Attention-based Neural Networks for Event Cameras Visual-Inertial Odometry on Edge Devices

Nunzio Barone[1], Danilo Pau[2], Saverio Mascolo[1], and Luca De Cicco[1]

***Abstract*—Visual Inertial Odometry (VIO) is a fundamental component in mobile robots' navigation stacks that employ video (stereo)cameras in their perception layer. VIO complements camera data with proprioceptive feedback provided by an Inertial Measurement Unit (IMU). In this work, we explore the problem of pose estimation by combining data from an *event-based camera* and an IMU within a VIO framework. To the purpose, this paper proposes an attention-based neural network which combines stacked transformer blocks with convolutional self-attention. The proposed neural network is designed for embedded systems with limited computational and energy resources. To enable deployment on low-power microcontrollers, the model was fully quantized to 8 bit integer precision, reducing memory and latency with minimal accuracy loss. This approach demonstrates that attention-based architectures can perform real-time, high-frequency motion estimation in Tiny ML and Edge AI scenarios. Validation was conducted on STM32 microcontrollers with hardware accelerators, showing the feasibility of fully quantized neural networks for real-time embedded pose estimation.**

***Index Terms*—Visual Inertial Odometry, Event Cameras, Edge AI, Tiny ML**

I. Introduction

Visual Inertial Odometry (VIO) is a well-established technique for pose estimation in autonomous mobile robots that employ cameras in their perception stack. Typically, VIO fuses high frequency Inertial Measurement Unit (IMU) data – which provides locally accurate but drifting estimates over time – with visual cues such as landmarks or relative pixel changes, resulting in a more stable pose estimation.

However, standard frame-based cameras (30–60 fps) suffer from motion-blur, low update frequencies and limited dynamic range, which hinders high-frequency control in dynamic or low-light scenarios. Event Cameras (ECs) address these limitations through asynchronous pixel updates with microsecond latency ($\sim$10 µs) and high dynamic range. Their sparse data representation further reduces memory and bandwidth requirements [1].

As a result, event cameras have recently attracted significant interest in the VIO community, as their sensing paradigm naturally complements IMU measurements in high-dynamic and visually challenging conditions.

This has motivated a growing body of literature on event-based VIO systems, including model-based and learning-based approaches. In particular, recent advancements in deep learning, together with the availability of large-scale datasets, have paved the way for learning-based methods for VIO.

Despite their high potential in achieving improved accuracies compared to classical solutions, learning-based methods typically rely on hardware acceleration, in particular GPUs, which are known to be power hungry, expensive, and characterized by limited portability. This limits their suitability for mobile robotics platforms where power consumption is a primary concern, such as aerial robots.

The need for low-power, low-latency inference motivates the adoption of Edge AI solutions. In this context, specialized hardware accelerators, such as Neural Processing Units (NPUs), together with quantization techniques are employed to reduce both computational and memory requirements. However, constraints on power consumption and memory budget impose the use of reduced instruction sets supported by edge hardware accelerators, significantly impacting model design.

Even though learning-based event-VIO solutions have shown promising performance, their computational and energy requirements limit practical deployment on resource-constrained robotic platforms. In particular, the use of GPU-based acceleration is not suitable for the stringent power, latency, and form-factor requirements of mobile robots.

In this paper, a learning-based VIO system fusing EC and IMU data is designed in such a way that it can be deployed on low power edge devices, namely microcontrollers. In particular, we propose an attention-based neural network which combines stacked transformer blocks with convolutional self-attention. To the purpose, we design a *hardware-aware* transformer-based architecture that replaces standard operations with *ST Neural-ART* accelerator NPU compatible equivalents (e.g., convolutional self-attention) and supports full 8-bit integer quantization. Moreover, we deploy the model on the STM32N6 microcontroller, demonstrating that the on-board NPU can handle the entire inference workload with negligible CPU intervention, achieving trajectory estimation with a fraction of the energy required by GPU implementations.

