, representing a positive shift along both the X and Y axes.

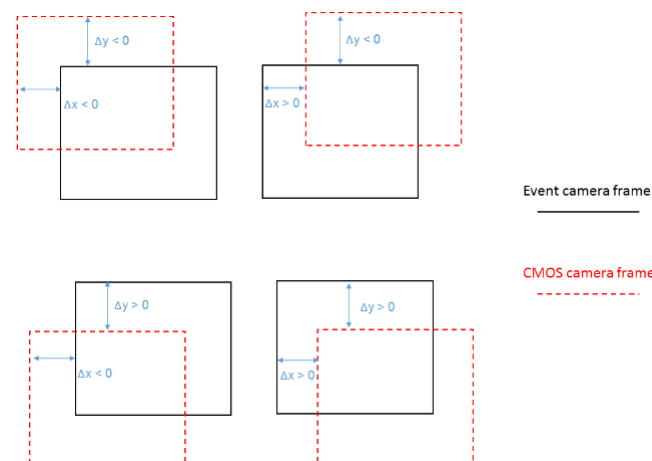


Figure 4. Illustration of frame shifting to produce labeled data, showing the positions of the CMOS camera frame (dotted line) relative to the event camera frame (solid line) for a diverse set of shifts along the X and Y axes.

For each 100×100 window, the frame shifts range from 0 to 70 pixels in any direction, resulting in an extensive range of 140×140 unique shift variations, spanning from -70 to $+70$ pixels for each axis. This systematic shifting creates a precisely labeled dataset that documents each window with known positional shifts. These labeled samples are instrumental for training spatial synchronization models that accurately align CMOS and event camera data.

Each depicted frame shift represents a specific labeled instance in our dataset, capturing a diverse array of relative positions between the CMOS and event camera frames.

Collectively, these labeled instances encompass a full spectrum of possible frame alignments within the stipulated range. This structured labeling approach ensures that our dataset supports the training of a generalized model capable of handling a wide range of real-world synchronization scenarios.

By leveraging the entire collection of films available in the DAVIS240C dataset, we assembled a dataset intended to support the training of a model that generalizes across different visual contexts. This approach aims to enhance the performance and applicability of our synchronization methodology by incorporating a diverse range of visual scenarios.

To evaluate the proposed method, we created a large dataset for both training and testing. Approximately 3 million observations with various movements were selected for the training set, ensuring an equal distribution from all three movies (“shapes rotation,” “boxes rotation,” and “outdoors walking”). This balanced selection helps the model to generalize across different visual scenarios. For the testing phase, we used 600,000 observations, also evenly distributed among the three movies. This dataset allowed us to evaluate the performance of our CNN-DGCNN method across different and dynamic environments.

2.4. Network Architecture for CMOS and Event Camera Calibration

The network architecture detailed in this manuscript features a dual-input system designed specifically for the task of calibrating data from two distinct camera technologies: the conventional CMOS (complementary metal–oxide–semiconductor) camera and the innovative event camera. This setup is essential for effectively processing and integrating the disparate data types that these cameras capture. Figure 4 displays the DGCNN-CNN calibration network architecture, which employs a dual-input strategy to process and synchronize inputs from both CMOS and event cameras efficiently. The diagram details the data flow within the network, showing separate pathways for CMOS video and event camera data that converge to a unified calibration output. This visual representation elucidates the system’s architecture, highlighting how distinct data streams are integrated and processed.

The dynamic graph convolutional neural network (DGCNN) processes event camera data by updating the graph structure as new data are received. This method identifies spatial relationships within the event data, which is necessary for accurate calibration. The DGCNN builds a graph $G = (V, E)$, where V represents the feature vectors of the vertices (events) at layer l and E represents the edges (connections between events). Each event detected by the camera is represented as a vertex in V , and the connections between these events are the edges in E .

The initial feature vector h_i^0 for each vertex i is based on the raw event data, including spatial coordinates (x_i, y_i) , the timestamp (t_i) , and the intensity change (p_i) . The EdgeConv operation, which updates the feature vectors of the vertices, is defined as:

$$h_i^{l+1} = \max_{j:(i,j) \in E} \text{ReLU} \left(h_i^l + \phi(h_j^l - h_i^l) \right) \quad (2)$$

Here, h_i^l is the feature vector of vertex i at layer l ; h_j^l is the feature vector of a neighboring vertex j ; and ϕ is a neural network function, often a multi-layer perceptron, that computes the edge features based on the difference between the feature vectors of neighboring vertices. ReLU (Rectified Linear Unit) is an activation function that introduces non-linearity into the model, allowing it to learn more complex patterns. The max operation aggregates features from neighboring vertices, helping the network understand spatial relationships effectively.

