Pose Estimation of Event Cameras using Transformer Neural Networks

Honghui Wu
Department of Electrical Engineering
University of Notre Dame
Email: hwu9@nd.edu

***Abstract*—Event cameras, as a bio-inspired vision sensing modality, present a promising solution for visual-to-pose estimation, with their high dynamic range, high temporal resolution, and low power consumption properties. This project introduces transformer neural networks for event camera pose estimation. Unlike existing methods, the proposed transformers process events as sequential inputs, preserving all valuable timestamp information. Initial experiments have produced promising preliminary results that the model fits a subset of the dataset well. Future work aims to scale up the training set and generalize the model.**

I. Introduction

An event camera is a bio-inspired vision sensor. It detects changes in brightness asynchronously at each pixel, mimicking the way biological eyes process visual information. Instead of recording full image frames over fixed time intervals, it outputs an event whenever a pixel detects the light intensity change reaches a threshold, resulting in high temporal resolution, low latency, and energy efficiency. This makes it highly suitable for dynamic scenes, fast-motion tracking, and applications in robotics, AR/VR, and autonomous driving. A conceptual illustration of an event camera is shown in figure 1.

Unlike traditional cameras, each pixel in an event camera responds asynchronously to changes in brightness, triggering an event $e(x, y, p, t)$ whenever the logarithmic intensity change exceeds a threshold:

$$\log I(x, y, t) - \log I(x, y, t - \Delta t) = \pm pC \tag{1}$$

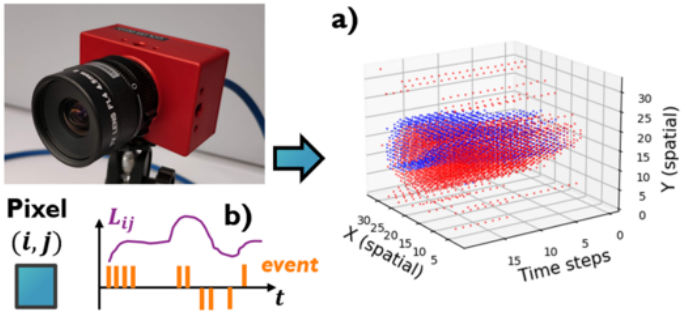


Fig. 1. Conceptual illustration of event-based vision [1]. a) The camera produces a continuous stream of spikes distributed across both space and time. b) Each pixel, x_{ij}, generates a spike whenever the change in its logarithmic light intensity, ΔL, exceeds a threshold. Positive spikes are emitted when $\Delta L > 0$ (red dots) and negative spikes are generated otherwise (blue dots).

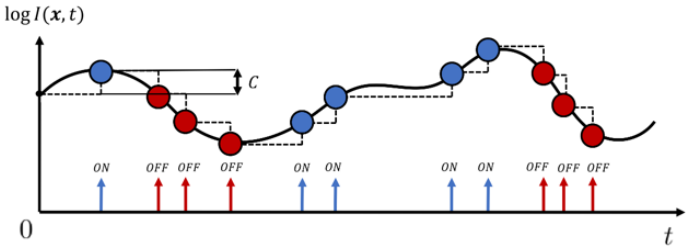


Fig. 2. Event-triggering mechanism of a pixel at an event camera [2]. An event is triggered whenever the light intensity change reaches a threshold.

where, x and y denote the pixel's position on the image plane, $p \in \{-1, +1\}$ indicates the polarity of brightness change (increase or decrease), t is the timestamp of the event, Δt is the time elapsed since the last event at the same pixel, and $C > 0$ is the threshold. See figure 2 for a visulization of the event-triggering mechanism of a pixel. The pixel-wise event-triggering mechanism enables event cameras to achieve extremely high temporal resolution (up to 10^{-6} seconds) while consuming only 10-30 mW of power. These features effectively eliminate motion blur in dynamic currents and tides, and significantly extend UAV battery life. Additionally, their logarithmic response to intensity allows event cameras to perform well in low-illumination environments.

Pose estimation of an event camera is to determine the camera's position and orientation in space over time, which is a fundamental issue of the applications mentioned above. Due to its high temporal resolution, the event camera is expected to outperform traditional cameras in the vision-to-pose estimation task. However, the challenge comes from the fact that the event camera outputs a sequence of events instead of an image frame, and thus, traditional methods such as triangulation cannot be directly applied.

