processes

MDPI

Article

Contrast Maximization-Based Feature Tracking for Visual Odometry with an Event Camera

Xiang Gao, Hanjun Xue and Xinghua Liu *

School of Electrical Engineering, Xi'an University of Technology, Xi'an 710048, China
* Correspondence: liuxh@xaut.edu.cn

Abstract: As a new type of vision sensor, the dynamic and active-pixel vision sensor (DAVIS) outputs image intensity and asynchronous event streams in the same pixel array. We present a novel visual odometry algorithm based on the DAVIS in this paper. The Harris detector and the Canny detector are utilized to extract an initialized tracking template from the image sequence. The spatio-temporal window is selected by determining the life cycle of the asynchronous event streams. The alignment on timestamps is achieved by tracking the motion relationship between the template and events within the window. A contrast maximization algorithm is adopted for the estimation of the optical flow. The IMU data are used to calibrate the position of the templates during the update process that is exploited to estimate camera trajectories via the ICP algorithm. In the end, the proposed visual odometry algorithm is evaluated in several public object tracking scenarios and compared with several other algorithms. The tracking results show that our visual odometry algorithm can achieve better accuracy and lower latency tracking trajectory than other methods.

Keywords: event camera; visual odometry; tracking template; contrast maximization

Citation: Gao, X.; Xue, H.; Liu, X. Contrast Maximization-Based Feature Tracking for Visual Odometry with an Event Camera. *Processes* **2022**, *10*, 2081. https://doi.org/10.3390/pr10102081

Academic Editor: Xiong Luo

Received: 7 August 2022
Accepted: 10 October 2022
Published: 14 October 2022

1. Introduction

Visual odometry has played an important role in robot navigation [1], intelligent transportation [2], and intelligent inspection [3]. Although several decades of active research have led to a certain level of maturity, we still face challenges in scenes with high dynamics, low texture, or harsh lighting conditions [4]. Conventional visual odometry generally acquires images from frame-based cameras and estimates camera motion from several adjacent images [5]. However, for objects with a high dynamic range or high-speed motion, frame-based cameras cannot obtain clear images. Therefore, it is difficult for a frame-based camera to extract the feature points of the image and estimate the camera pose in this case. Furthermore, during the blind period between frames, the frame-based camera does not capture the precise motion of the feature information. Frame-based cameras capture redundant information in static scenes, resulting in a waste of storage and computational resources [6].

Bio-inspired event cameras, such as the dynamic vision sensor (DVS) [7], overcome the above-mentioned limitations of frame-based cameras [8]. As a new type of bio-heuristic vision sensor, the event camera has a completely different mode from the conventional camera [9], and event cameras only output the change in pixel-level brightness. For each pixel, when the intensity change reaches a certain threshold, an event is triggered; the event carries information on pixel coordinates, timestamps, and polarity [10]. Compared with the low frame rate, large delay, and insufficient dynamic response range in conventional visual cameras, an event camera has the characteristics of fast response speed, high dynamic range, and low power consumption [11]. Although traditional frame-based tracking algorithms have been vigorously developed, asynchronous event streams cannot be directly handled by current frame-based pose estimation methods [12]. As a result, event-based tracking algorithms are required. The dynamic and active-pixel vision sensor (DAVIS)

is a sensor that combines a frame-based camera with an asynchronous event-based camera in the same pixel array. Based on the assumption that events are mainly triggered by high gradient edges in the image, the optimal motion parameters for events can be computed by maximizing contrast in the Image of Warped Events (IWE) [13]. Various reward functions for evaluating contrast were proposed and analyzed in recent work by Gallego et al. [14]. Contrast maximization algorithms for events have also been successfully used to solve various problems of event cameras, such as optical flow estimation [15,16], motion segmentation [17,18], 3D reconstruction [19], and motion estimation [20]. Since event streams cannot provide absolute brightness values and synchronized output with image frames, a contrast maximization algorithm is utilized to resolve event and image frame data associations in this paper, while the event-based feature tracking can be further simplified.

In this paper, we present a novel visual odometry algorithm based on template edges using a DAVIS camera. In our tracking approach, we leverage a combination of event- and frame-based camera measurements. The tracked initial tracking template is extracted by the feature detector in the image, and the spatio-temporal windows are selected by determining the life cycle of the asynchronous event streams. The calibrated tracking templates are utilized to calculate the event camera trajectory. The main contributions are summarized as follows:

1. Compared with traditional event-by-event tracking methods [20–22], a new tracking mechanism is presented to resolve data associations. A contrast maximization method is adopted to calculate the displacement parameters of the events, and the IMU data are used to calibrate the rotation parameters of the events, which greatly enhances the calculation speed and accuracy of the event stream.
2. Since the ICP algorithm is highly dependent on the depth of the scene, a robust Beta-Gaussian distribution depth filter is presented to obtain a more accurate depth of the tracking template than depth estimation with only triangulation [23,24].
3. We successfully apply our method to evaluation experiments in several different scenarios on public event camera datasets. Compared with the visual odometry algorithm [7,9], the proposed algorithm can achieve better performance and obtain a lower-latency camera trajectory.

The rest of this article is organized as follows. The related work on event-based camera visual odometry is presented in Section 2. The realization steps of fusion events and image frame tracking are described in Section 3. The effectiveness of the algorithm is verified on several DAVIS datasets in Section 4. Finally, conclusions are given in Section 5.

2. Related Work

Feature detection and tracking methods for frame-based cameras are well established. However, these frame-based methods cannot track the blind time between two adjacent frames, and frame-based cameras still capture information in all pixel arrays, even in scenes without motion [17]. In contrast, an event-based camera only acquires information about the areas of the scene where the intensity value has changed, and it fills the blind spots between adjacent frames with a higher asynchronous response rate. The advantages of event-based cameras are better suited for applications such as driverless vehicles and motion tracking [22,25].

Event-based visual odometry can be divided into two main methods: one is to use traditional methods to accumulate event information for tracking, and the other is to track directly based on asynchronous event streams. Gallego et al. [26] presented the pose tracking of a 6-DoF event camera from an existing photometric depth map (intensity + depth information). Stoffregen et al. introduced event accumulation frames with edge contrast maximization for motion segmentation [27]. Alzugaray and Chli [28] proposed a purely event-based corner detector and a new corner tracker, proving that it is possible to directly detect and track corners in event streams. Although there is a great deal of research devoted to event-based feature detection, very little work has been done to consider the

problem of event tracking. Some approaches for localization and mapping with event cameras have several similarities with our method. Gallego et al. [29] detected independent moving objects by tracking corners detected in event images integrated over short time windows. Mueggler et al. [30] proposed a DVS-based ego-motion estimation method that uses a continuous-time framework to directly integrate the information transmitted by the camera. The pose trajectory of the DVS is estimated based on the observed events. Kim et al. [31] estimated 6-DoF camera motion, log intensity gradient, and inverse depth and used three decoupled probabilistic filters in real time.

Recently, Mueggler et al. [32] extended a frame and event tracker for DAVIS cameras and integrated events from event cameras for high-speed object tracking. In contrast to the above-mentioned approach, we consider the connection between event stream and images in our modeling process, and higher-rate trajectory tracking is provided.

3. Main Methods

This paper proposes a visual odometry algorithm based on DAVIS. The main methods are divided into two parts: feature detection and feature tracking. As shown in Figure 1, the entire process was divided into six steps. We detect features in the image sequence. The contrast maximization algorithm is then utilized to match the event stream in the corresponding spatio-temporal windows and calculate the optical flow of the event. The estimated motion parameters and the IMU measurement are exploited to calibrate the tracking template. We then detect the depth values of the tracking template using a depth filter. Finally, the 6-DoF event camera pose can be estimated by the ICP algorithm.

