Neural Implicit Event Generator for Motion Tracking

Mana Masuda\textsuperscript{1\ast}, Yusuke Sekikawa\textsuperscript{2\ast}, Ryo Fujii\textsuperscript{1}, Hideo Saito\textsuperscript{1}

Abstract—We present a novel framework of motion tracking from event data using implicit expression. Our framework use pre-trained event generation MLP named implicit event generator (IEG) and does motion tracking by updating its state (position and velocity) based on the difference between the observed event and generated event from the current state estimate. The difference is computed implicitly by the IEG. Unlike the conventional explicit approach, which requires dense computation to evaluate the difference, our implicit approach realizes efficient state update directly from sparse event data. Our sparse algorithm is especially suitable for mobile robotics applications where computational resources and battery life are limited. To verify the effectiveness of our method on real-world data, we applied it to the AR marker tracking application. We have confirmed that our framework works well in real-world environments in the presence of noise and background clutter.

\textsuperscript{\ast} Equal contribution
\textsuperscript{1}Mana Masuda, Ryo Fujii and Hideo Saito are with the School of Science for Open and Environmental Systems, Keio University, Tokyo, Japan
\textsuperscript{2}Yusuke Sekikawa is with Denso IT Laboratory, Tokyo, Japan

I. INTRODUCTION

Motion tracking from spatiotemporal data such as video frames is one of the fundamental functionality for robotics applications. It either tracks a specific target position from the camera or tracks a camera position from worlds coordinates. The application of tracking algorithms ranges from robotic arm manipulation, and mobile robot localization. Before the tracking, the target’s initial position is identified by globally matching known patterns with camera observations. And then, a tracking algorithm sequentially updates the position using. This update is usually done by using gradient-based algorithms. However, in existing systems using frame cameras \cite{1}, \cite{2}, the loss function defined by the difference between the known pattern and the observation is often trapped by the local minimum when the motion of the camera or the target is fast; results in the failure of tracking.

Event cameras are bio-inspired novel vision sensors...
that mimic biological retinas and report per-pixel intensity changes in the form of asynchronous event stream instead of intensity frames. Due to the unique operating principle, event-based sensing offers significant advantages over conventional cameras: high dynamic range (HDR), high temporal resolution, and blurless sensing. As a result, it has the potential to achieve robust tracking in intense motion and severe lighting environments.

Many methods have been proposed which use these features to achieve tracking in high-speed or high-dynamic-range environments [3], [4], [5], [6], [7], [8], [9], [10]. In particular, Bryner et al. [7] and Gehrig et al. [6], [10] proposed method as an extension of Lucas-Kanade tracker (KLT) [11], [12] developed for frame-based video data. They extend KLT to sparse intensity difference data from the event-based camera. In their method, position and velocity are updated by minimizing the difference between the observation and the expected observation, as shown in Fig. 1(a). The expected observation is computed by warping the given photometric 2D/3D map using the current estimate of position and velocity. Unfortunately, these methods could not fully take advantage of the sparseness of event data. Still, they require dense computation to compute the difference, and this optimization needs to be repeated many times until convergence.

To realize efficient tracking, we focus on leveraging the sparsity of event data. To this end, we propose a novel framework for object tracking using implicit expression. As shown in Fig. 1(b), our method achieves the goal by updating the position and velocity based on the difference between the events generated by the implicit event generator (IEG) and the observed events. In our method, the difference is directly computed by the feed-forward operation of sparse event data without explicitly predict and generate events from the current position and velocity estimates. This implicit approach is totally different from the conventional explicit approach, which requires dense computation to evaluate the difference. The gradient of position and velocity w.r.t. the IEG is computed by simply backpropagating the error through IEG realized by MLP. The entire operation, namely feed-forward operation for difference evaluation and derivative computation w.r.t. the difference, are executed sparsely; computation happens only where the events occur.