Within each DGCNN layer, the feature vectors of the vertices are updated by considering the features of their neighboring vertices. For each vertex i , a new feature embedding e_i^{l+1} is computed as:

$$e_i^{l+1} = \max_{j:(i,j) \in E} \emptyset(h_i^l, h_j^l - h_i^l) \quad (3)$$

This step involves computing the new feature embedding for each vertex by looking at the differences in the feature vectors of neighboring vertices and applying the function $\emptyset$. After processing through all the DGCNN layers, the final feature vector for each vertex is obtained. The final feature vector h_i^L for each vertex i is represented as:

$$h_i^L = \text{AGGREGATE}(\{e_i^l | l = 1, \dots, L\}) \quad (4)$$

The AGGREGATE function combines the embeddings from all layers l to form the final feature vector h_i^L for each vertex i . On the left side of the illustration presented in Figure 4, the input from a CMOS camera is depicted as a 1 s video clip segmented into 22 frames, each with a resolution of 100×100 pixels. These video data undergo processing through multiple CNN blocks designed to progressively refine the data. Initially starting with 64 channels, the architecture expands to 256 channels, all using a 3×3 kernel size. Each block is engineered to enhance the network's understanding of the video by extracting features at progressively higher levels of abstraction. As the network advances, the depth of the feature maps increases, enabling the capture of increasingly complex patterns and details. This refinement process, which includes convolution, batch normalization, and max-pooling, effectively reduces dimensionality and accentuates key features.

On the right side, the input from the event camera is represented as a point cloud with n -by-3 dimensions. This sparse and asynchronous event data undergo processing through a series of EdgeConv layers, which are structured to progressively increase in complexity from 64 to 512 output channels. The EdgeConv layers are tasked with constructing and updating a dynamic graph that captures spatial relationships within the event data, enabling the network to adaptively learn and integrate event information over time. This is vital for accurate temporal tracking of events captured by the event camera.

The final layer of the DGCNN outputs features that are fused with the CNN-processed CMOS data through a multi-layer perceptron (MLP) to predict the spatial calibration parameters $(\Delta x, \Delta y)$:

$$(\Delta x, \Delta y) = \text{MLP}(h_{\text{cmos}}, h_{\text{DGCNN}}) \quad (5)$$

Here, h_{cmos} and h_{DGCNN} represent the feature vectors from the CNN and DGCNN, respectively. This fusion (as illustrated in Figure 5) enables precise alignment of CMOS and event camera data, ensuring accurate spatial synchronization.