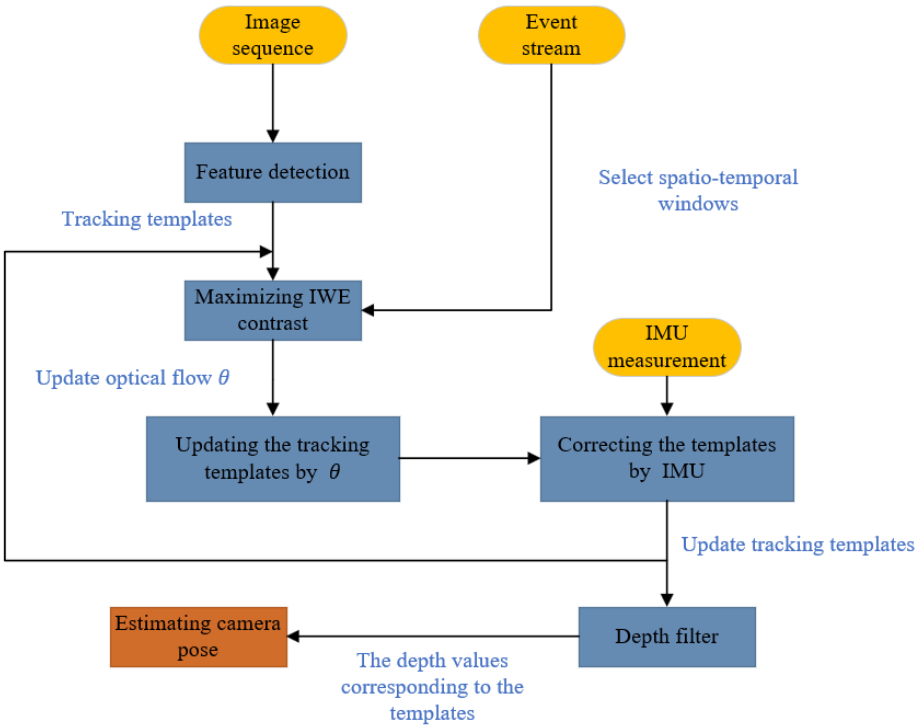
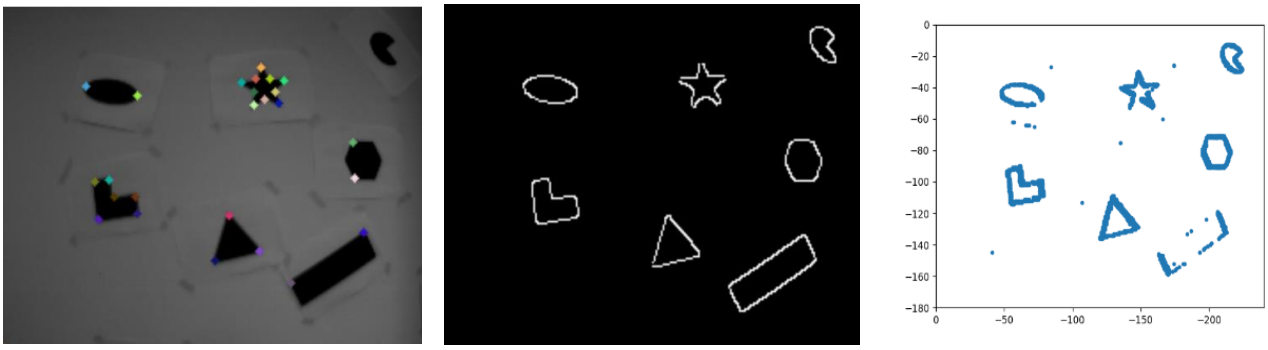


Figure 1. An overview of the visual odometry method. θ is the optical flow of the event in the corresponding spatio-temporal window.

3.1. Feature Detection

Since events in edge regions of the scene are triggered more frequently than those in low-texture regions, we devise suitable features for tracking. As shown in Figure 2, the method first extracts feature points and edge maps from image frames by the Harris detector [9] and the Canny detector [20] in the feature detection stage. Then, it selects the edge in the specified area around each feature point as the template edge of the feature point. All regions are square in the same size, which is a tunable parameter. Our method

does not need to provide frames at a constant rate, as they are only used to initialize features. Keyframes can be added to replace missing features.



(**a**) (**b**) (**c**)

Figure 2. Detection of the tracking template. Using the feature points image in (**a**) and the edge points image in (**b**), we can obtain the tracking template. As shown in (**c**), we visualize the events of the first spatio-temporal window.

3.2. Feature Tracking

Assuming that feature point x is detected from the image frame at time t_0, the motion of feature point x can be described as follows:

$$x(t) = x(t_0) + \int_{t_0}^{t} \dot{x}(s)ds \tag{1}$$

where $\dot{x}(s)$ represents the position differential of the feature point x at time s.

A set of events is selected from the event stream, the time of feature point x is selected as the spatio-temporal window corresponding to initial time t_0, and W is the set of events in the spatio-temporal window.

$$W = \{e_i | t_0 < t_{e_i} < t_1\}_{i=1}^{n} \tag{2}$$

Here, e_i represents the $i-th$ event in the spatio-temporal window, and t_{e_i} represents the time when event e_i occurs. n represents the number of events. Since $[t_0, t_1]$ is the first sub-time interval, the value of t_1 can be calculated by setting the size of the spatio-temporal window. For example, let W contain 10,000 events, that is, $n = 10{,}000$.

3.2.1. Choice of Spatio-Temporal Windows

In order to achieve asynchrony of the feature point tracking method, the size of the sub-time interval is determined by the method during real-time operation. The faster the scene moves, the smaller the sub-time interval and the faster the update frequency of feature points.

The specific calculation process is as follows. After the optical flow of all feature points in this iteration is obtained, the size of the next sub-time interval is calculated from the optical flow. We define x_i to represent the $i-th$ feature point, $i = \{1, \cdots, m\}$, and the number of feature points is m. $\boldsymbol{\theta}_i^n$ is the optical flow of feature point x_i in the $n-th$ sub-time interval $[t_{n-1}, t_n]$. Given the optical flow of all feature points $\{\boldsymbol{\theta}_i^n\}_{i=1}^{m}$ in the sub-time interval $[t_{n-1}, t_n]$, the next sub-time interval $[t_n, t_{n+1}]$ can be calculated:

$$\begin{cases} \boldsymbol{\theta}_{average}^n = (\sum_{i=1}^{m} \boldsymbol{\theta}_i^n)/m \\ t_{n+1} = t_n + 3/\Theta_{average}^n \end{cases} \tag{3}$$

where the unit of the number 3 is pixels, and $\Theta_{average}^n$ represents the average optical flow of all feature points in the sub-time interval $[t_{n-1}, t_n]$. The t_{n+1} value is calculated through

the formula, and the time required for the feature points in the previous time interval to move by 3 pixels on average is used as the estimated value of the current interval $[t_n, t_{n+1}]$.

3.2.2. Maximizing IWE Contrast

After the spatio-temporal window is determined by corresponding to the feature point, we use the contrast maximization algorithm to match the event set W around the feature point x with the template point set. It is assumed that all template points have the same optical flow as feature point x (the optical flow θ of all pixels in the region is the same), and the optical flow of feature point x is constant in the sub-time interval $[t_0, t_1]$. Let the optical flow θ of the feature point x in the time interval $[t_0, t_1]$ be defined as v. For event e_i in W, as shown in Figure 3, Image of Warped Events (IWE) is used to calculate its position X'_k at time t_0. The formula is as follows:

$$X'_k = \begin{bmatrix} x'_k \\ y'_k \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \bullet \left(t_0 - t_{ref} \right) \tag{4}$$