Existing pose estimation methods, the most fundamental component of SLAM, using event cameras can be categorized into two classes: direct and indirect methods. Indirect methods first extract feature correspondences from the raw event data at different camera poses, such as optical flow [3], and then re-localize the camera pose using structure-from-motion techniques, like the eight-point algorithm (see [4] chapter 5). Another indirect method involves generating per-pixel depth maps from the raw data [5], followed by ego-motion estimation using the Iterative Closest Point (ICP) algorithm [6]. These

methods extend traditional camera-based pose estimation techniques (see [7]–[9]) to event cameras. Direct methods, which have become mainstream due to their real-time capabilities, include end-to-end learning and optimization-based approaches. [10] and [11] proposed Long Short-Term Memory (LSTM) networks and convolutional networks, respectively, to predict 6 degrees-of-freedom (6DOF) poses for event cameras. [12] introduced an optimization-based method that leverages the property of event cameras where the first-order approximation of (1) is the dot product of optical flow and image brightness gradient. However, while indirect methods struggle with real-time performance, existing direct methods rely on frame-based and tensor-based event data representations, which lose valuable timestamp information. Furthermore, these methods lack an attention mechanism to distinguish between static backgrounds and moving objects in dynamical underwater scenarios, leading to potential failures in such environments.

These motivate us to apply transformer neural networks to the pose estimation problem using event cameras. This is the first attempt in the literature to address event camera pose estimation using an end-to-end transformer architecture, treating events as sequential input to preserve all temporal information, and leveraging the attention mechanism to focus on the static background, ensuring robust performance in dynamic scenarios.

The remainder of this report is organized as follows: Section II outlines the project's pipeline and methodology. Section III details the experimental setup and results. Section IV concludes the current progress of this project.

II. Methodology

The pose estimation of the event camera is formulated as a supervised regression problem. The input is the event camera output between two pose labels, i.e., a set of events $\{e(x, y, t, p) : t_{k-1} < t \leq t_k\}$ cast into an appropriate data representation to be designed, where t_k is the timestamp of the pose label to be predicted and t_{k-1} is the timestamp of the previous pose label. The target is the 6 degree-of-freedom (6DOF) pose vector $\mathbf{y} = [\mathbf{p}, \mathbf{\Theta}]$, where $\mathbf{p} = [x, y, z]$ denotes the position in the Cartesian coordinate and $\mathbf{\Theta} = [\alpha, \beta, \gamma]$ denotes the Euler angles with α, β, and γ as roll, pitch, and yaw, respectively. The model set consists of transformer architecture neural networks to be designed and the loss function is chosen as mean square error:

$$Loss(\mathbf{y}, \hat{\mathbf{y}}) = \|\mathbf{w_1} \cdot (\mathbf{p} - \hat{\mathbf{p}})\|_2 + \|\mathbf{w_2} \cdot (\mathbf{\Theta} - \hat{\mathbf{\Theta}})\|_2$$

with $\hat{\mathbf{y}} = [\hat{\mathbf{p}}, \hat{\mathbf{\Theta}}]$ is the prediction by the model, and $\mathbf{w_1} = [w_{11}, w_{12}, w_{13}]$ and $\mathbf{w_2} = [w_{21}, w_{22}, w_{23}]$ are the weight vectors to be chosen as hyper-parameters.

Remark: We use Euler angles to represent the relative orientations of an event camera. The Gimbal lock effect can be avoided by leveraging the high-temporal resolution of event cameras, which ensures the Euler angles are in a small range.

A. Data Preprocessing

This part illustrates how the raw event data and the pose labels are processed before being fed into the model for training.

1) Event Representation: To retain all information from the event camera output, we treat events as sequential signals, utilizing the transformer's sequential processing capabilities. For computational efficiency, we group N consecutive events based on their timestamps into one token, applying zero padding to the final token if it contains fewer than N events.

2) Label Preprocessing: The pose labels in the raw data are provided as quaternions with respect to the global coordinate system. To predict the next pose using events between two pose labels, we must preprocess the absolute pose labels into relative poses with respect to the camera's body frame at the previous label. The following preprocessing steps are necessary for training:

Step 1: *Convert pose labels to absolute transformation matrices.* Given the k-th pose label $[\mathbf{p}_k, \mathbf{q}_k]$ where $\mathbf{p}_k = [x_k, y_k, z_k]$ denotes the position and $\mathbf{q}_k = [q_{w_k}, q_{x_k}, q_{y_k}, q_{z_k}]$ the orientation unit quaternion with q_{w_k} as the scalar part and $q_{x_k}, q_{y_k}, q_{z_k}$ defines the vector part. The absolute transformation matrix T_k from the origin of the global coordinate to the k-th pose is computed by

$$T_k = \begin{bmatrix} R_k & \mathbf{p}_k \\ 0 & 1 \end{bmatrix} \tag{2}$$

where

$$R_k = 2 \begin{bmatrix} q_{w_k}^2 + q_{x_k}^2 - 1 & q_{x_k} q_{y_k} - q_{w_k} q_{z_k} & q_{x_k} q_{z_k} - q_{w_k} q_{y_k} \\ q_{x_k} q_{y_k} + q_{w_k} q_{z_k} & q_{w_k}^2 + q_{y_k}^2 - 1 & q_{y_k} q_{z_k} - q_{w_k} q_{x_k} \\ q_{x_k} q_{z_k} - q_{w_k} q_{y_k} & q_{y_k} q_{z_k} + q_{w_k} q_{x_k} & q_{w_k}^2 + q_{z_k}^2 - 1 \end{bmatrix} \tag{3}$$

