2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Event-LAB: Towards Standardized Evaluation of Neuromorphic Localization Methods

Adam D Hines*[1,2] Alejandro Fontan[1] Michael Milford[1] Tobias Fischer[1]

***Abstract*— Event-based localization research and datasets are a rapidly growing area of interest, with a tenfold increase in the cumulative total number of published papers on this topic over the past 10 years. Whilst the rapid expansion in the field is exciting, it brings with it an associated challenge: a growth in the variety of required code and package dependencies as well as data formats, making comparisons difficult and cumbersome for researchers to implement reliably. To address this challenge, we present Event-LAB: a new and unified framework for running several event-based localization methodologies across multiple datasets. Event-LAB is implemented using the Pixi package and dependency manager, that enables a single command-line installation and invocation for combinations of localization methods and datasets. To demonstrate the capabilities of the framework, we implement two common event-based localization pipelines: Visual Place Recognition (VPR) and Simultaneous Localization and Mapping (SLAM). We demonstrate the ability of the framework to systematically visualize and analyze the results of multiple methods and datasets, revealing key insights such as the association of parameters that control event collection counts and window sizes for frame generation to large variations in performance. The results and analysis demonstrate the importance of fairly comparing methodologies with consistent event image generation parameters. Our Event-LAB framework provides this ability for the research community, by contributing a streamlined workflow for easily setting up multiple conditions.**

I. Introduction

Autonomous robots that navigate their environment require localization systems to provide an estimate of their position. Systems used for visual localization often use Visual Place Recognition (VPR) [1] or Simultaneous Localization and Mapping (SLAM) [2] to determine a robot's position from sensor inputs, without the need for satellite-based positioning. In recent years, there has been interest in developing brain-inspired localization and mapping systems [3], [4] using lessons from neuroscience, often termed *neuromorphic computing* [5]. Neuromorphic localization provides the ability to develop and implement low-latency and low-energy systems for size, weight, and power (SWaP) constrained platforms, using data-rich information streams from event-based cameras [4], [6].

[1]These authors are with the QUT Centre for Robotics, School of Electrical Engineering and Robotics, Queensland University of Technology, Brisbane, QLD 4000, Australia. [2]This author is affiliated with the School of Natural Sciences, Macquarie University, Sydney, NSW 2109, Australia. * Correspondence should be addressed to: adam.hines@qut.edu.au

This work received funding from an ARC Laureate Fellowship FL210100156 to MM and an ARC Discovery Early Career Researcher Award DE240100149 to TF. The authors acknowledge continued support from the Queensland University of Technology (QUT) through the Centre for Robotics.

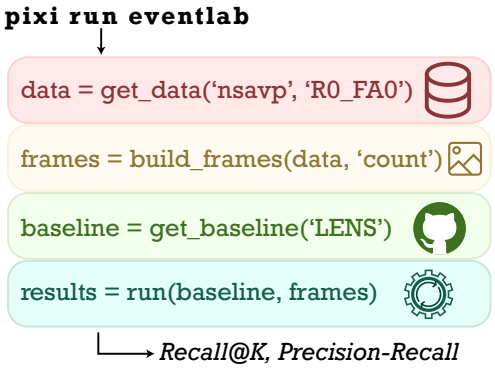

Fig. 1. Overview of the Event-LAB system. Single command-line invocation for a desired baseline method and dataset triggers a series of events. Data is downloaded from an external database and setup to a standard format. Event frames (counts or reconstructions) are generated based on a configuration file consisting of multiple parameters. Baseline methods are cloned from external repositories, including any necessary model weights and checkpoints. Finally, the generated frames and baseline method is run to produce recall and precision metrics for VPR evaluation.

Several neuromorphic systems have been developed to perform VPR [4], [7]–[11] and SLAM [12]–[15], including a series of benchmark datasets [7], [10], [16]–[20] to provide a community-based framework for the development and advancement of this field. Despite being early in this venture, however, there are already fragmentations in the implementation of baseline methods and datasets, which presents a challenge for the fair assessment of multiple comparisons. As new hardware and technology is developed and used by the research community, dataset formats and implementations differ, causing compatibility issues with downstream methods [21].

With the advent of vastly improved code package and dependency management, we propose a unified framework to perform event-based localization and mapping evaluations across multiple baselines and datasets dubbed *Event-LAB*. Using single command-line invocation, Event-LAB can be used to run multiple baseline methods with a variety of datasets in a standardized format, with evaluation metrics performed for each. This framework ensures reproducible analysis of event-based localization, streamlined workflows with minimal setup tim, and a platform for community-based contributions.

A challenge with event-based VPR is the variety of ways to define what constitutes a 'place' in localization datasets. Methods are often developed with an event count collection or time window set as a default and fixed metric, which can translate to poor performance in certain baseline methods. In this work, we describe a generalized pipeline through Event-

LAB that allows for fair, easy evaluation of baseline methods across a variety of user-defined parameters for a more complete analysis of new and emerging systems. There is a clear distinction in the performance of event-based methods when using different frame generation methods and parameters. We additionally provide an analysis of how equivalencies across different frame generation methods, event counts, and time windows can be established.

Specifically, our contributions are as follows:

1) We present Event-LAB, a comprehensive unified framework to simply run event-based localization methods with multiple datasets using single command-line invocation.
2) A thorough evaluation of available localization methods and datasets across different frame generation parameters, highlighting key advantages and disparities when comparing methods.
3) A method to find localization equivalencies across different event count or time window collections for generating reference and query datasets.