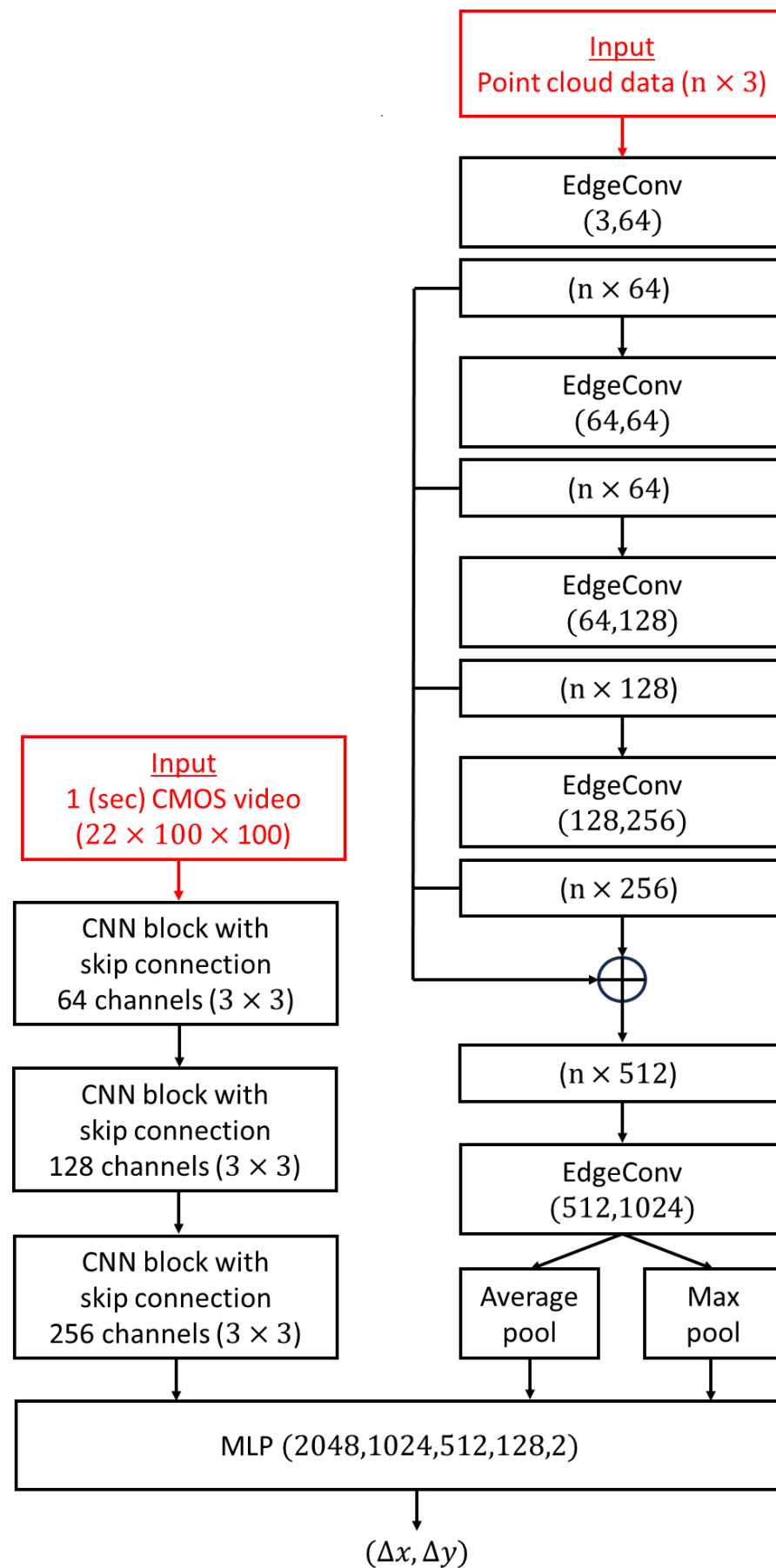


Figure 5. DGCNN-CNN calibration network architecture.

2.4.1. Convolutional Neural Network (CNN) for CMOS Data Processing

The CMOS camera data are processed through a series of convolutional neural network (CNN) blocks, each designed to handle 1 s video segments consisting of 22 frames at a resolution of 100×100 pixels. These blocks include convolutional layers with a 3×3 kernel, followed by batch normalization, which ensures consistent learning by centering and scaling the layer inputs [17,18]. Batch normalization helps to maintain a steady learning process across the network. After normalization, a max-pooling operation with a 2×2 kernel reduces the spatial dimensions of the feature maps, decreasing computational demands and reducing the risk of overfitting.

Each CNN block incorporates a skip connection, as shown in Figure 6. The skip connection provides a direct gradient path, bypassing one or more layers, which prevents vanishing gradient issues in deeper networks. To ensure compatibility with the main CNN output, an additional 2D convolutional layer with a 1×1 kernel adjusts the feature map dimensions within the skip pathway. This element-wise addition merges the bypassed information with the main pathway's output, maintaining data integrity while benefiting from the learned features.

Figure 6 illustrates a single CNN block with a skip connection, detailing its structure. The input initially passes through a 2D convolutional layer, capturing basic spatial features. Batch normalization stabilizes the learning process, and an activation function introduces non-linearity. A max-pooling operation follows to reduce the feature map dimensions and data volume for subsequent layers. The skip pathway contains a 2D convolutional layer to adjust feature map dimensions, aligning them with the main pathway output. This ensures that both immediate and deeper features contribute to the final representation of the video data, encapsulating low-level details and high-level abstractions. By combining these components, the network architecture is optimized for processing CMOS camera data, facilitating accurate calibration when integrated with event camera data.