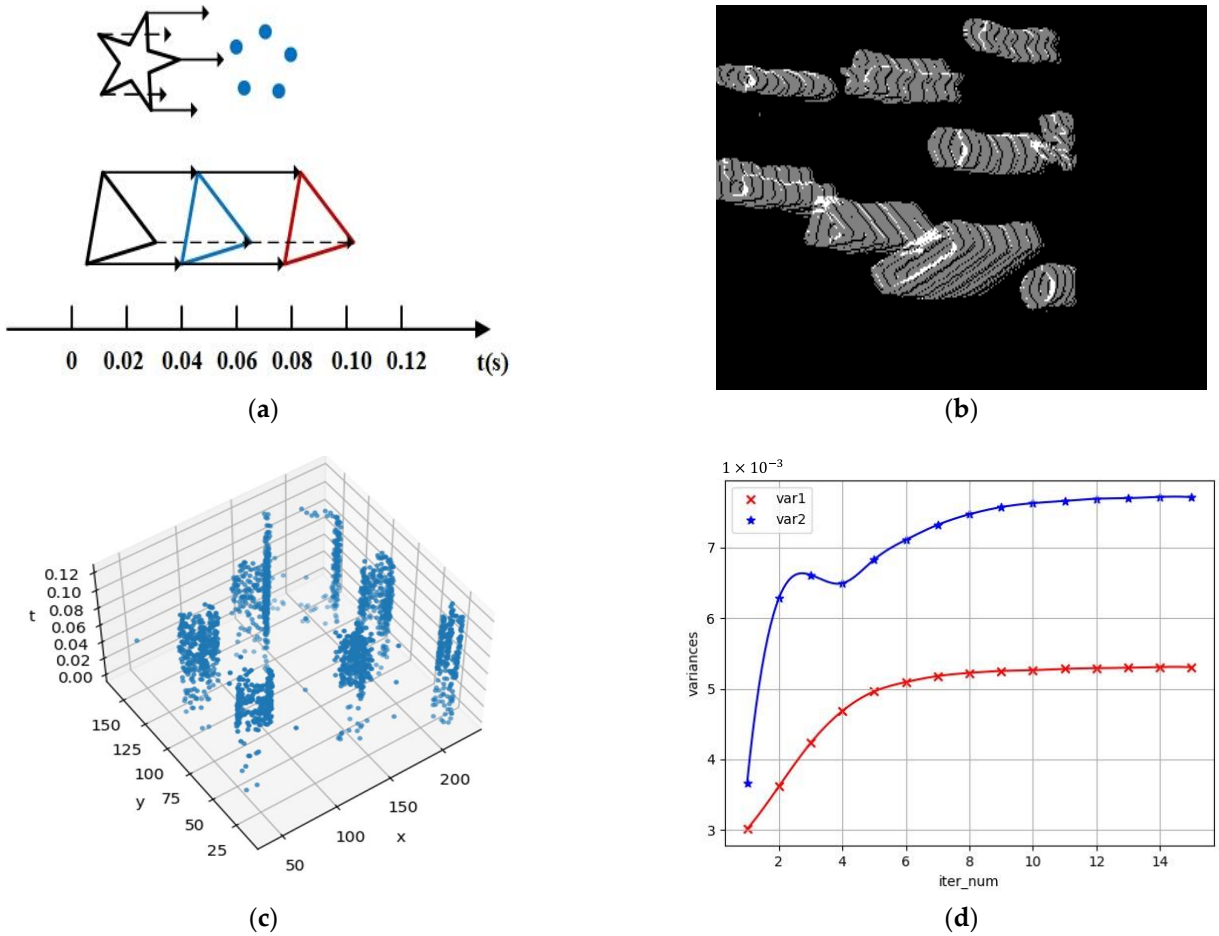


Figure 3. Updating the tracking template. Figure (**a**) is the update process of feature points and template points in a fixed time interval, while Figure (**b**) is an optical flow visualization of the tracking template. As shown in Figure (**c**), the events obtain the optical flow in the first spatio-temporal window. Figure (**d**) presents variance convergence curves. Variance 1 and Variance 2 represent the variance change in IWE after events are warped along with different optical flows.

The weighted IWE is defined as:

$$I_j(x) = \sum_{k=1}^{N_e} P_{kj} \delta(X - X'_{kj}) \tag{5}$$

where X'_{kj} is the position of the $k-th$ event after it is distorted along the $j-th$ optical flow θ, N_e represents the number of events, δ represents the Dirac function, P_{kj} represents the probability that the $k-th$ event belongs to the $j-th$ optical flow, and $I_j(x)$ represents the IWE corresponding to the $j-th$ optical flow.

Events are aligned by image contrast, which is defined by a sharpness/dispersion metric, such as the variance:

$$Var(I_j) = \int_{\Omega} \left(I_j(x) - \mu_j\right)^2 dx \tag{6}$$

where Ω is the image plane and μ_j is the mean of the $j-th$ Image of Warped Event.

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mu \nabla_{\theta} \left(\sum_{j=1}^{N_l} Var(I_j) \right) \tag{7}$$

where, μ represents the step size, and N_l represents the number of clusters.

3.2.3. Template Edge Update

After the optical flow of the feature points is obtained, the feature points and template edges are updated. However, when the camera is rotated, the template edges and template points move at significantly different speeds, and the points farther from the center of rotation move faster. Therefore, if the template points are updated by only using optical flow, the position of feature points will quickly deviate from the true value due to the existence of rotation factors. The process of updating the stencil edge is divided into two steps. First, the position of the stencil edge is updated by using the optical flow, and then the position of the stencil edge is corrected by using the IMU data.

The optical flow θ is used to update the position of feature point x and the corresponding position of the template edge x_j. Assuming that the optical flow of the feature point is constant in the sub-time interval $[t_0, t_1]$, the optical flow θ is then utilized to update the position of feature point x; the formula is as follows:

$$x(t_1) = x(t_0) + \boldsymbol{\theta} \cdot \left(t_j - t_0\right) \tag{8}$$

$x(t_0)$ is the position of the initially extracted feature point, $x\left(t_j\right)$ is the position of the updated feature point, and t_j is the time of the tracking template in the sub-time interval $[t_0, t_1]$.

In order to eliminate the influence of rotation on the template edge update, we introduce IMU data to correct the template edge position. As a result, the corrected position is closer to the true value of the template edge position. The relative position of the template points relative to the feature point is calculated at time t_j, and then the IMU data are used to correct the relative position. For the template point x_j, we define its relative position:

$$x_j^{relative} = x_j(t) - x(t) \tag{9}$$

Since the data in the IMU coordinate system and the tracking template from the camera coordinate system are not in the same system, we need to transform the data. The IMU measurements contain the accelerometer measures and gyroscope measures for each axis, but what we need during the update process is the rotation matrix between the templates. So, we convert linear acceleration and angular velocity into Euler angles and then convert the Euler angles into a rotation matrix. The accelerometer can calculate the roll angle α_{acc} and pitch angle β_{acc} at the time of rest. The gyroscope is the angular velocity integral in the

time interval and can calculate three angles: the roll angle α_{gyro}, pitch angle β_{gyro}, and yaw angle γ_{gyro}. The Euler angle fusion formula can be expressed as:

$$\begin{cases} \alpha = \alpha_{gyro} + (\alpha_{acc} - \alpha_{gyro}) \cdot k \\ \beta = \beta_{gyro} + (\beta_{acc} - \beta_{gyro}) \cdot k \\ \gamma = \gamma_{gyro} \end{cases} \tag{10}$$

where α, β, and γ are obtained after the complementary fusion of the accelerometer and gyroscope pose; k represents the proportional coefficient, which needs to be adjusted according to the actual situation, such as by choosing 0.3. The conversion of Euler angles to a rotation matrix is expressed as:

$$\boldsymbol{R}_{wi} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{11}$$

where $\boldsymbol{R}_{wi}$ is the rotation matrix in the IMU coordinate system.

Then, we solve the transformation of the rotation matrix between the camera coordinate system and the IMU coordinate system. The transformation relation is denoted as:

$$\boldsymbol{R}_{wc} = \boldsymbol{R}_{wi} \cdot \boldsymbol{R}_{ic} \tag{12}$$

where $\boldsymbol{R}_{wc}$ is the rotation matrix in the camera coordinate system, and $\boldsymbol{R}_{ic}$ represents the transformation matrix between the IMU and camera coordinate systems during camera calibration. The IJRR dataset takes the first frame of the camera as the world coordinate system. We assume that the pose transformation of the camera is linear. The rotation matrix of the tracking template can be obtained by linear interpolation of the time scale. The process is represented as follows:

$$\boldsymbol{R}_j = inv\left(\boldsymbol{R}_{wc}^{t_0}\right) \cdot \boldsymbol{R}_{wc}^{t_1} \cdot \frac{t_j - t_1}{t_0 - t_1} \tag{13}$$

where $\boldsymbol{R}_{wc}^{t_0}$ is the rotation matrix in the camera coordinate system at time t_0. $\boldsymbol{R}_{wc}^{t_1}$ is the rotation matrix in the camera coordinate system at time t_1, and it is calculated from the rotation matrix $\boldsymbol{R}_{wi}^{t_1}$ in the IMU coordinate system at time t_1 and the transformation matrix $\boldsymbol{R}_{ic}$ between the camera and IMU; $\boldsymbol{R}_j$ is the interpolated rotation matrix between t_0 and t_j.

The symbols X_j and X respectively represent the 3D coordinates corresponding to the template point x_j and the feature point x in the camera coordinate system at time t_0.

At time t_0:

$$\begin{cases} s_{x_j}^0 x_j(t_0) = KX_j \\ s_x^0 x(t_0) = KX \end{cases} \tag{14}$$

At time t_j:

$$\begin{cases} s_{x_j}^1 x_j(t_j) = K(\boldsymbol{R}_j X_j + \boldsymbol{t}) \\ s_x^1 x(t_j) = K(\boldsymbol{R}_j X + \boldsymbol{t}) \end{cases} \tag{15}$$

where $\boldsymbol{t}$ is the translation vector between t_0 and t_j, $\boldsymbol{R}_j$ is the interpolated rotation matrix between t_0 and t_j, and K represents the internal parameter matrix of the camera.

Substituting the above formula into (9) and normalizing the 3D coordinates, we obtain:

$$x_j^{relative}(t_j) = Nor\left(K\boldsymbol{R}_j K^{-1} x_j(t_0)\right) - Nor\left(K\boldsymbol{R}_j K^{-1} x(t_0)\right) \tag{16}$$

At this time, the position of x_j at time t_j is obtained by adding the position of the feature point x and the relative position of $x_j^{relative}$ at time t_j:

$$x_j(t_j) = x_j^{relative}(t_j) + x(t_j) \tag{17}$$

Finally, the updating of all template points can be completed through the formula.