Step 2: *Compute relative transformation matrices.* The relative transformation matrix from the k-th to $k+1$-th pose is computed by

$${}^{k+1}T_k = T_{k+1} T_k^{-1} = \begin{bmatrix} {}^{k+1}R_k & {}^{k+1}\mathbf{p}_k \\ 0 & 1 \end{bmatrix} \tag{4}$$

which is equivalent to

$${}^{k+1}R_k = R_{k+1} R_k^T \tag{5}$$

$${}^{k+1}\mathbf{p}_k = (\mathbf{p}_{k+1} - \mathbf{p}_k) \cdot R_k^T \tag{6}$$

Step 3: *Convert relative transformation matrices to relative poses.* This step is to convert the relative transformation matrix in (4) to the target $\mathbf{y} = [\mathbf{p}, \mathbf{\Theta}]$ that feeds into the model for training. The position label $\mathbf{p} = {}^{k+1}\mathbf{p}_k$ from (6). The orientation label $\mathbf{\Theta} = [\alpha, \beta, \gamma]$ converted from the relative quaternions ${}^{k}\mathbf{q}_{k+1} = \mathbf{q}_k^* \mathbf{q}_{k+1}$ using the formulas:

$$\alpha = \arctan \frac{2(q_w q_x + q_y q_z)}{1 - 2(q_x q_x + q_y q_y)} \tag{7}$$

$$\beta = \begin{cases} \arcsin\big(2(q_w q_y - q_z q_x)\big), & \text{if } -1 \leq 2(q_w q_y - q_z q_x) \leq 1 \\ \arcsin 1, & \text{if } 2(q_w q_y - q_z q_x) > 1 \\ \arcsin -1, & \text{if } 2(q_w q_y - q_z q_x) < -1 \end{cases} \tag{8}$$

$$\gamma = \arctan \frac{2(q_w q_z + q_x q_y)}{1 - 2(q_y q_y + q_z q_z)} \tag{9}$$

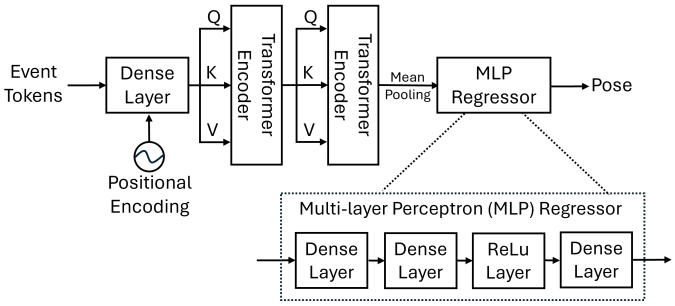
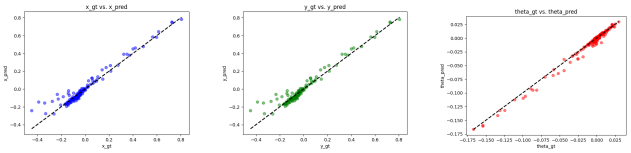
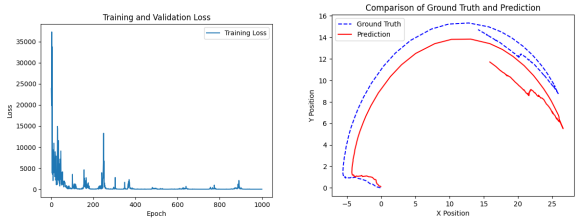


Fig. 3. Transformer-based architectures for event-based pose estimation.



(a) (b) (c)

Fig. 4. Parity Plots.(a) Prediction vs groud-truth position x. (b) Prediction vs groud-truth position y. (c) Prediction vs groud-truth rotation.



(a) (b)

Fig. 5. Training loss and .(a) Training loss (b) Predicted trajectory and ground-truth trajectory.

B. The Transformer Pose Regressor Model

This project explores transformer-based architectures for processing event data to predict pose, as shown in Fig 1. The event tokens are processed by a dense layer to transform into embeddings, with positional encoding to retain the order of events tokens. The embeddings are then processed by two self-attention transformer encoder layers, capturing the complex relationship and dependencies of the embedded event data. Mean pooling is applied to the transformer encoder's outputs to fix the size. The averaged outputs are then processed by a multi-layer perceptron regressor. The multi-layer perceptron regressor is consists of two dense layer for linear transformation, a ReLu layer for nonlinear transformation, and another dense layer to map into the dimension of pose.