II. Related work

Early studies on the use of ECs in VIO considered a pure geometric approach [2] whereas more recently event camera data have been integrated in Pose Graph Optimization frameworks [3], [4]. A new line of research focuses on deep learning data-driven approaches that should allow a more robust feature extraction that is resilient to different scenarios and calibration setups. Among these approaches, RAMP-VO [5] proposes to integrate asynchronous event data

This work has been partially supported by Regione Calabria FESR FSE 2021–2027 programme under the project MEDUSA, CUP J69I24002520005.

[1]Nunzio Barone, Saverio Mascolo, and Luca De Cicco are with the Dipartimento di Ingegneria Elettrica e dell'Informazione at Politecnico di Bari, Via Orabona 4, 70125, Bari, Italy, Emails: name.surname@poliba.it.

[2]Danilo Pau is with ST Microelectronics, Agrate, Italy, Email danilo.pau@st.com.

with synchronous image frames through massively parallel encoders and a pose forecasting module. Very recently, Deep Event Inertial Odometry (DEIO) was proposed [6]: in this work, a deep recurrent network for event data is combined with a differentiable bundle adjustment layer to couple event associations and IMU pre-integration. From a computational complexity perspective, state-of-the-art data-driven approaches typically involve high parameter counts and rely on floating-point arithmetic. This high resource demand inherently bounds their execution to high-end, power-hungry GPUs; to the best of our knowledge, none of these architectures have been successfully deployed on ultra-low-power microcontrollers.

Several VIO datasets integrate ECs and IMU data, such as MVSEC [7], DSEC [8], and TUM-VIE [9]. While MVSEC provides MoCap ground truth, it suffers from low sensor resolution. DSEC offers higher resolution but focuses on outdoor autonomous driving without MoCap support. In this work, we adopt TUM-VIE as the reference dataset due to its higher resolution, diverse indoor/outdoor scenarios, and reliable MoCap ground truth.

A critical challenge is to fit the trained models within the size and power constraints of autonomous mobile robots. While GPUs offer high throughput, their power consumption is prohibitive or not sustainable for battery-operated agents. To bridge this gap, the embedded computing landscape is exploring architectures equipped with dedicated hardware accelerators, commonly referred to as *Neural Processing Units* (NPUs) to maximize efficiency. These units employ domain-specific architectures (e.g. Convolution engines) and low-precision arithmetic (i.e. 8-bit quantization) to maximize efficiency. However, while NPU deployment is well-established for basic tasks [10], [11], its application to complex regression tasks such as 6-DoF VIO (in particular with ECs) remains unexplored, presenting challenges in terms of input representation, quantization management, and operator support which impact the deployment of specific architectures.

This paper aims to fill this gap, particularly focusing on the design of learning-based models for 6-DoF VIO integrating IMU and event camera data that are deployable on low-power NPUs.

III. Methodology

The proposed model is a lightweight auto-regressive architecture based on a Transformer encoder and a Multi Layer Perceptron (MLP) regression head. For the sake of efficient deployability, the standard attention mechanism is substituted with Convolutional Self-Attention (CSA) [12]. The output of this network is a vector of 7 elements, representing the absolute 3D translation vector (p_x, p_y, p_z) in meters and the unit quaternion (q_x, q_y, q_z, q_w) encoding the absolute orientation.

A. Input Features

The input features for the model consist of 17 values, organized as a tensor of shape (B, T, C) where B is the batch size, $T = 50$ is the sequence length, and $C = 17$ is the number of channels. These features are the concatenation of the following inputs:

- **Event Data (4 values):** including the pixel coordinates (x, y), the timestamp (t) in μs, and the polarity $(p \in \{-1, 1\})$. To align the high-frequency event stream with the IMU sampling rate (200 Hz), a random down-sampling is applied, selecting a single event in a $5\,\mathrm{ms}$ window. This reduction is justified by the high spatio-temporal redundancy in event streams, which inherently mitigates information loss.
- **Inertial Measurements (6 values):** These measurements are obtained from the 6-axis IMU (Bosch BMI160) synchronized with the camera. They consist of the linear accelerations (a_x, a_y, a_z) in $\mathrm{m/s^2}$ and the angular velocities (g_x, g_y, g_z) in $\mathrm{rad/s}$.
- **Previous State (7 values):** Due to the auto-regressive design of the perception loop, the model receives the pose estimated at the previous time step $(t-1)$ as an input to maintain temporal consistency. This vector is composed of the position (p_x, p_y, p_z) and the orientation represented as a unit quaternion (q_x, q_y, q_z, q_w). In order to reduce exposure bias, during training, noise is injected into the ground truth regressed features to simulate prediction uncertainty at inference time.