3.2.4. Depth Estimation

After the corresponding position of the edge map of each frame is obtained, the 3D position coordinates of the edge in the space can be recovered through the matching relationship of the pixels on the edge between different edge maps. The triangulation method is used, and the coordinates of the 3D point in space are recovered by using the pixel positions of the 3D points observed from different viewing angles. The same 3D point can be observed in multiple frames, and the corresponding spatial 3D point coordinates can be calculated for any two frames. The strategy of our method is using the properties of the Gaussian distribution to fuse the current observations, assuming that the depth values follow a Gaussian distribution. In the final estimation of the same 3D point, the strategy is based on a uniform Gaussian mixture distribution.

First, a triangulation method is used to recover the depth of pixels in the edge map. Assuming that a certain pixel p_0 of the edge graph and pixel point p_1 of the edge graph are a pair of matching points, and the two pixels correspond to the same 3D point **P** in the space, the formula holds as follows:

$$s_0 K^{-1} p_0 = s_1 \boldsymbol{R} K^{-1} p_1 + \boldsymbol{t} \tag{18}$$

where $\boldsymbol{R}$ and $\boldsymbol{t}$ are the rotation matrix and translation vector of the event camera from the edge graph to the edge graph, respectively. s_0 and s_1 represent the depths of the 3D point **P** in the coordinate system of the event camera at times t_i and t_j, respectively. Equation (18) is further simplified to obtain:

$$s_1 = \frac{-(K^{-1} p_0)^{\wedge} \boldsymbol{t}}{(K^{-1} p_0)^{\wedge} \boldsymbol{R} K^{-1} p_1} \tag{19}$$

Then, a depth value y_i can be calculated from the corresponding points in the two images. The distribution of y can be jointly represented by a Gaussian distribution and a uniform distribution:

$$p(y_i | \hat{y}, \pi) = \pi N\left(y_i \middle| \hat{y}, \tau_i^2\right) + (1 - \pi) U(y_i | y_{min}, y_{max}) \tag{20}$$

where $N\left(y_i | \hat{y}, \tau_i^2\right)$ is a Gaussian distribution centered on the true value $\hat{y}$, and τ_i^2 is its variance. π is the probability of estimating the correct depth; the closer the depth measurement is to the true value, the closer π is to 1. As shown in Figure 4, y_{min} and y_{max} are the upper and lower limits of the uniform distribution.

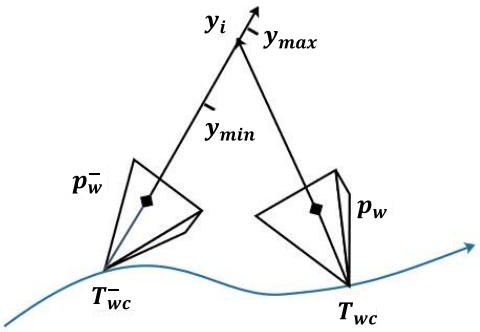


Figure 4. Depth filter measurement model. 3D space projection is performed between the current template position T_{wc} and the previous template $T_{\bar{w}c}$ corresponding to the 2D coordinates. As the number of corresponding template matching increases, the distribution of depth gradually converges.

Equation (20) can be approximated by a Beta-Gaussian distribution. The step 4 of Algorithm 1 summarizes the depth estimation steps of our algorithm.

$$p(Z, \pi | a_i, b_i, \mu_i, \sigma_i) = \text{Beta}(\pi \Big| a_i, b_i) N(Z \Big| u_i, \sigma_i^2) \tag{21}$$

3.2.5. Pose Estimation

According to Algorithm 1, we can calculate corresponding 3D point sets $P = \{p_1, p_2, \cdots, p_n\}$ and $Q = \{q_1, q_2, \cdots, q_n\}$ through Sections 3.2.3 and 3.2.4, where the number of corresponding point pairs is n. Then, the optimal coordinate transformation is calculated by the ICP algorithm, that is, the rotation matrix $\boldsymbol{R}$ and the translation vector $\boldsymbol{t}$; the problem can be described as follows:

$$(\boldsymbol{R}, \boldsymbol{t}) = \underset{R \in SO(3), \boldsymbol{t} \in \mathbb{R}^3}{\text{argmin}} \sum_{i=1}^{n} w_i \parallel (\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t}) - \boldsymbol{q}_i \parallel^2 \tag{22}$$

where w_i represents the weight of each point. $\boldsymbol{R}$ and $\boldsymbol{t}$ are our required rotation matrix and translation vector. To reduce the reprojection error during tracking, the Tukey weight function is used as follows:

$$w_i = \begin{cases} \left(1 - \frac{x^2}{b^2}\right)^2 & |x| \leq |b| \\ 0 & otherwise \end{cases} \tag{23}$$

where $x = \|p_w^- - p_w\|$ and $b = 5$ pixels. Additionally, we find that with w_i multiplied by the internal probability π, the estimated results are significantly better. Template points that are well tracked get the highest weight, while template points whose matching does not converge are usually removed due to their errors being too large.

Algorithm 1: Pseudo-code of pose estimation.

Step 1: Extracting feature points and edge points

1: Extract feature points by Harris detector

2: Extract edge points by Canny detector

3: Pick edge points around feature points as template edges

Step 2: Computing the optical flow and the spatio-temporal windows of events

4: Input: template edges P and event point set W

Output: optical flow $\boldsymbol{\theta}$

Initialize optical flow $\boldsymbol{\theta} = 0$

for $i = 1 : m$ do

for $j = 1 : n$ do

$$I(x) = \sum_{j=1}^{n} \delta(X_i - X_j'); Var(I) = \int_{\Omega} (I(x) - \mu)^2 dx; \theta \leftarrow \theta + \mu \nabla_{\theta} \left(\sum_{k=1}^{N_l} Var(I_k) \right)$$

end for

end for

5: Update the spatio-temporal window: $t_{n+1} = t_n + 3/\boldsymbol{\theta}_{average}^{n}$

Step 3: Updating the template edges using the optical flow and IMU data

6: Update the postion of feature points: $x_i(t_1) = x_i(t_0) + v.(t_1 - t_0)$

7: Update the relative position: $x_j^{relative}(t_1) = Nor\left(K\boldsymbol{R}_j K^{-1} x_j(t_0)\right) - Nor\left(K\boldsymbol{R}_j K^{-1} x_i(t_0)\right)$

8: Update the postion of edge points: $x_j(t_1) = x_j^{relative}(t_1) + x(t_1)$

Step 4: Computing the depth value of template edges using a depth filter

9: Triangulation depth: y_i

10: Depth filter: $q(y, \pi | a_i, b_i, \mu_i, \sigma_i) = \text{Beta}(\pi | a_{\,i}, b_i) N(y \mid u_i, \sigma_i^2)$

Input: Triangulation depth: y_i

Output: depth value y

for $i = 1 : n$ do

$$\mu_i' = C_1' m_i + C_2' \mu_i; \sigma_i'^2 = C_1' \left(y_i^2 + m_i^2\right) + C_2' \left(\sigma_i^2 + \mu_i^2\right)$$

$$f_i = C_1' \frac{a_i + 1}{a_i + b_i + 1} + C_2' \frac{a_i}{a_i + b_i + 1}$$

$$e_i = C_1' \frac{(a_i + 1)(a_i + 2)}{(a_i + b_i + 1)(a + b_i + 2)} + C_2' \frac{a_i(a_i + 1)}{(a_i + b_i + 1)(a_i + b_i + 2)}$$

$$a_i' = \frac{e_i - f_i}{f_i - \frac{e_i}{f_i}}; b_i' = \frac{1 - f_i}{f_i} \cdot a_i'; \frac{1}{y_i^2} = \frac{1}{\sigma_i^2} + \frac{1}{\tau_i^2}; m_i = y_i^2 \left(\frac{\mu_i}{\sigma_i^2} + \frac{x_i}{\tau_i^2} \right)$$

end for

Step 5: Pose estimation using the ICP agorithm

11: ICP agorithm: $(\boldsymbol{R}, \boldsymbol{t}) = \underset{R \in SO(3), \boldsymbol{t} \in \mathbb{R}^3 i=1}{\text{argmin}} \sum_{i=1}^{n} w_i \parallel (\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t}) - \boldsymbol{q}_i \parallel^2$

Output

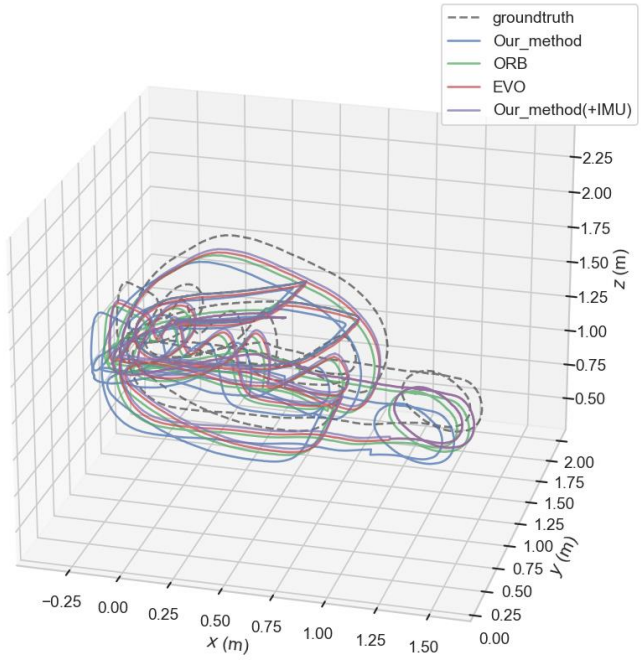
$6-$DoF event camera pose translation $\in R^3$, rotation $\in SO(3)$.