In this paper, we first introduce a specific tracking algorithm based on this new framework. For the sake of proof of concept, we conducted experiments using an AR marker captured by an actual event-based camera to show that the method based on this new framework can be applied to real problems.

Our main contributions are summarized as follows:

- We propose a novel object tracking framework by using an implicit event generator (IEG). In our method, computation is performed only at the event’s location; therefore, it could be significantly efficient than the conventional explicit method requiring dense computation.
- We conduct experiments using data captured from AR markers using a real event-based camera; demonstrate the proposed method can be applied to practical problems.

II. RELATED WORKS

Most of the existing tracking algorithms are developed based on RGB cameras and track the target object frame by frame. Recently, the Siamese network-based approach achieve a better result and many methods have been proposed based on it [13], [14], [15]. RGB images, however, cannot provide sufficient tracking performance under fast motion and low-illumination conditions.

By taking advantage of the HDR and no-motion blur properties of event cameras, a method for tracking fast objects and in harsh light environments has been proposed. Rebecq et al. [3] and Zihao et al. [4] convert IMU data into lower rate pieces of information at desired times where events are collected. Chamorro et al. [9] and Alzugaray et al. [8] explore the high-speed feature tracking with the dynamic vision sensor (DVS) [16], and Gallego et al. [5] presented a filter-based approach that requires to maintain a history of past camera positions which are continuously refined. Bryner et al. [7] and Gehrig et al. [6], [10] use the extended Lucas-Kanade method [11], [12] and update the position and velocity by warping the given photometric 3D map or frame and taking the difference between the observation and the expected observation.

Considering the processing in the neural network, sparse event data is inherently difficult to handle using a conventional deep neural network. Therefore, many neural network-based
methods for event data first convert sparse events to dense frames and then process it by a dense neural network such as CNN [17], [18], [19], [20], [21]. Comparing with these methods, the computational complexity of our approach could be significantly smaller.

Since NeRF [22], implicit volumetric representations have attracted in computer vision community [23], [24], [25]. In this paradigm, the RGB-D scene representation is learned in a parameterized manner by a multi-layer perceptron (MLP). Some methods have been proposed to estimate the position of an object by back-propagating the scene representation MLP, such as iNeRF [26], [27]. These methods, however, are proposed for dense RGB images and do not apply to sparse event data with time-series information. We used IEG to represent events that occur in response to the position, time, and velocity of a pixel, making it possible to implicitly represent events under a variety of conditions.

III. METHOD

We present a framework for motion tracking using a neural implicit event generator (IEG) to realize efficient motion update by sparse operations. In our framework, position \( T \) and velocity \( \dot{T} \) is updated by simply feed-forwarding sparse event observation into IEG and then backpropagate the output through the IEG. The IEG \( G \) is realized by differential MLP (parameterized by \( \Theta \)), and it is trained before inference to output intensity changes of tracking-target for given position \( T \) and velocity \( \dot{T} \). To track the motion, i.e., estimate \( (T, \dot{T}) \), at inference, \( (T, \dot{T}) \) is optimized to minimized the difference \( L \) between observation and estimation for observation of event stream \( \{ e \} \) such that:

\[
\dot{T}, \ddot{T} = \arg \min_{T, \ddot{T}} L(T, \dot{T} | \{ e \}, \Theta)
\]  

(1)

In this formulation, explicit generation of events using by current estimate \( (T, \dot{T}) \) does not required as existing method (Fig.1 left). Instead, ours implicitly generates events and computes the difference directly from the sparse event (Fig.1 right).

This paper applied the proposed framework for 3DoF camera motion in a 2D planar scene.

A. Formulation of IEG

We represent intensity change appearance as a 6D vector-valued function whose input is a 2D location \( x = (x, y, r) \) and time \( t \) and 3DoF \( \dot{T} = (v_x, v_y, \omega) \), and whose output is an detected intensity change \( \delta \) in \([+/−/NaN]\). We approximate the intensity change appearance with continuous 3DoF with the IEG \( G_\Theta : (x, y, t, v_x, v_y, \omega) \rightarrow \delta \) and optimize its weights \( \Theta \) to map from each input 6D vector to its corresponding intensity change \( \delta \).