All input features are normalized using a standard scaler and, in the deployment phase, quantized as 8-bit integers to be compatible with the Neural-ART accelerator.

B. Convolutional Self-Attention

As the *attention* mechanism has proven useful in handling time-series data, we decided to employ it in our architecture. However, due to NPU limitations, matrix multiplications (*tf.matmul*) between two dynamic tensors, which are necessary to implement the attention mechanism, cannot be mapped as hardware operations. Thus, we decided to employ Convolutional Self-Attention [12] as a substitute. This implementation relies exclusively on operations such as `tf.multiply`, `Conv1D`, and `DepthwiseConv2D`. This way, the CSA block ensures full compatibility with the NPU's *CONV_ACC* and *MUL* units, enabling efficient 8-bit quantized inference with minimal latency.

C. Loss Function

The loss function is designed to minimize the error on the position and angular components. Since the model output consists of a translation vector $\mathbf{p} \in \mathbb{R}^3$ and a raw quaternion vector $\mathbf{q}_{raw} \in \mathbb{R}^4$, the loss function combines two terms weighted by constant factors. Notice that unit quaternions, useful for representing spatial rotations, belong to the unit 3-Sphere $S^3 \subset \mathbb{H}$. Since the raw network output $\mathbf{q}_{raw}$ does not necessarily lie in S^3, handling the rotational component requires normalizing the raw network output to obtain an element of S^3. For this reason, normalization with a stabilization term $\epsilon = 10^{-7}$ is applied:

$$\tilde{\mathbf{q}} = \frac{\mathbf{q}_{raw}}{\|\mathbf{q}_{raw}\|_2 + \epsilon} \tag{1}$$

This normalized $\tilde{\mathbf{q}}$ is also used as the auto-regressive feedback for the next time-step.

To solve the antipodal ambiguity of quaternions (where $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation), the Geodesic Loss [13] is adopted. The distance is computed as the minimum Euclidean norm between $\tilde{\mathbf{q}}$ and the ground truth $\mathbf{q}$ or its opposite:

$$d_q = \min(\|\mathbf{q} - \tilde{\mathbf{q}}\|_2, \|\mathbf{q} + \tilde{\mathbf{q}}\|_2) \tag{2}$$

From this, the angular error is derived:

$$\mathcal{L}_{rot} = \frac{1}{N} \sum_{i=1}^{N} \left(4 \arcsin \left(\frac{d_{q,i}}{2} \right) \right) \tag{3}$$

where N is the number of evaluated samples.

For the positional component, the Mean Squared Error (MSE) is employed:

$$\mathcal{L}_{pos} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{p}_i - \tilde{\mathbf{p}}_i\|^2 \tag{4}$$

The final total loss is a weighted sum:

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{pos} + \beta \mathcal{L}_{rot} \tag{5}$$

Despite the existence of adaptive weighting strategies [14], empirical results showed that fixed hyperparameters (α, β) provided stable convergence while maintaining lower computational overhead for the target embedded deployment.

D. Hyperparameter optimization

Due to all the possible hyperparameters combinations, the framework *Weights&Biases*[1] was employed to efficiently explore the hyperparameters space through a Bayesian Optimization, whose objective was the minimization of the root mean squared absolute translation error measured in centimeters. Table V, provided in the appendix, describes the parameters in details and highlights in bold the best values resulted from our experiments. The training was performed on an NVIDIA RTX A6000 GPU. Cumulatively, 5137 combinations were explored using parallel training agents.

IV. Edge deployment and Evaluation

Once trained, the model has to be optimized to fit in the target device. In particular, the model is quantized and optimized to adhere to memory constraints and to map its layers into hardware operations. Additionally, an evaluation is performed to assess the potential impact on the performance, obtain comparison metrics with other approaches, and metrics on the target device such as inference time and energy consumption.