4. Experiments

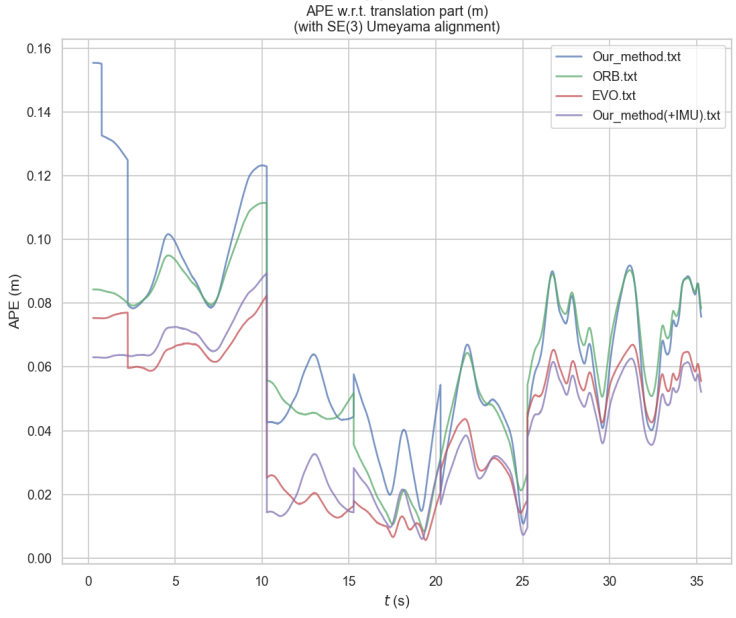
In order to demonstrate the performance of our proposed algorithm, some widely used event datasets were utilized to estimate camera trajectories. These datasets were generated using the DAVIS240C from iniLabs, and they contain events, images, and IMU measurements. The algorithm was evaluated by selecting some challenging sequences from the IJRR event camera datasets [33]. The sequences are from several indoor and outdoor scenes with different lighting conditions. Experiment 1 and Experiment 2 were evaluated indoors. Compared with Experiment 1, Experiment 2 had richer scene textures. In Experiment 3, the camera trajectory was evaluated outdoors. In several experiments, the camera trajectories obtained by our method were compared with frame-tracking-based (ORB) visual odometry and event-tracking-based (EVO) visual odometry, and we also compared the trajectories of our method with and without IMU data. Among the methods tested, only ours uses IMU data to calibrate rotation. The absolute pose error was calculated between the poses estimated by other algorithms and our method and the ground truth.

In the IJRR datasets [33], we selected representative scenes for trajectory estimation. We tested different visual odometry methods on three sequences: shapes_6dof, boxes_6dof, and outdoors_6dof. Figures 5–7 show the test results of the different algorithms on the

three sequences. These figures show the trajectories produced by visual odometry such as Our_method, ORB, and EVO. Table 1 shows the position error, orientation error, and absolute pose error of different methods compared to the ground truth in different scenarios. The position error was calculated as the Euclidean distance between the estimated value of the camera trajectory and the true value. Rotation error was computed as the angular error based on the estimated and true values of the rotation matrix. The absolute pose error compares the estimated trajectory and the reference trajectory and calculates the statistics of the entire trajectory, which is suitable for testing the global consistency of the trajectory. Furthermore, to reflect the contribution of the IMU to the estimation calibration, we obtained the results of our method on the shapes_6dof, boxes_6dof, and outdoors_6dof sequences without fusing the IMU data.



(**a**)



(**b**)

Figure 5. *Cont.*

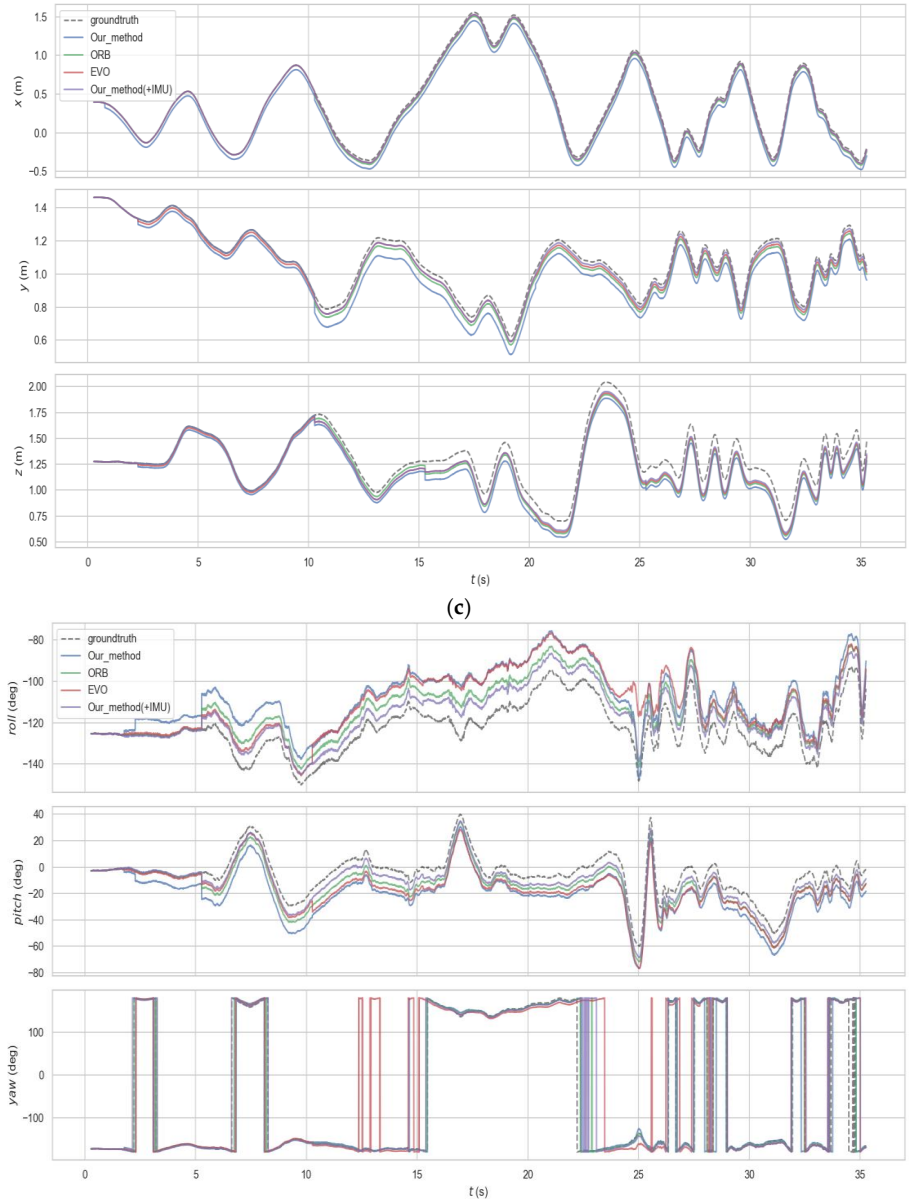
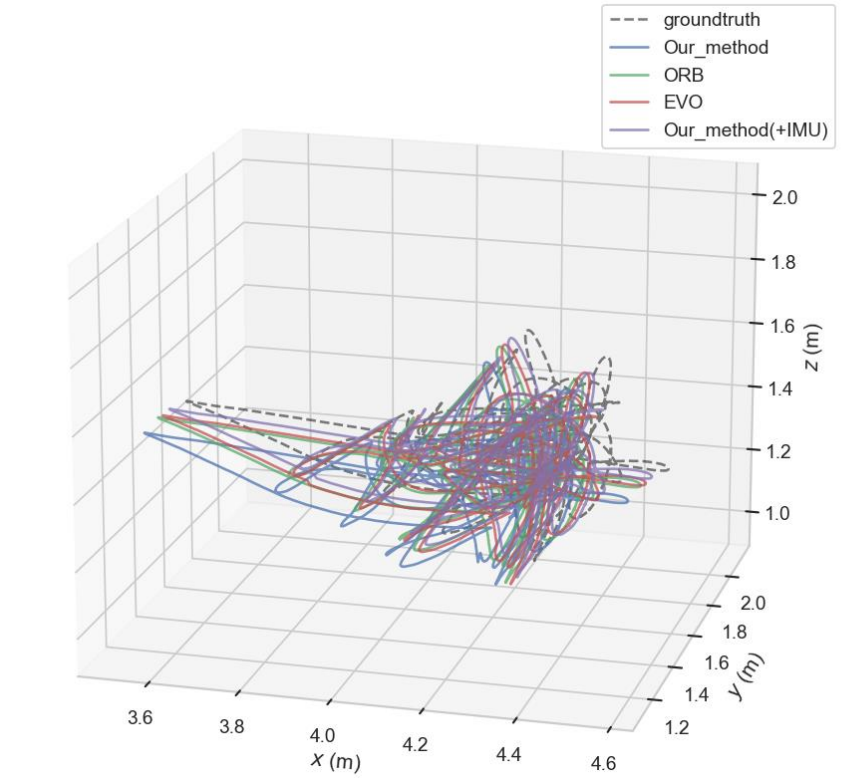
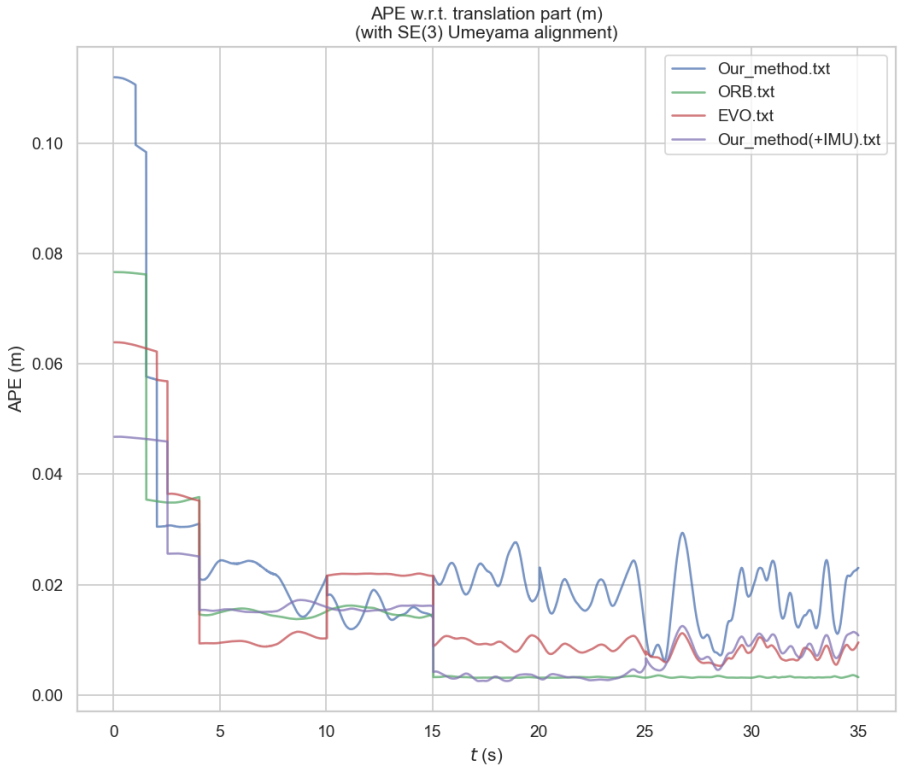