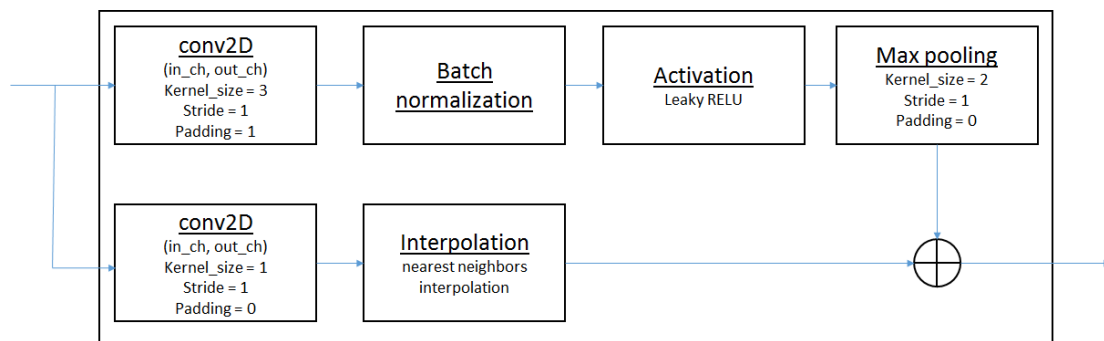


Figure 6. Visualization of a single CNN block with skip connection.

2.4.2. Dynamic Graph CNN (DGCNN) for Event Data Processing

Event camera data differ significantly from traditional camera data due to their unique characteristics. Represented as a point cloud (where each point has dimensions n -by-3), the data are processed using a specialized neural network called a dynamic graph CNN (DGCNN). Unlike standard CNNs, the DGCNN is well-suited for handling the unstructured nature of point cloud data. Inspired by successful architectures used for gesture recognition with event cameras [19], the DGCNN is adept at capturing spatial relationships and patterns within irregular and sparse data distributions [20]. It constructs a graph that dynamically adjusts as new data are received, allowing the network to remain current with the most recent spatial relationships between points. The network architecture gradually increases in complexity, starting with layers configured for simpler feature detection (3×64) and then progressing to layers capable of identifying more intricate

features (512×1024). This step-by-step increase in layer complexity enables the DGCNN to analyze and understand the full depth of event data comprehensively.

2.4.3. Network Fusion and Calibration Output

After extracting features from the standard video using the CNN blocks and from the event data using the DGCNN, these features are combined for calibration in a multi-layer perceptron (MLP). The MLP in our system consists of layers with 2048, 1024, 512, 128, and 2 neurons. This structure processes data from both sources and outputs spatial calibration parameters (Δx , Δy). This step aligns spatial information from the CMOS camera with temporal data from the event camera, creating a detailed representation of motion and space. By merging these data streams, our network supports real-time spatial calibration, potentially enhancing systems that rely on mixed-modality visual information, such as autonomous navigation, robotics, and augmented reality.

2.5. Computational Setup and Training Parameters

To evaluate our CNN-DGCNN model, we used a local machine equipped with a 13th Gen Intel Core i9-13900K processor, 128 GB of RAM, and an Nvidia GeForce RTX 4080 GPU with 16 GB of RAM. These specifications ensured efficient handling of large datasets and complex computations.

Training was conducted using PyTorch version 1.10. We set the number of epochs to 100 and used early stopping based on validation loss to prevent overfitting. The Adam optimizer was chosen for its efficiency in managing large datasets and its adaptive learning rate capabilities, set at an initial rate of 0.001. We used a batch size of 64 to balance memory usage and computational efficiency. Early stopping was implemented with a patience of 10 epochs, meaning training would halt if there was no improvement in validation loss over 10 consecutive epochs. Additionally, a learning rate scheduler, ReduceLROnPlateau, was applied with a factor of 0.1 and patience of 5 epochs to dynamically adjust the learning rate and refine the training process.

During training, we closely monitored the model's performance using validation loss. Specifically, we tracked the difference between the predicted and actual calibration parameters (Δx , Δy). This allowed us to apply early stopping when there was no significant reduction in validation loss, ensuring that the model did not overfit to the training data. By dynamically adjusting the learning rate based on validation performance, we were able to fine-tune the model for optimal accuracy and efficiency.