A. Deployment

Before deployment, the model is quantized and optimized to respect memory constraints and map its layers into hardware operations. To do this, first we apply Post-Training Quantization (PTQ) to convert the `Float32` weights and activations into `Int8` values. This is done using a calibration dataset to minimize the quantization error. Following this stage, the model is compiled using the *ST Edge AI Developer Cloud*[2]. In this phase, optimization steps are performed in order to map operations into single hardware operations, exploiting the NPU inference as much as possible, limiting CPU usage.

[1]https://wandb.ai/

[2]https://stm32ai.st.com/st-edge-ai-developer-cloud/

TABLE I
SOTA ON TUM-VIE SEQUENCES. FOR EACH METHOD, THE FIRST ROW REPORTS ARE [°], THE SECOND ROW REPORTS ATE [CM].

Method	1d-trans	3d-trans	6DoF	desk	desk2
EVO [2]	-	-	-	-	-
	7.50	12.50	85.50	54.10	75.20
Ultimate-SLAM [16]	-	-	-	-	-
	3.90	4.70	35.30	19.50	34.10
ESVO [17]	13.47	19.20	17.59	14.56	5.86
	12.54	17.19	13.46	12.92	4.42
ESVIO [3]	-	-	-	-	-
	3.90	18.90	failed	9.00	9.50
ES-PTAM [18]	**6.02**	15.62	14.01	3.37	10.12
	1.05	8.53	10.25	2.50	7.20
IMU-Aided ESVO [19]	6.67	17.93	failed	6.95	4.32
	3.86	18.90	failed	8.99	9.47
ESVO2 [20]	6.30	**6.61**	**4.17**	**2.40**	**3.88**
	3.33	7.26	3.21	6.16	4.02
DEIO [6]	-	-	-	-	-
	0.40	**1.10**	**1.40**	**1.40**	**0.70**
Proposed	**6.00**	**6.51**	12.88	15.03	7.76
	24.64	15.62	11.94	13.30	5.33

Sources: [6], [18], [20].

TABLE II
SOTA: ATE/RMS [CM] ON STEREO DAVIS [15] SEQUENCES

Method	bin	boxes	desk	monitor
RAMP-VO [5]	3.1	5.1	3.1	4.2
Proposed	14.71	15.12	13.38	5.85

Source: [5]

B. Evaluation

To compare with other SOTA methods, we evaluate the system mainly using the TUM-VIE dataset and exclusively for the RAMP-VO method [5] the Stereo DAVIS dataset [15] is employed. In particular, we report the Absolute Trajectory Error (ATE) RMSE in centimeters and the Absolute Rotation Error (ARE) RMSE in degrees obtained using the `evo` python package[3]. Additionally, hardware performance metrics including latency, memory footprint and power consumption are profiled directly on the target board.

V. Results

A. Comparison with SOTA methods

The proposed quantized model was evaluated against state-of-the-art methods on the TUM-VIE and Stereo DAVIS datasets. The quantitative comparisons are reported in Tables I and II. The selected baselines include both geometry-based pipelines and recent deep learning approaches that operate on floating point arithmetic and high-end hardware. On TUM-VIE, the proposed approach achieves rotational accuracy (ARE) comparable to several established pipelines (ESVO, ESVO2, and ES-PTAM) despite running entirely in `Int8` precision on an embedded NPU.

[3]https://github.com/MichaelGrupp/evo

TABLE III
ABLATION WITH ALTERNATIVE INPUT CONFIGURATIONS

Metric	Event&Pose	IMU&Pose	Proposed
ARE [°]	11.28	12.75	**9.63**
ATE [cm]	14.74	**14.11**	14.17
Inference time [ms]	27.96	2.472	**0.401**

Regarding the translational component (ATE), floating-point deep models such as DEIO or ESVO2 achieve lower errors, but they rely on GPUs and significantly higher computation budgets. On the other hand, our model is designed to satisfy memory and power consumption constraints that are typical of microcontrollers. The resulting translational errors remain within the same order of magnitude for multiple sequences, showing that embedded deployment is possible. A similar trend is observed on the Stereo DAVIS dataset (Table II). RAMP-VO benefits from a more complex recurrent architecture and unconstrained resources, leading to lower ATE. The proposed model exhibits larger errors but remains competitive given its target platform. The evaluation on an out-of-distribution dataset indicates a reasonable level of generalization despite reduced numerical precision and compact architecture.