Figure 5. VO Experiment 1 (shapes_6dof): (**a**) 3D event camera trajectories estimated by different algorithms; (**b**) Comparison of absolute pose errors between different algorithms; (**c**) The translation position of the camera estimated by different algorithms; (**d**) The rotation position of the camera estimated by different algorithms.

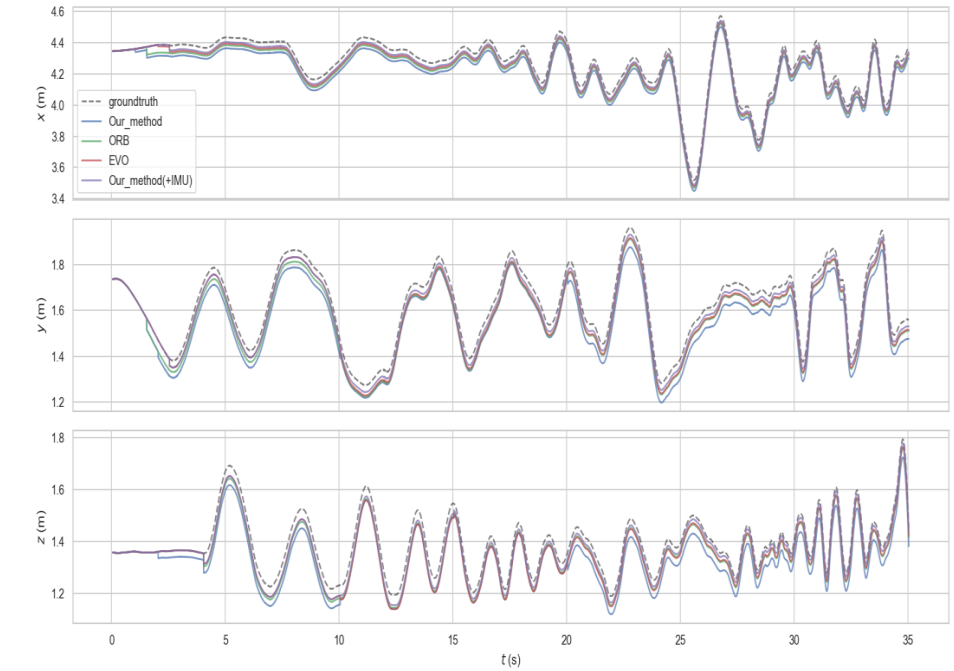


(**a**)

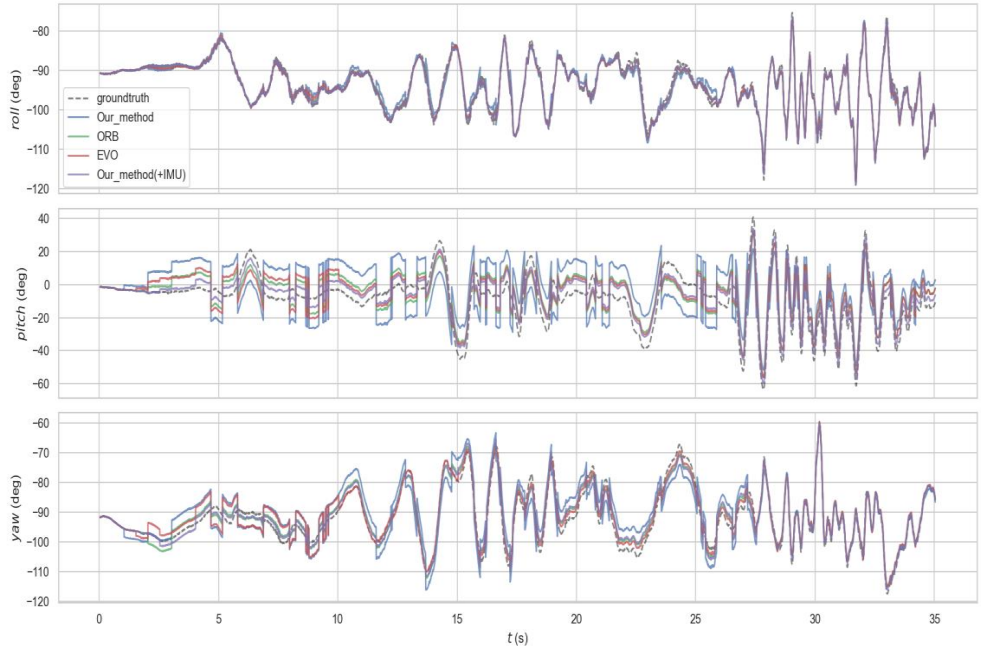


(**b**)

Figure 6. *Cont*.