3. Results

Our study investigates how the spatial distribution of events within an image, described through entropy, impacts the precision of video calibration algorithms. Notably, areas with denser event clusters, which correspond to lower entropy values, improve the calibration's accuracy. This empirical observation is supported by the entropy equation detailed in Equation (2), which calculates the entropy based on the distribution of events:

$$Entropy = - \sum_{x=1}^m \sum_{y=1}^n p(x,y) * \log_2(p(x,y)) \quad (6)$$

where:

m is the number of rows in the input video and the corresponding two-dimensional histogram $p(x,y)$. In this case, m is defined as 100.

n is the number of columns in the input video and the corresponding two-dimensional histogram $p(x,y)$. In this case, n is defined as 100.

$p(x,y)$ is the ratio of the number of events at point (x,y) in the two-dimensional histogram to the total of all events in the histogram.

To evaluate the calibration process precisely, we measure the calibration error using the following distance formula:

$$\delta_{cal} = \sqrt{(\Delta x - \widetilde{\Delta x})^2 + (\Delta y - \widetilde{\Delta y})^2} \quad (7)$$

where:

δ_{cal} represents the calibration error distance;

$\Delta x, \Delta y$ are the labeled calibration shift values per each window;

$\widetilde{\Delta x}, \widetilde{\Delta y}$ are the estimated calibration shift values per each window.

The effect of entropy on calibration precision is captured in Figure 7, which portrays a significant finding from our study: the calibration error distance increases with entropy, yet this relationship exhibits variability depending on the dataset. Thus, this graph serves as an indicator of how different environmental conditions can uniquely impact calibration performance.

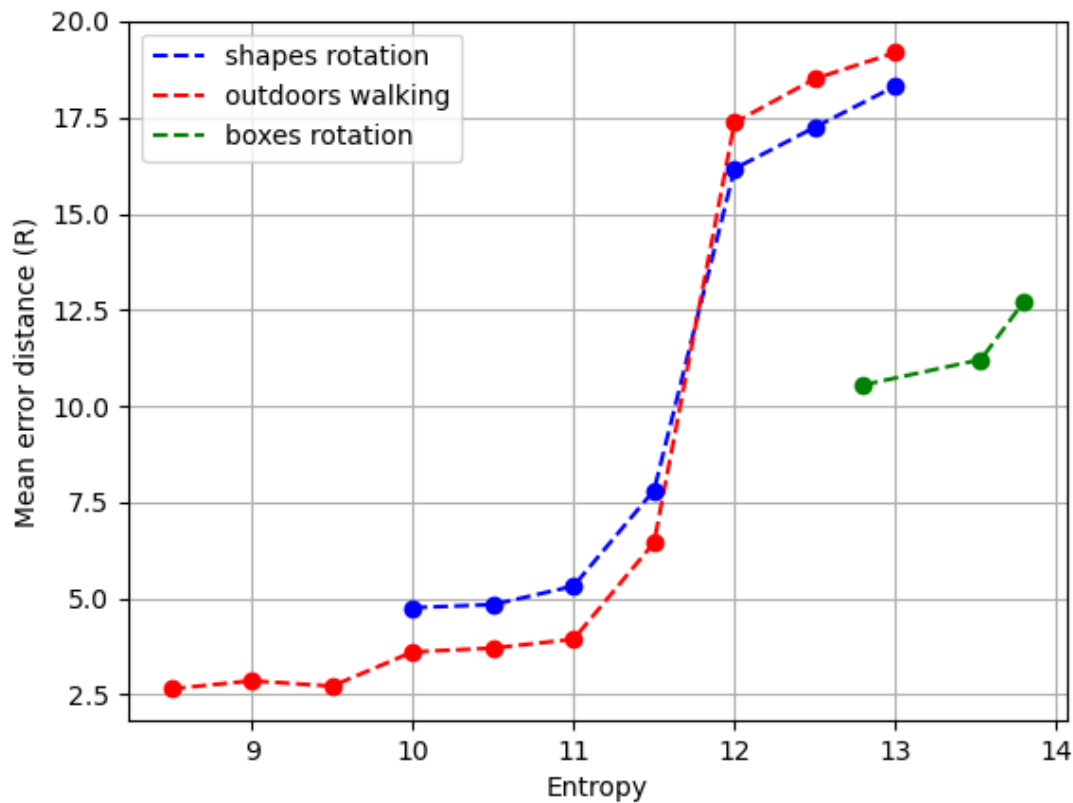


Figure 7. Calibration results as a function of entropy value and dataset.