B. Input Ablation

We evaluated the contribution of each sensing modality by training three independent models (Event-only, IMU-only, and the proposed fusion), each optimized via the same hyperparameter sweep described in Section III-D. The results are reported in Table III. The fusion model achieves the lowest ARE while maintaining ATE comparable to the unimodal variants. This confirms that combining asynchronous event information with inertial measurements provides a more informative representation of motion than either modality alone.

Interestingly, the proposed fused configuration also achieves the lowest inference time. This behavior can be ascribed to the hyperparameter optimization: when richer multi-modal inputs are available, the search identifies a more compact architecture. Although this means the architectures differ across tests, precluding a strict input ablation, it demonstrates that providing complementary data allows the optimizer to find a lighter model without losing accuracy.

C. Quantization impact

We evaluated the performance gap between full-precision and quantized models using the TUM-VIE dataset. Figures 1 and 2 compare the predicted trajectories for position and orientation against the ground truth, while Figures 3 and 4 report the corresponding error employing Cumulative Distribution Functions (CDFs).

The proposed quantized model closely follows the full-precision model, with minor deviations over time. The CDFs also confirm that the distribution of the errors is only negligibly affected by the quantization: position errors are nearly uniformly distributed between 2 and 30 cm , while orientation errors range from 2° to 9°.

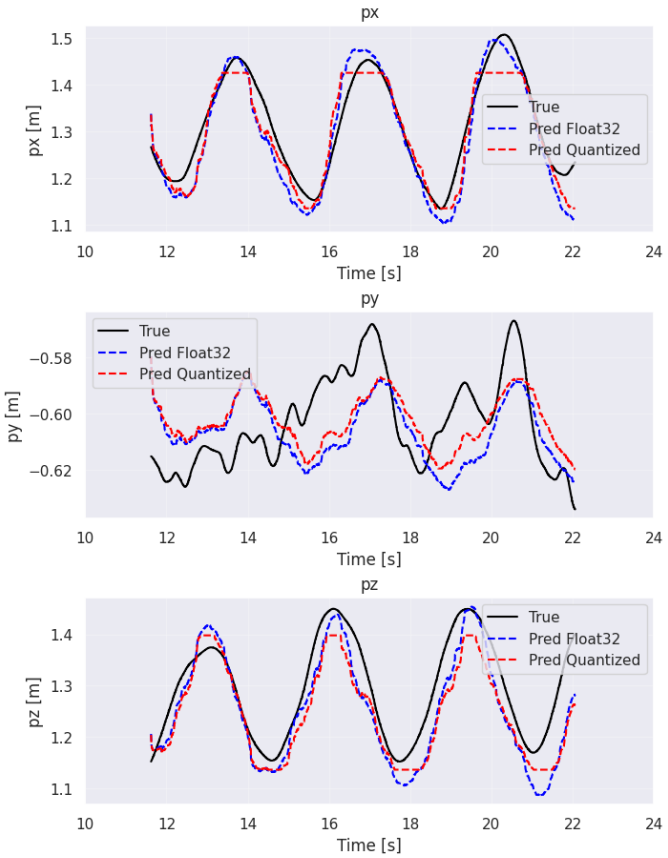

Fig. 1. Test position prediction on a TUM-VIE dataset sample with full precision and quantized model

TABLE IV
HARDWARE PERFORMANCE AND RESOURCE FOOTPRINT COMPARISON IN END-TO-END DEPLOYMENT

Metric	Float32 (Desktop GPU)	Int8 (NPU)
Weights Memory	∼468 KiB	120.61 KiB
Activations Memory	—	11.23 KiB
Inference Time	68.10 ms	0.437 ms
Net Power Consumption	2.53 W	0.098 W
Energy per Inference	138 mJ	0.14 mJ

D. Deployment efficiency

To evaluate the benefits of quantization and hardware deployment, we compared the original full-precision model executed on a Desktop GPU with the `Int8` quantized model running on the STM32N6 NPU. The main metrics are reported in Table IV.