B. Artificial Data of Intensity Change

To learn the intensity change with velocity using IEG, we create artificial intensity change data based on the following equation:

\[
|\frac{\partial I}{\partial x} \cdot v(x) \Delta t| > C,
\]

(2)

where \( v(x) \) is the speed of coordinate \( x \). \( C \) is the threshold of event firing.

To create intensity change data at various velocities, we use a randomly determined velocity \( \dot{T} = (v_x, v_y, \omega) \), and from the edge coordinates of a known object \((x_e, y_e)\) and the intensity derivative data at those coordinates \( \frac{\partial I}{\partial x} = (\partial_i x, \partial_i y) \). From these values, we can calculate the theoretical intensity change value \( \delta_t \) at the coordinate \((x_t, y_t)\), which corresponding to the edge coordinate \((x_e, y_e)\) when the object moves by time \( t \) as follows:

\[
\delta_t = U \left( \frac{\partial I}{\partial x}, v, \omega, t \right) = \left( \frac{\cos(\omega t)}{\sin(\omega t)} - \sin(\omega t) \right) \left( \frac{\partial_i x}{\partial_i y} \right) \left( v_x + \cos(\omega t) \right) \left( v_y + \sin(\omega t) \right) t
\]

(3)

We set the value corresponding to the threshold of event firing as \( \delta \), and define \( \delta_t' \) as follows:

\[
\delta_t' = \begin{cases} +1 & (\delta_t > \delta) \\ NaN & (-\delta < \delta_t <= \delta) \\ -1 & (\delta_t < -\delta) \end{cases}
\]

(4)

To perform the optimization operation described in [III-D], we applied a Gaussian blur to the intensity change \( \delta_t' \) to make it differentiable. Depending on the distance \((dx, dy)\) to the intensity change \( \delta_t' \), we use an extended Gaussian function \( G(dx, dy) \) with mean 0 and variance \( \sigma \):

\[
G(dx, dy) = \exp \left( -\frac{dx^2 + dy^2}{2\sigma^2} \right).
\]

(5)

To avoid interference with events generated by other edges, the blur width is set to \( w \), and the Gaussian blurred intensity


Algorithm 1 Tracking using gradient of IEG

**Ensure:** \( \hat{T}_k = (x_k, y_k) \) is the inferred position at \( k \)-th iteration and \( \hat{T}_k = (v_{x,k}, v_{y,k}, \omega_k) \) is the inferred velocity at \( k \)-th iteration.

1. \( \epsilon \leftarrow \inf \), \( k \leftarrow 0 \)
2. \( \hat{T} \leftarrow T_0 \)
3. \( \hat{T} \leftarrow (0, 0, 0) \)
4. **while** \( \epsilon \geq \bar{\epsilon} \) **do**
5. \( L_k = \sum_{i \in M_k} (1 - |G_{\Theta}(W(x_i, \hat{T}_k), \hat{T})|) \)
6. **Update** \( \hat{T}_k, \hat{v}_k \)
7. \( \epsilon \leftarrow L_{k-1} - L_k, k \leftarrow k + 1 \)
8. **end while**

change \( \delta_a \) at \((x'_i, y'_i) = (x_i + dx, y_i + dy)\) is calculated as follows:

\[
\delta_a = \begin{cases} 
G(dx, dy) \times \delta'_t & (\sqrt{dx^2 + dy^2} \leq w) \\
0 & (\sqrt{dx^2 + dy^2} > w)
\end{cases}.
\]

**(6)**

C. Training IEG

We trained the IEG with the artificial data as described in III-B. When we input the 6D vector of artificial data \((x, y, t, v_x, v_y, \omega)\) and the IEG outputs the estimated intensity change \( \delta_p \). The IEG is updated by taking the difference between the generated intensity change \( \delta_g \) and the artificially calculated intensity change \( \delta_a \):

\[
L_{\text{train}} = ||\delta_g - \delta_a||_2.
\]