In “shapes rotation” (represented by the blue dash-dot line), the calibration error remains relatively stable and then shows a gradual increase as the entropy grows. The “outdoors walking” dataset (indicated by the red dash-dot line) starts with a lower error distance at low entropy levels, but as entropy increases, the error distance rises more sharply than in “shapes rotation”. This sharp increase suggests that the model has more difficulty achieving precise calibration when events are uniformly distributed, which is common in outdoor environments.

Most notably, the “boxes rotation” dataset (marked by the green dash-dot line) shows a distinct pattern. The calibration error starts off minimal and increases at a slower rate than the other datasets, despite having the highest entropy. However, once reaching a certain threshold, there is a noticeable jump in the error distance, after which it stabilizes.

This unique trend may be attributed to the specific characteristics of the dataset, such as the presence of high-contrast textures, which could initially assist in the calibration process but become less beneficial as entropy increases beyond a certain point.

These variations in error response to entropy across the datasets emphasize the complexity of calibration tasks. Different scene characteristics, such as the presence of distinct textures in “boxes rotation” or the more dynamic elements in “outdoors walking”, are likely influencing factors in how well the calibration algorithm can perform.

Table 1 complements the graphical data, summarizing the average and variability of entropy and event counts for the datasets. “Shapes rotation” records a mean entropy of 12.69 (SD: 0.51), with an event count average of 95K (SD: 37K). “Outdoors walking” shows a lower average entropy of 12.11 (SD: 0.92) and a higher average event count of 145K (SD: 67K). “Boxes rotation” stands out, with the highest average entropy of 13.23 (SD: 0.05) and the largest average number of events at 712K (SD: 334K).

Table 1. Mean and SD of entropy and events count by dataset.

Dataset/Metric	Mean Entropy	STD Entropy	Mean Number of Events	STD Number of Events
Shapes rotation	12.69	0.51	95K	37K
Outdoors walking	12.11	0.92	145K	67K
Boxes rotation	13.23	0.05	712K	334K

When comparing the data from Table 1 with the calibration error trends depicted in Figure 7, we can see a clear connection between entropy and calibration accuracy. This observation reinforces the importance of considering the distribution of events when designing and refining calibration methods for video analytics.

4. Discussion

Our study reveals that the complex interaction between entropy, event density, and calibration accuracy significantly impacts the outcomes of video data processing. Regions within a video frame with dense event distributions provide strong reference points, enhancing calibration accuracy. This is particularly evident when event cameras capture dense clusters of changes, which serve as precise anchor points for the calibration process. These “high information” zones not only increase the overall entropy, but also directly enhance the calibration effectiveness.

The effectiveness of dynamic graph CNNs (DGCNNs) in handling event data has improved real-time spatial synchronization and fostered the development of algorithms that adaptively respond to variations in event density and distribution, enhancing synchronization accuracy across diverse environments [3,21]. Our empirical tests confirm that calibration quality is heavily dependent on both the volume and spatial concentration of events, with denser distributions yielding better outcomes. These insights underscore the value of integrating CMOS and event camera technologies for applications demanding high precision and responsiveness, and they pave the way for future research into optimizing mixed-modality visual systems [3,21].

These calibration advancements extend into environmental monitoring, where accurate, real-time calibration of sensor data across various modalities is critical. For instance, edge-detection algorithms calibrated for specific underwater conditions have enabled more precise monitoring of coral ecosystems, contributing to conservation efforts [22]. Similarly, precise calibration in atmospheric monitoring using techniques like Fabry–Perot interferometers, which measure light wavelengths to detect CO₂ levels, plays a vital role in efforts to mitigate the greenhouse effect [23].

Furthermore, the scope of these advancements has broad implications for industries such as autonomous vehicle navigation, robotic surgery, and augmented reality, where enhanced calibration techniques enable more precise and real-time interactions between event and CMOS cameras [24,25]. The integration of these technologies with emerging IoT

devices and smart city infrastructure has also opened up transformative opportunities for urban development. Enhanced calibration techniques are essential for effective traffic management and interactive public safety solutions, ensuring accurate, real-time data processing. Techniques like the attention-based encoder–decoder network using atrous convolution are being utilized to effectively calibrate data from urban environments, aiding in urban water management [26].