Quantization reduces the memory footprint by approximately 74%, enabling on-chip storage within the microcontroller memory budget, while the activation footprint remains limited to few kilobytes. The most significant improvements are in terms of latency and power consumption: inference time decreases from 68.10 ms on the GPU to 0.437 ms on the NPU, and the net power consumption decreases from 2.5 W

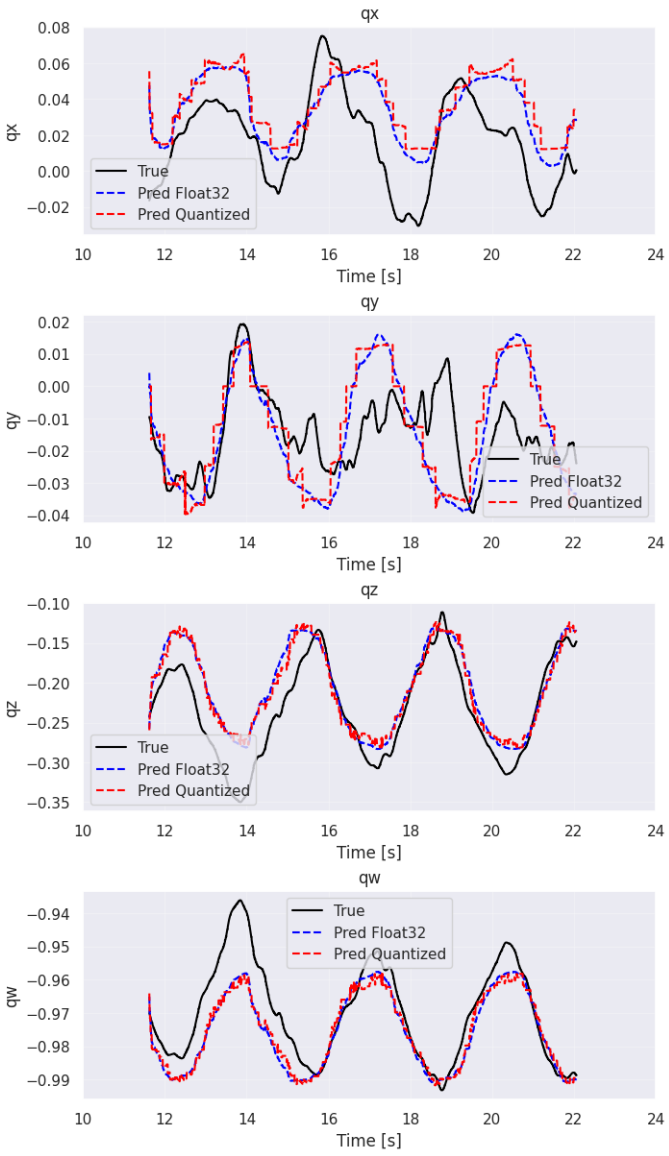

Fig. 2. Test orientation prediction on a TUM-VIE dataset sample with full precision and quantized model

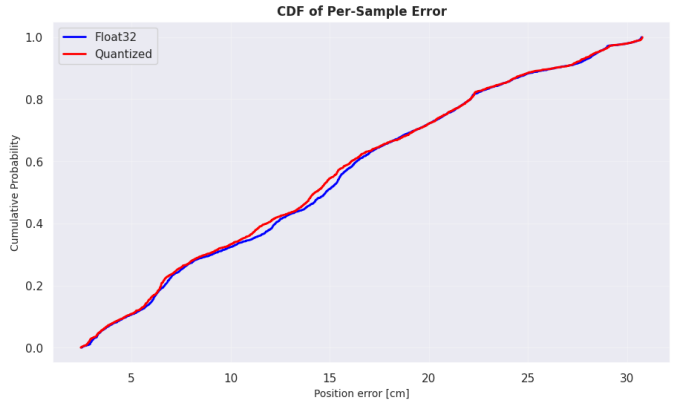

Fig. 3. Comparison of prediction error on position between full precision and quantized model