**(7)**

D. Tracking using gradient of IEG

Let \( \Theta \) be the parameters of the trained and fixed IEG \( G \), \( x_i = (x_i, y_i, t_i) \) the place \((x_i, y_i)\) and time \( t_i \) of \( i \)-th detected intensity change, \( \delta(x_i) \) is the intensity change sensed at \( x_i \), \((\hat{T}_k, \hat{v}_k)\) the estimated position and velocity at current optimization step \( k \). Before inputting into the IEG, \( x_i \) is transformed by the transformation function \( W \) based on \( T_k \) in the 2D plane.

\[
W(x_i, T) = \begin{pmatrix} \cos(-T_x) & -\sin(-T_x) \\ \sin(-T_x) & \cos(-T_x) \end{pmatrix} \begin{pmatrix} x_i - T_{ix} \\ y_i - T_{iy} \end{pmatrix},
\]

**(8)**

where \( T_\| \) is the translation component and \( T_\perp \) is the rotation component of \( T \). During tracking, optimization is performed so that the absolute value output by IEG is close to 1;

\[
L_{\text{update}} = \sum_{i \in M} (1 - |G_{\Theta}(W(x_i, \hat{T}_k), \hat{T})|),
\]

**(9)**

where \( M \) is the number of detected events. We employ gradient-based optimization to solve for \((\hat{T}, \hat{v})\) as defined in Eq. [1]. The optimization is iterated until the change in loss \( \epsilon \) is less than the optimization threshold \( \bar{\epsilon} \).

In other words, this optimization problem can be defined as finding \((\hat{T}, \hat{v})\) by solving the function \( 1 - |G| = 0 \).

E. Object Tracking

During tracking, the events of an existing small group (window) consisting of \( M \) events are processed as in III-B to obtain the position \( T_j \) and velocity \( \dot{T}_j \) at \( j \)-th iteration. Then, at the \( j + 1 \)-th iteration, slide the window with sliding width \( K \), and update the position \( T_{j+1} \) and velocity \( \dot{T}_{j+1} \) by optimizing again as in III-D. The position and velocity do not change much between the \( j \)-th and \( j + 1 \)-th iterations, so to update the \((T_j, \dot{T}_j)\) at \( j + 1 \)-th iteration could help to reduce the latency.

Our pipeline does not take acceleration \( \ddot{T} \) into account, but this is not a problem because if the window size \( M \) is sufficiently small, the motion within the window can be approximated as constant velocity motion.

IV. EXPERIMENTAL RESULTS

The effectiveness of this method is that it can estimate the motion of an object only from the event data. To demonstrate the effectiveness, we experiment using an image pattern with AR markers as a target scene to verify whether the motion of the AR marker can be accurately estimated or not. Since the events only occur from the edge of the AR marker, we can effectively show that the camera motion can be estimated using only sparse events. Another reason for using AR markers is that they are easy to annotate (detect and map feature points) and can be used as a criterion for accuracy evaluation. To capture the event data, we used Prophesee’s event camera Gen3 [28] with 120dB at 10000fps. As the problem setup for object tracking, we can assume that the initial position of the object is known by the object detection algorithm, so we gave its approximate position as the initial value for tracking. The evaluation was done quantitatively and qualitatively.

A. Experimental Setup

We evaluated the performance of our algorithm by tracking AR Marker in two environments; one on the white background and the other is on the wood grain pattern. Since our proposed tracking algorithm is limited to 2 dimension, we fixed the camera on the dolly as shown in Fig. 4.
so that the distance between the camera and the AR marker is constant. We use optimizer Adam [29] with the recommended hyperparameter settings of: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$ for training IEG and stochastic gradient descent (SGD) for gradient-based optimization. The learning rate was 0.0001 for both training and inference and kept constant. We set the threshold of synthetic event firing as $\delta = 0.001$ and the optimization threshold as $\bar{\epsilon} = 1.0 \times 10^{-6}$.