In the Internet of Vehicles (IoV), innovative traffic management solutions, such as the dynamic ant colony optimization algorithm with a look-ahead mechanism, are revolutionizing urban traffic management by leveraging real-time data from wireless sensor networks. This enhances traffic signal control efficiency and reduces carbon emissions, highlighting the essential role of advanced calibration in maintaining data accuracy and reliability in these systems [27].

However, our study has limitations due to its reliance on specific datasets and conditions. Future research should extend these findings across a wider range of scenarios, including those with challenging lighting and fast-moving subjects. Additionally, improving the computational efficiency of our proposed architecture will be important to ensure its utility in real-world, time-sensitive applications [28].

Looking ahead, we suggest exploring different deep learning architectures to improve the calibration speed and efficiency [28]. Addressing non-linear distortions from camera lenses is also crucial [29,30]. Testing these improvements in practical settings will help to determine their real-world effectiveness [31,32]. Integrating this technology into wearable devices could expand its uses. Wearables with advanced calibration capabilities could enhance user experiences in fitness tracking, health monitoring for prosthetic adjustments, and interactive environments, blending digital and physical interactions [33].

In conclusion, our research on improving calibration methods is key to advancing 3-D reconstruction capabilities, as demonstrated by our study and supported by related literature on photometric stereo [34]. These improvements demonstrate the reliability of our methods and provide greater insight into how light interacts with surfaces, thereby enhancing 3-D imaging techniques. In future research, it will be important to refine visual systems in order to better adapt to the complexities of modern environments.

Author Contributions: Conceptualization, D.M., I.L. and I.Z.; Methodology, D.M., I.L. and I.Z.; Software, D.M.; Validation, D.M., I.L. and I.Z.; Formal analysis, D.M., I.L. and I.Z.; Investigation, D.M.; Data curation, D.M.; Writing—original draft, D.M., I.L. and I.Z.; Writing—review & editing, D.M., I.L. and I.Z.; Visualization, D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ben-Ezra, M.; Nayar, S.K. Motion deblurring using hybrid imaging. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Madison, WI, USA, 18–20 June 2003.
2. Rebecq, H.; Ranftl, R.; Koltun, V.; Scaramuzza, D. High speed and high dynamic range video with an event camera. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *43*, 1964–1980.
3. Gallego, G.; Delbrück, T.; Orchard, G.; Bartolozzi, C.; Tabatabaei, B.; Censi, A.; Leutenegger, S.; Davison, A.J.; Conradt, J.; Daniilidis, K.; et al. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *44*, 154–180.
4. Gehrig, D.; Loquercio, A.; Derpanis, K.G.; Scaramuzza, D. End-to-end learning of representations for asynchronous event-based data. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 5633–5643.
5. Li, X.; Yu, S.; Lei, Y.; Li, N.; Yang, B. Intelligent machinery fault diagnosis with event-based camera. *IEEE Trans. Ind. Inform.* **2023**, *20*, 380–389.