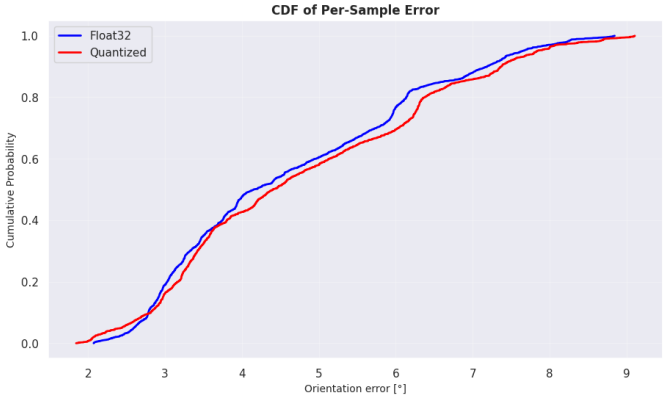

Fig. 4. Comparison of prediction error on orientation between full precision and quantized model

to 98 mW. This corresponds to an energy per inference of 0.14 mJ, which is compatible with continuous operation on battery-powered robotic platforms.

From a control systems standpoint, the 0.437 ms latency supports pose updates at rates well above typical IMU sampling frequency (e.g., 200 Hz). This prevents the perception module from becoming a computational bottleneck in the closed-loop architecture and allows real-time execution on microcontroller-class hardware.

VI. Conclusion

This work demonstrates the training and deployment of a 6-DoF VIO model integrating event camera inputs and exploiting attention mechanism, capable of being executed on an *STM32N6* by exploiting its NPU, reaching an inference time of 0.437 ms. Even though the results do not surpass the SOTA methods in terms of prediction accuracy and the model is not able to generalize on all motion types, this work demonstrates the feasibility of these applications on low-power and small-sized devices, with an estimated net consumption of less than 100 mW per inference.

Appendix

This appendix reports in Table V the complete hyperparameter search space explored during Bayesian optimization. The final configuration used for deployment is highlighted in bold.

Acknowledgements

The authors would like to thank Simona Vatinno for her valuable contribution to the experimental campaign carried out during her MSc thesis work.

References

[1] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022.

[2] H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza, "Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 593–600, 2017.

TABLE V
COMPLETE LIST OF HYPERPARAMETERS TESTED DURING THE BAYESIAN OPTIMIZATION. THE VALUES SELECTED FOR THE BEST PERFORMING MODEL ARE HIGHLIGHTED IN **BOLD** OR SPECIFIED IN THE DESCRIPTION.

Parameter	Tested Values	Description
Model Architecture		
`num_blocks`	**1**, 2, 3, 4, 5	Number of stacked Transformer encoder blocks.
`ff_dim`	17, 34, 68, **136**	Feed-Forward layer dimension (multiples of input size).
`embedding_dim`	0, **32**, 64, 128	Dimension of the initial projection layer (0 = no embedding).
`dw_kernel_size`	**5**, 10, 25, 50	Kernel size for the Convolutional Self-Attention (CSA).
`mlp_layers_dim`	32, 64, 96, 128, 256	Neurons in the 7 MLP head layers. *Best Config (L1→L7):* **128-96-128-256-128-32-96**
`pooling_type`	avg, **max**, none	Global pooling strategy before the MLP head.
`positional encoding`	True, **False**	Use of Positional Encoding.
Regularization & Noise		
`dropout`	0.0, 0.05, 0.1, 0.15, **0.2**, 0.3	Dropout rate within Transformer blocks.
`mlp_dropout`	0.1, 0.2, **0.3**, 0.4, 0.5	Dropout rate in the final MLP head.
`l2_reg`	0, 1e-4, **1e-2**	L2 weight regularization factor.
`weight_decay`	0, **1e-5**, 1e-4	Weight decay applied by the optimizer.
`noise_angle`	0.5, 2.0, 4.0, **6.0** (°)	Gaussian noise std on previous orientation input.
`noise_pos`	0.01, 0.05, **0.1** (m)	Gaussian noise std on previous position input.
`use_batch_norm`	**True**, False	Usage of Batch Normalization layers.
Training & Optimization		
`optimizer`	**Adam**, AdamW, SGD, Lion	Optimization algorithm selected.
`learning_rate`	**5e-4**, 1e-3, 5e-3	Initial learning rate.
`batch_size`	32, 64, **128**, 256	Size of training mini-batches.
`alpha`, `beta`	$\alpha \in \{\mathbf{1}, 2, 3\}, \beta \in \{\mathbf{1}, 2, 3, 4\}$	Weights balancing the multi-objective loss components.
`seq_len`	**50**, 100, 150	Input sequence length (T).