### TABLE I

**Error Measurements of on White Background and on Wood Grain Background**

| $K$ | $w \text{ [pixel]}$ | $x \text{ [pixel]}$ | $y \text{ [pixel]}$ | $\delta v_y \text{ [pixel/s]}$ | $\delta v_x \text{ [pixel/s]}$ | $\omega \text{ [rad/s]}$ |
|-----|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 2   | 1.065           | 2.121           | 0.013           | 3.040           | 3.175           | 0.021           |
| 4   | 1.238           | 2.641           | 0.009           | 2.074           | 1.541           | 0.069           |
| 16  | 1.424           | 3.113           | 0.124           | 1.403           | 1.807           | 0.091           |

**A. White Background**

- $K = 300$
- $K = 500$
- $K = 3000$
- $K = 5000$

**B. Wood Grain Background**

- $K = 300$
- $K = 500$
- $K = 3000$
- $K = 5000$

A. On the White Background: For on the white background data, we conducted the experiment with $M = 2 \times 10^4$ and $K = 300/500$ and evaluated for every $1.5 \times 10^4$ events, i.e., for every five iterations until the 5000-th iteration for $K = 300$ and for every three iterations until the 3000-th iteration for $K = 500$.

B. On the Wood Grain Background: For on the wood grain background data, we conducted the experiment with $M = 2 \times 10^5$ and $K = 3000/5000$ and evaluated for every $1.5 \times 10^5$ events, i.e., for every five iterations until the 500-th iteration for $K = 3000$ and for every three iterations until the 300-th iteration for $K = 5000$.

### TABLE II

**Average Number of Iteration**

| $w$ | White Background | Wood Grain Background |
|-----|-----------------|------------------------|
|     | $K = 300$ | $K = 500$ | $K = 3000$ | $K = 5000$ |
| 2   | 8.65           | 13.66                  | 16.57       | 22.66       |
| 4   | 8.72           | 19.45                  | 20.37       | 27.75       |
| 8   | 11.89          | 20.64                  | 18.16       | 27.64       |
| 16  | 11.80          | 20.90                  | 18.82       | 29.83       |

A. White Background

1. $K = 300$
2. $K = 500$

B. Wood Grain Background

1. $K = 3000$
2. $K = 5000$

Fig. 5. Tracking result visualization of A: on the white background and B: on the wood grain background when $w = 16$. A-(1) is when $K = 300$, A-(2) is when $K = 500$, B-(1) is when $K = 3000$ and B-(2) is when $K = 5000$. The white box indicates the predicted position of the AR marker, the white arrows indicate the rotation and translation speed. The green box indicates the ground truth position of the AR marker and the green arrows indicate the ground truth rotation and translation speed. Positive events are shown in red and negative events are shown in blue.

The position estimation error and velocity estimation error are given by the mean square error for each value and we conduct experiments for $w = 2, 4, 8, 16$. The result of the data with the white background the wood grain pattern is shown in Table II.

In the case of on white background and on wood grain background, the wood grain background is supposed to be a more difficult problem setting for object tracking, but by comparing the result of on white background and on wood grain background. In this experiment, stable tracking was achieved regardless of the width of the Gaussian blur $w$ and we can see that the tracking performance does not deteriorate even in the case of on wood grain pattern. This shows the possibility that our gradient-based algorithm can work effectively in various situations.