6. Bardow, P.; Davison, A.J.; Leutenegger, S. Simultaneous optical flow and intensity estimation from an event camera. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 884–892.
7. Mueggler, E. Event-Based Vision for High-Speed Robotics. Ph.D. Thesis, University of Zurich, Zürich, Switzerland, 2017.
8. Glover, A.; Vasco, V.; Bartolozzi, C. A controlled-delay event camera framework for on-line robotics. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018.
9. Iaboni, C.; Patel, H.; Lobo, D.; Choi, J.-W.; Abichandani, P. Event camera based real-time detection and tracking of indoor ground robots. *IEEE Access* **2021**, *9*, 166588–166602.
10. Lee, M.S.; Jung, J.H.; Kim, Y.J.; Park, C.G. Event-and Frame-based Visual-Inertial Odometry with Adaptive Filtering based on 8-DOF Warping Uncertainty. *IEEE Robot. Autom. Lett.* **2023**, *9*, 1003–1010.
11. Muglikar, M.; Gehrig, M.; Gehrig, D.; Scaramuzza, D. How to calibrate your event camera. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 1403–1409.
12. Rebecq, H.; Ranftl, R.; Koltun, V.; Scaramuzza, D. Events-to-video: Bringing modern computer vision to event cameras. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 3857–3866.
13. Kumar, P.P.; Kant, K.; Pal, A. Non-intrusive driving behavior characterization from road-side cameras. *IEEE Internet Things J.* **2023**, *11*, 502–509.
14. Panetsos, F.; Karras, G.C.; Kyriakopoulos, K.J. Aerial Transportation of Cable-Suspended Loads With an Event Camera. *IEEE Robot. Autom. Lett.* **2023**, *9*, 231–238.
15. Reinbacher, C.; Munda, G.; Pock, T. Real-time panoramic tracking for event cameras. In Proceedings of the IEEE International Conference on Computational Photography (ICCP), Stanford, CA, USA, 12–14 May 2017; pp. 1–9.
16. Gallego, G.; Scaramuzza, D. Accurate angular velocity estimation with an event camera. *IEEE Robot. Autom. Lett.* **2017**, *2*, 632–639.
17. Garbin, C.; Zhu, X.; Marques, O. Dropout vs. batch normalization: An empirical study of their impact to deep learning. *Multimed. Tools Appl.* **2020**, *79*, 12777–12815.
18. Li, Y.; Wang, N.; Shi, J.; Liu, J.; Hou, X. Revisiting batch normalization for practical domain adaptation. *arXiv* **2016**, arXiv:1603.04779.
19. Chen, J.; Meng, J.; Wang, X.; Yuan, J. Dynamic graph cnn for event-camera based gesture recognition. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 12–14 October 2020; pp. 1–5.
20. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.* **2019**, *38*, 1–12.
21. Schindler, G.; Dellaert, F. Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 27 June–2 July 2004.
22. Awalludin, E.A.; Yussof, W.N.J.H.W.; Bachok, Z.; Aminudin, M.A.F.; Din, M.S.C.; Hitam, M.S. Monitoring climate change effects on coral reefs using edge-based image segmentation. *Int. J. Electr. Comput. Eng.* **2024**, *14*, 398–408.
23. Song, W.; Chuang, Z.; Su-Wen, L.; Fu-Sheng, M. Method of measuring atmospheric CO₂ based on Fabry-Perot interferometer. *Acta Phys. Sin.* **2024**, *73*, 020702.
24. El Shair, Z.A.; Hassani, A.; Rawashdeh, S.A. CSTR: A Compact Spatio-Temporal Representation for Event-Based Vision. *IEEE Access* **2023**, *11*, 102899–102916.
25. AlRemeithi, H.; Zayer, F.; Dias, J.; Khonji, M. Event Vision for Autonomous Off-Road Navigation. In *Artificial Intelligence for Robotics and Autonomous Systems Applications*; Springer: Cham, Switzerland, 2023; pp. 239–269.
26. Song, Y.; Rui, X.; Li, J. AEDNet: An attention-based encoder-decoder network for urban water extraction from high spatial resolution remote sensing images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2023**, *17*, 1286–1298.
27. Liao, S.; Wu, Y.; Ma, K.; Niu, Y. Ant Colony Optimization with Look-ahead Mechanism for Dynamic Traffic Signal Control of IoV Systems. *IEEE Internet Things J.* **2023**, *11*, 366–377.
28. Menghani, G. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Comput. Surv.* **2023**, *55*, 1–37.
29. Zhang, Y.-J. Camera calibration. In *3-D Computer Vision: Principles, Algorithms and Applications*; Springer: Singapore, 2023; pp. 37–65.
30. Lu, Z.; Cai, L. Camera calibration method with focus-related intrinsic parameters based on the thin-lens model. *Opt. Express* **2020**, *28*, 20858–20878.
31. Chen, G.; Cao, H.; Conradt, J.; Tang, H.; Rohrbein, F.; Knoll, A. Event-based neuromorphic vision for autonomous driving: A paradigm shift for bio-inspired visual sensing and perception. *IEEE Signal Process. Mag.* **2020**, *37*, 34–49.
32. Maqueda, A.I.; Loquercio, A.; Gallego, G.; García, N.; Scaramuzza, D. Event-based vision meets deep learning on steering prediction for self-driving cars. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 5419–5427.

33. Ahmadizadeh, C.; Galli, V.; Luder, V.; Menon, C. Inductive Textile Sensing for Movement Monitoring. *IEEE Sens. Lett.* **2023**, *8*, 5500104.
34. Yang, Y.; Rigall, E.; Fan, H.; Dong, J. Point Light Measurement and Calibration for Photometric Stereo. *IEEE Trans. Instrum. Meas.* **2023**, *73*, 5001011. <https://doi.org/10.1109/TIM.2023.3336457>.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.