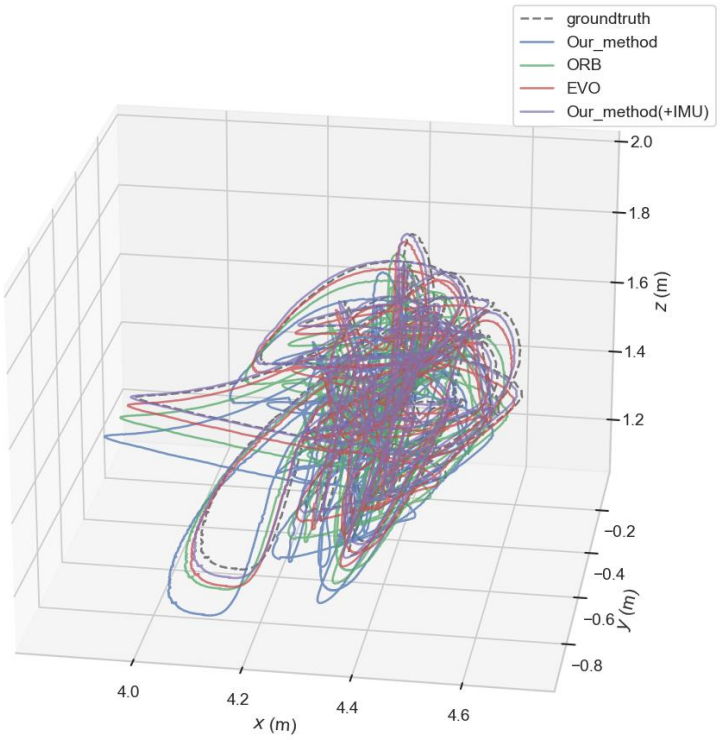


(**c**)

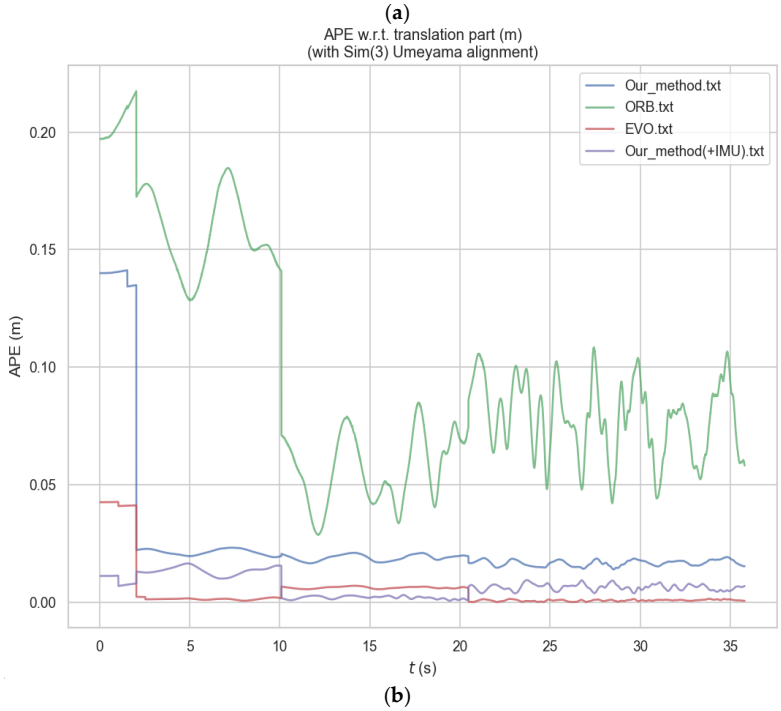


(**d**)

Figure 6. VO Experiment 2 (boxes_6dof): (**a**) 3D event camera trajectories estimated by different algorithms; (**b**) Comparison of absolute pose errors between different algorithms; (**c**) The translation position of the camera estimated by different algorithms; (**d**) The rotation position of the camera estimated by different algorithms.

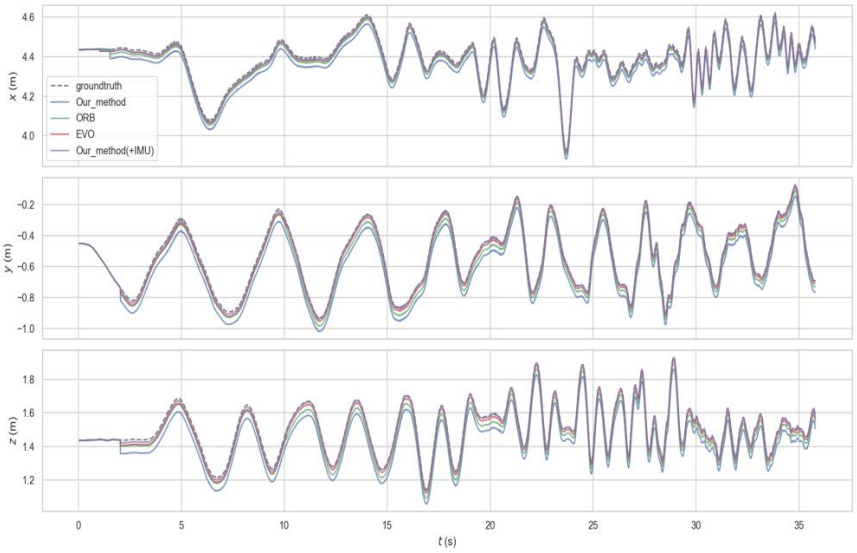


(**a**)

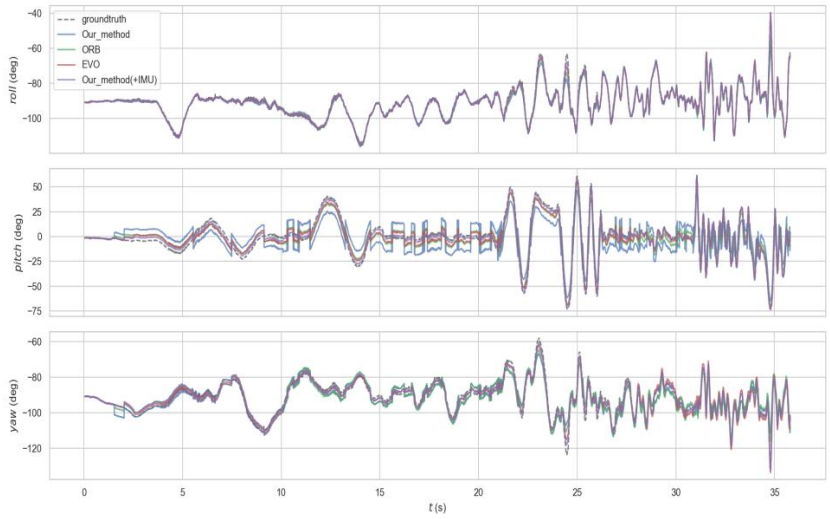


(**b**)

Figure 7. *Cont.*



(**c**)



(**d**)

Figure 7. VO Experiment 3 (outdoors_6dof): (**a**) 3D event camera trajectories estimated by different algorithms; (**b**) Comparison of absolute pose errors between different algorithms; (**c**) The translation position of the camera estimated by different algorithms; (**d**) The rotation position of the camera estimated by different algorithms.

Table 1. Absolute pose error (m/s), position error (m/s), and rotation error (deg/s) test results in different scenarios with different methods [33].

	Sequences	Our_Method	ORB	Our_Method (+IMU)	EVO
Absolute pose error	shapes_6dof	0.0815	0.0780	**0.0315**	0.0435
	boxes_6dof	0.0434	0.0369	**0.0204**	0.0344
	outdoors_6dof	0.1416	0.2357	0.0461	0.0227
Position error	shapes_6dof	0.1562	0.0845	**0.0391**	0.0496
	boxes_6dof	0.2255	0.0956	**0.0436**	0.0586
	outdoors_6dof	0.1728	0.0735	0.0325	0.0265
Rotation error	shapes_6dof	2.4562	1.8342	**0.6443**	0.9093
	boxes_6dof	3.5432	1.6901	0.5426	0.5084
	outdoors_6dof	3.4734	1.6042	**0.4453**	0.7253

In Experiment 1 (Figure 5), we tracked simple shapes using different visual odometry methods. As the camera movement speed increases, the resolution of the captured image gradually decreases. The ORB method does not extract enough feature points, which leads to tracking failure. Thus, we chose a tracking duration of only 35 s. The position errors of the different methods were 15.62 cm, 8.45 cm, 3.91 cm, and 4.96 cm for Our_method, ORB, Our_method(+IMU), and EVO, respectively. The rotation errors of the different methods were 2.46 deg, 1.83 deg, 0.64 deg, and 0.91 deg for Our_method, ORB, Our_method(+IMU), and EVO, respectively. In Experiment 2 (Figure 6), the natural textures in the box scene are richer and also generate more event information. Since the EVO algorithm tracks the camera trajectory through pure event information, too many event streams cause the tracking time to be longer. The position errors of the different methods were 22.55 cm, 9.56 cm, 4.36 cm, and 5.86 cm for Our_method, ORB, Our_method (+IMU), and EVO, respectively. The rotation errors of the different methods were 3.54 deg, 1.69 deg, 0.54 deg, and 0.51 deg for Our_method, ORB, Our_method (+IMU), and EVO, respectively. In Experiment 3 (Figure 7), we evaluated several methods on outdoor datasets. The position errors of the different methods were 17.28 cm, 7.35 cm, 3.25 cm, and 2.65 cm for Our_method, ORB, Our_method (+IMU), and EVO, respectively. The rotation errors of the different methods were 3.47 deg, 1.60 deg, 0.45 deg, and 0.73 deg for Our_method, ORB, Our_method (+IMU), and EVO, respectively. The mean scene depth was 3 m. Overall, our algorithm performed slightly better than EVO in tracking performance and far outperformed ORB, but our tracking time was much faster than that of EVO in complex textured scenes. Our algorithm provides low-latency pose updates and preserves the nature of event-based data. These results represent the success of fusing frames and events for tracking in natural scenes and 6-DoF motion.