III. Experiments and Results

A. Data Preprocessing

The author employs the MVSCE dataset [13], using the *outdoor_day1* driving scenario for training.

The event data and ground-truth poses are provided in ROS bag format. The author developed a Python script to preprocess the event data and pose labels. Events occurring between two pose labels are loaded into a NumPy array and sorted by timestamp. x, y, and t of events are normalized to range of $[0, 1]$. The processed event arrays are stored as PKL files for use in training. The ground-truth pose labels are in 2D for autonomous driving scenarios. The 2D pose labels are converted into format $[x, y, \gamma]$ where x, y are the absolute position, and γ is the relative rotation angle. Although these pose labels are 2D, the approach follows the same principles as 3D pose representation and can be easily generalized to 3D when using a 3D dataset.

B. Model Implementation, Training, and Evaluation

This section describes the implementation of the transformer neural network shown in Fig. 3. The transformer processes every 100 consecutive events as one token. The event tokens are passed through an MLP layer with a dimension of 512 and combined with sinusoidal positional embeddings. The model consists of two transformer encoder layers, each with 4 attention heads, a model dimension of 512, and a feedforward layer dimension of 512. The encoder's output is passed through another MLP where each layers have a dimension of 512 and transformed into a 2D pose.

The model is trained using the Adam optimizer on a NVIDIA A10 GPU from the Notre Dame CRC cluster. The model is fit on 160 data points and then scaled up to 1280 data points from *outdoor_day1* dataset from MVSCE. The batch size is set to 80, the learning rate to 0.001, and the number of epochs to 1000.

The main results are shown in Fig. 4 to 5. Fig. 4 shows the parity plots, showing how well the predictions match the ground truth of the position x, y, and rotation γ, respectively. Fig. 5 (a) shows the training loss. Fig. 5 (b) shows the predicted trajectory and ground-truth trajectory, respectively.

IV. Conclusion

This project has laid the groundwork for developing a transformer neural network for event camera pose regression, covering the process from data preprocessing to model fitting. Preliminary results are obtained that the model fits a subset of the dataset well. Future work includes scaling up the training set, generalizing the model, and evaluating the model.

References

[1] A. Safa, T. Verbelen, I. Ocket, A. Bourdoux, H. Sahli, F. Catthoor, and G. Gielen, "Fusing event-based camera and radar for slam using spiking neural networks with continual stdp learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2782–2788.

[2] B. Shao, Y. Wang, Z. Cai, and J. Zhao, "Event camera visualization," in *International Conference on Guidance, Navigation and Control*. Springer, 2022, pp. 6023–6032.

[3] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 2, pp. 407–417, 2013.

[4] Y. Ma, S. Soatto, J. Košecká, and S. Sastry, *An invitation to 3-d vision: from images to geometric models*. Springer, 2004, vol. 26.

[5] J. Furmonas, J. Liobe, and V. Barzdenas, "Analytical review of event-based camera depth estimation methods and systems," *Sensors*, vol. 22, no. 3, p. 1201, 2022.

[6] H. Bai, "ICP algorithm: Theory, practice and its SLAM-oriented taxonomy," *arXiv preprint arXiv:2206.06435*, 2022.
[7] X. Ban, H. Wang, T. Chen, Y. Wang, and Y. Xiao, "Monocular visual odometry based on depth and optical flow using deep learning," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–19, 2020.
[8] C. Chuanqi, H. Xiangyang, Z. Zhenjie, and Z. Mandan, "Monocular visual odometry based on optical flow and feature matching," in *2017 29th Chinese Control And Decision Conference (CCDC)*. IEEE, 2017, pp. 4554–4558.
[9] M. Tomono, "Robust 3D SLAM with a stereo camera based on an edge-point ICP algorithm," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 4306–4311.
[10] A. Nguyen, T.-T. Do, D. G. Caldwell, and N. G. Tsagarakis, "Real-time 6dof pose relocalization for event cameras with stacked spatial LSTM networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.
[11] A. Tabia, F. Bonardi, and S. Bouchafa-Bruneau, "Fully convolutional neural network for event camera pose estimation." in *VISIGRAPP (4: VISAPP)*, 2023, pp. 594–599.
[12] S. Klenk, M. Motzet, L. Koestler, and D. Cremers, "Deep event visual odometry," in *2024 International Conference on 3D Vision (3DV)*. IEEE, 2024, pp. 739–749.
[13] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3d perception," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2032–2039, 2018.