[3] P. Chen, W. Guan, and P. Lu, "Esvio: Event-based stereo visual inertial odometry," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3661–3668, 2023.

[4] P. Li, H. Yao, Z. Dai, and X. Zhu, "Asynchronous visual-inertial odometry for event cameras," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 48, pp. 651–657, 2024.

[5] R. Pellerito, M. Cannici, D. Gehrig, J. Belhadj, O. Dubois-Matra, M. Casasco, and D. Scaramuzza, "Deep visual odometry with events and frames," in *Proc. of IEEE/RSJ IROS*, 2024, pp. 8966–8973.

[6] W. Guan, F. Lin, P. Chen, and P. Lu, "Deio: Deep event inertial odometry," in *Proc. of IEEE/CVF International Conference on Computer Vision*, 2025, pp. 4606–4615.

[7] 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.

[8] M. Gehrig, W. Aarents, D. Gehrig, and D. Scaramuzza, "Dsec: A stereo event camera dataset for driving scenarios," *IEEE Robotics and Automation Letters*, 2021.

[9] S. Klenk, J. Chui, N. Demmel, and D. Cremers, "Tum-vie: The tum stereo visual-inertial event dataset," in *Proc. of IEEE/RSJ IROS*, 2021.

[10] P. Bartoli, T. Bondini, C. Veronesi, A. Giudici, N. Antonello, and F. Zappa, "End-to-end efficiency in keyword spotting: A system-level approach for embedded microcontrollers," *arXiv preprint arXiv:2509.07051*, 2025.

[11] Q. Zhu, J. Lv, and Q. Xia, "A lightweight real-time salient object detection algorithm-hardware co-optimization method for stm32 platforms," in *Int. Conference on Optoelectronic Information and Computer Engineering (OICE 2025)*, vol. 13806. SPIE, 2025, pp. 67–74.

[12] J. Yang, L. An, and J. Park, "Emulating the Attention Mechanism in Transformer Models with a Fully Convolutional Network [Online]," https://developer.nvidia.com/blog/emulating-the-attention-mechanism-in-transformer-models-with-a-fully-convolutional-network/, 2024.

[13] R. Algabri, H. Shin, A. Abdu, J.-H. Bae, and S. Lee, "WQuatNet: Wide Range Quaternion-Based Head Pose Estimation," in *Computer Vision – ACCV 2024 Workshops*, ser. Lecture Notes in Computer Science. Springer, 2025.

[14] A. Kendall and R. Cipolla, "Geometric Loss Functions for Camera Pose Regression with Deep Learning," *Computer Vision and Pattern Recognition*, 2017.

[15] Y. Zhou, G. Gallego, H. Rebecq, L. Kneip, H. Li, and D. Scaramuzza, "Semi-dense 3d reconstruction with a stereo event camera," in *Proc. of European conference on computer vision (ECCV)*, 2018, pp. 235–251.

[16] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[17] Y. Zhou, G. Gallego, and S. Shen, "Event-based stereo visual odometry," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1433–1450, 2021.

[18] S. Ghosh, V. Cavinato, and G. Gallego, "ES-PTAM: Event-Based Stereo Parallel Tracking and Mapping," *European Conference on Computer Vision (ECCV) Workshops, Milan, Italy*, 2024.

[19] J. Niu, S. Zhong, and Y. Zhou, "IMU-Aided Event-Based Stereo Visual Odometry," in *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[20] J. Niu, S. Zhong, X. Lu, S. Shen, G. Gallego, and Y. Zhou, "ESVO2: Direct Visual-Inertial Odometry with Stereo Event Cameras," *IEEE Transactions on Robotics*, 2024.