### C. Number of Iterations

The average numbers of iterations for 3000 times of optimization for white background and 300 times of optimization for wood grain pattern after the second optimization are shown in Table II. The first optimization required a large number of iterations compared to the second optimization to localize the object, 1301.25 for the white background and 1684.25 for the wood grain background. As can be seen from Table II the number of optimizations increases with larger $K$, which indicates that the larger the motion of the object, it takes time to convergence.
especially suitable for mobile robotics applications. We show events sparsely, thereby realizing highly efficient processing, in contrast to conventional approaches that explicitly generate events, this method proposed implicit event generator (IEG). Unlike conventional optimizer for inference, we would like to consider the extension to 6-DoF, speed-up of inference time, and improvement of Optimizer.

a) Visualization of tracking results: Fig. 5A shows the event of AR marker on white background with tracking result in white box (position) and white arrow (velocity) and ground truth in green box (position) and green arrow (velocity). Fig. 5B shows the event of AR marker on wood grain background with tracking result and ground truth as in Fig. 5A. This shows our algorithm can track the movements with small errors for real data regardless of the size of sliding window \( K \). In addition to that, as shown in Fig. 5B, our algorithm gracefully tracks the object precisely despite the considerable amount of events generated by the background.

b) Robustness over time: The transition of estimated position \( T = (x, y, r) \) (solid line) and ground truth position (dashed line) is shown in Fig. 6 (a) shows the result of on a white background and (b) shows that of on a wood grain background. These results show that not only is the difference from the ground truth small but also that the misalignment does not increase over time. Also, the tracking performance did not depend on the white background or the wood grain background.

### D. Qualitative Evaluation

a) Visualization of tracking results: Fig. 5A shows the event of AR marker on white background with tracking result in white box (position) and white arrow (velocity) and ground truth in green box (position) and green arrow (velocity). Fig. 5B shows the event of AR marker on wood grain background with tracking result and ground truth as in Fig. 5A. This shows our algorithm can track the movements with small errors for real data regardless of the size of sliding width \( K \). In addition to that, as shown in Fig. 5B, our algorithm gracefully tracks the object precisely despite the considerable amount of events generated by the background.

b) Robustness over time: The transition of estimated position \( T = (x, y, r) \) (solid line) and ground truth position (dashed line) is shown in Fig. 6 (a) shows the result of on a white background and (b) shows that of on a wood grain background. These results show that not only is the difference from the ground truth small but also that the misalignment does not increase over time. Also, the tracking performance did not depend on whether the background was wood grain or white. This indicates that our method is not affected by a large number of events from the background, and can perform gradient optimization and find the position properly and accurately in every iteration.

### V. CONCLUSION

We proposed a novel concept of motion tracking using implicit expression. Our method estimates an object’s position and velocity by gradient-based optimization using our proposed implicit event generator (IEG). Unlike conventional approaches that explicitly generate events, this method generates events implicitly. It makes it possible to handle events sparsely, thereby realizing highly efficient processing, especially suitable for mobile robotics applications. We show that our framework could work in the practical scenario through evaluation using event data capture using a real event-based camera.

### A. Limitation and Future Work

The purpose of this study is to verify the principle of our novel concept of estimating position and velocity by gradient-based optimization using IEG as shown in Fig. 2 and for comparison with existing methods, we need to consider the extension to 6-DoF, speed-up of inference time, and improvement of Optimizer.

a) Extension to 6-DoF: For performance verification, in this paper, we have applied it to a simple 3-DoF problem and confirmed its operation. In the future, we would like to extend the idea to 6-DoF, as shown in [7], [6], [10]. The problem with extending this method to 6-DoF is that the number of variables that need to be updated increases. It is necessary to consider a new scheme for updating each variable appropriately using backpropagation, rather than simply updating each variable as in this proposed method.

b) Speed up using Look-up table: In this paper, we have only verified our concept and have not studied the reduction of inference time, which is necessary for practical applications such as AR/VR. To conduct the fast inference, we think it is an effective algorithm that reduces inference time by combining look-up tables and MLP, as proposed by Sekikawa et al. [30].

c) Optimizer: In this paper, we used SGD as the optimizer for inference. However, it is necessary to consider optimizers for speeding up and stabilizing inference, such as Momentum [31], Nesterov’s accelerated gradient [32]. In addition, we would like to consider the application of the Gauss-Newton method or the Levenberg-Marquardt method, which use the (approximate) second-order derivatives.