5. Conclusions

In this paper, we presented a novel visual odometry algorithm based on the DAVIS. The initial tracking template was extracted from the image sequence. A contrast maximization algorithm was presented to estimate the optical flow estimated by minimizing the distance between the event and the templates in the spatio-temporal windows. Then, IMU data were used to calibrate the rotational position of the tracking template. A Beta-Gaussian distribution depth filter was presented to update the depth of each pixel on the edge of templates. These templates were used to achieve lower-latency camera trajectories by the ICP algorithm. We tested our method on several scenes with different textures in the DAVIS dataset. Compared with the visual odometry algorithm ORB and EVO, the proposed algorithm showed more advantageous performance in terms of accuracy and robustness.

Author Contributions: Formal analysis, H.X.; investigation, X.G.; methodology, X.G.; project administration, X.L.; writing—original draft, H.X.; writing—review & editing, X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Natural Science Foundation of China under Grant U2003110, and in part by the Key Laboratory Project of Shaanxi Provincial Department of Education (No. 20JS110), Xi'an Science and Technology Planning Project (No. 2020KJRC0084).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

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

References

1. Li, J.; Qin, J.H.; Wang, J.; Li, J. OpenStreetMap-Based Autonomous Navigation for the Four Wheel-Legged Robot Via 3D-Lidar and CCD Camera. *IEEE Trans. Ind. Electron.* **2022**, *69*, 2708–2717. [CrossRef]
2. Liu, D.; Yang, T.L.; Zhao, R.; Wang, J.; Xie, X. Lightweight Tensor Deep Computation Model with Its Application in Intelligent Transportation Systems. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 2678–2687. [CrossRef]

3. Rong, S.; He, L.; Du, L.; Li, Z.; Yu, S. Intelligent Detection of Vegetation Encroachment of Power Lines with Advanced Stereovision. *IEEE Trans. Power Deliv.* **2021**, *36*, 3477–3485. [CrossRef]
4. Leng, C.; Zhang, H.; Li, B.; Cai, G.; Pei, Z.; He, L. Local feature descriptor for image matching: A Survey. *IEEE Access* **2019**, *7*, 6424–6434. [CrossRef]
5. Tedaldi, D.; Gallego, G.; Mueggler, E.; Scaramuzza, D. Feature Detection and Tracking with the Dynamic and Active-pixel Vision Sensor (DAVIS). In Proceedings of the Second International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP), Krakow, Poland, 13–15 June 2016.
6. Schuman, C.D.; Potok, T.E.; Patton, R.M.; Birdwell, J.D.; Dean, M.E.; Rose, G.S.; Plank, J.S. A survey of neuromorphic computing and neural networks in hardware. *arXiv* **2017**, arXiv:1705.06963.
7. Rebecq, H.; Horstschaefer, T.; Gallego, G.; Scaramuzza, D. EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time. *IEEE Robot. Autom. Lett.* **2017**, *2*, 593–600. [CrossRef]
8. Chen, S.; Guo, M. Live demonstration: CELEX-V: A 1m pixel multi-mode event-based sensor. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–17 June 2019.
9. Zhang, Z.; Wan, W. DOVO. Mixed Visual Odometry Based on Direct Method and Orb Feature. In Proceedings of the 2018 International Conference on Audio, Language and Image Processing (ICALIP), Shanghai, China, 16–17 July 2018.
10. Guo, M.; Huang, J.; Chen, S. Live demonstration: A 768 × 640 pixels 200Meps dynamic vision sensor. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017.
11. Posch, C.; Matolin, D.; Wohlgenannt, R. A QVGA 143 DB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain cds. *IEEE J. Solid-State Circuits* **2011**, *46*, 259–275. [CrossRef]
12. Li, Y.; Li, J.; Yao, Q.; Zhou, W.; Nie, J. Research on Predictive Control Algorithm of Vehicle Turning Path Based on Monocular Vision. *Processes* **2022**, *10*, 417. [CrossRef]
13. Stoffregen, T.; Gallego, G.; Drummond, T.; Kleeman, L.; Scaramuzza, D. Event-Based Motion Segmentation by Motion Compensation. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October 2019.
14. Stoffregen, T.; Kleeman, L. Event Cameras, Contrast Maximization and Reward Functions: An Analysis. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
15. Gallego, G.; Delbruck, T.; Orchard, G.M.; Bartolozzi, C.; Taba, B.; Censi, A.; Leutenegger, S.; Davison, A.; Conradt, J.; Daniilidis, K.; et al. Event-based Vision: A Survey. *arXiv* **2020**, arXiv:1904.08405.
16. Chiang, M.L.; Tsai, S.H.; Huang, C.M.; Tao, K.T. Adaptive Visual Serving for Obstacle Avoidance of Micro Unmanned Aerial Vehicle with Optical Flow and Switched System Model. *Processes* **2021**, *9*, 2126. [CrossRef]
17. Duo, J.; Zhao, L. An Asynchronous Real-Time Corner Extraction and Tracking Algorithm for Event Camera. *Sensors* **2022**, *22*, 1475. [CrossRef] [PubMed]
18. Mitrokhin, A.; Fermuller, C.; Parameshwara, C.; Aloimonos, Y. Event-based moving object detection and tracking. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018.
19. Li, K.; Shi, D.; Zhang, Y.; Li, R.; Qin, W.; Li, R. Feature Tracking Based on Line Segments with the Dynamic and Active-Pixel Vision Sensor (DAVIS). *IEEE Access* **2019**, *7*, 110874–110883. [CrossRef]
20. 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* **2022**, *9*, 166588–166602. [CrossRef]
21. Zhu, A.Z.; Chen, Y.; Daniilidis, K. Realtime time synchronized event-based stereo. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018.
22. Ozawa, T.; Sekikawa, Y.; Saito, H. Accuracy and Speed Improvement of Event Camera Motion Estimation Using a Bird's-Eye View Transformation. *Sensors* **2022**, *22*, 773. [CrossRef]
23. Kim, H.; Kim, H.J. Real-Time Rotational Motion Estimation with Contrast Maximization Over Globally Aligned Events. *IEEE Robot. Autom. Lett.* **2022**, *6*, 6016–6023. [CrossRef]
24. Pal, B.; Khaiyum, S.; Kumaraswamy, Y.S. 3D point cloud generation from 2D depth camera images using successive triangulation. In Proceedings of the 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bengaluru, India, 21–23 February 2017.
25. Umair, M.; Farooq, M.U.; Raza, R.H.; Chen, Q.; Abdulhai, B. Efficient Video-based Vehicle Queue Length Estimation using Computer Vision and Deep Learning for an Urban Traffic Scenario. *Processes* **2021**, *9*, 1786. [CrossRef]
26. Gallego, G.; Lund, J.E.A.; Mueggler, E.; Rebecq, H.; Delbruck, T.; Scaramuzza, D. Event-based, 6-DOF camera tracking from photometric depth maps. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 2402–2412. [CrossRef]
27. Shao, F.; Wang, X.; Meng, F.; Rui, T.; Wang, D.; Tang, J. Real-time traffic sign detection and recognition method based on simplified Gabor wavelets and CNNs. *Sensors* **2018**, *18*, 3192. [CrossRef]
28. Alzugaray, I.; Chli, M. Asynchronous corner detection and tracking for event cameras in real time. *IEEE Robot. Autom. Lett.* **2018**, *3*, 3177–3184. [CrossRef]
29. Zhou, Y.; Gallego, G.; Shen, S. Event-Based Stereo Visual Odometry. *IEEE Trans. Robot.* **2022**, *37*, 1433–1450. [CrossRef]
30. Mueggler, E.; Gallego, G.; Scaramuzza, D. Continuous-Time Trajectory Estimation for Event-based Vision Sensors. In Proceedings of the Robotics: Science and Systems, Rome, Italy, 17 July 2015.

31. Kim, H.; Leutenegger, S.; Davison, A.J. Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 16 October 2016; Springer: Cham, Switzerland, 2016.
32. Mueggler, E.; Gallego, G.; Rebecq, H.; Scaramuzza, D. Continuous-Time Visual-Inertial Odometry for Event Cameras. *IEEE Trans. Robot.* **2018**, *34*, 1425–1444. [CrossRef]
33. Mueggler, E.; Rebecq, H.; Gallego, G.; Delbruck, T.; Scaramuzza, D. The Event Camera Dataset and Simulator: Event-Based Data for Pose Estimation, Visual Odometry, and SLAM. *Int. J. Robot. Res.* **2017**, *36*, 142–149. [CrossRef]