We have made the code publicly available at: `https://github.com/EventLAB-Team/Event-LAB`.

II. Related Works

This section briefly summarizes current baseline methods for event-based localization in Section II-A and datasets used to evaluate baseline methods in Section II-B. This is followed by a discussion of other frameworks that aim for standardizing benchmarking methods in Section II-C.

A. Event-based localization methods

For event-based localization, there are two typical means of representing the concept of a place from asynchronous event streams for baseline methods. The first is a simple event frame in which events are counted over a specified time window per pixel [12]. The second is a more complex process involving the reconstruction of images from event streams over a specified time window, using methods such as the popular E2VID [22], [23]. Additional event frame representation methods have been developed but have not been used specifically in localization, such as Hierarchy of Time Surfaces (HOTS) [24], Histograms of Averaged Time Surfaces (HATS) [25], and Time-Ordered Recent Event (TORE) volumes [26].

For event count frames, several methods have been proposed to make conventional VPR methods compatible with the event-based representation of a place. EventVLAD [8] trains a CNN to produce reconstructed edges from event-count frames, from which features are then extracted using NetVLAD [27]. Similarly, Event-VPR [9] examined and used voxel grids of event streams for the extraction of NetVLAD features.

In contrast to these methods, Sparse-Event-VPR [10] proposed a unique method of selecting a subset of pixels based on the variation in activity for a simple sum-of-absolute-differences (SAD) comparison with sequence matching techniques [28]. Ev-ReconNet [11] employed, for the first time, spiking neural networks to perform direct processing of event streams for edge reconstruction and NetVLAD features for energy-efficient event VPR. Most recently, Locational Encoding with Neuromorphic Systems (LENS) [4] fully integrates SNNs and a SynSense Speck neuromorphic processor for direct VPR using asynchronous event streams.

For reconstructed images, conventional and state-of-the-art VPR systems are often employed as they show impressive performance on traditional RGB images. Event reconstruction frames provide richer detail that conventional VPR methods excel with, however they come at the cost of requiring computationally expensive and time-consuming image generation. Relevant to this work, Fischer & Milford [7] devised a strategy for using ensembles of temporal windows of event accumulation with reconstructed images showing enhanced performance for large scale traverses. VPR benchmark suites are publicly available to test a variety of conventional and state-of-the-art image-based feature extractors, such as AnyLoc [29], CosPlace [30], SALAD [31], and EigenPlaces [32]. Recently, an ensembling method that used several place recognition algorithms on both frame count and reconstructed images was found to improve performance by combining the features of several algorithms [33].

For SLAM, there are multiple methods available that perform event-based pose tracking and mapping, through both monocular and stereo depth estimation. Feature-based methods such as Ultimate-SLAM (U-SLAM) [13] and PL-EVIO [14] use image-like event count frames for traditional feature detection and extraction. In [34], an event and IMU odometry (EIO) monocolar feature-based method was developed that provided real-time 6-DoF state estimation. Purely event-based monocular depth estimation was introduced by [35], with work done by [36] successfully calculating event-based optical flow asynchronously on a neuromorphic processor.

B. Event-based datasets

Several datasets for event-based localization methods exist and range from small-scale to large-scale distances using a variety of dynamic vision sensor hardware. Small-scale datasets, such as QCR-Event-VPR [10] and Fast-Slow-Event [18], were generated to test specific VPR and camera setup methods. Larger-scale VPR datasets such as Brisbane-Event-VPR [10], Novel Sensors for Autonomous Vehicle Perception (NSAVP) [19], NYC-Event-VPR [20], and DD20 [17] provide multi-environment, multi-condition traverses for testing of localization robustness. The seminal event-based dataset and simulator by Meuggler et al. [37] introduced a variety of scenarios for pose estimation, visual odometry, and SLAM. The Multi Vehicle Stereo Event Camera (MVSEC) dataset records several traverse across road and aerial environments using multiple modes of transportation [16]. Recently, the Event Camera Rotation Dataset (ECRot) was introduced with panoramas of event-based data in motorized and hand-held sequences in up to 7K resolution [15].

TABLE I

IMPLEMENTED METHODS AND DATASETS IN EVENT-LAB.

VPR Methods	
LENS	Hines et al. 2025 [4]
VPR-evaluation-methods	Berton et al. 2022 [32]
Sparse-Event-VPR	Fischer & Milford 2022 [10]
EventVLAD	Lee & Kim 2021 [8]
Ensemble-Event-VPR	Fischer & Milford 2020 [7]
VPR Datasets	
NSAVP	Carmichael et al. 2024 [19]
Fast-and-Slow	Nair et al. 2024 [18]
QCR-Event-VPR	Fischer & Milford 2022 [10]
Brisbane-Event-VPR	Fischer & Milford 2020 [7]
SLAM Methods	
Ultimate-SLAM	Rosinol et al. [13] & Rebecq et al. [41]
PL-EVIO	Guan et al. [14]
SLAM Datasets	
Event-camera dataset	Mueggler et al. [37]

C. Benchmark standardization methods

To the best of our knowledge, this is the first attempt at unifying neuromorphic localization methods and datasets for event-based systems. Previous work such as NeuroBench focuses on benchmarking algorithmic and systems level operations to compare neuromorphic architectures easily [21]. For VPR-specific benchmarking, the VPR Bench method was introduced to standardize baseline method implementation across a variety of standard datasets and evaluation metrics [38]. A more recent extension on this work by Berton et al. [32] offers recent and current state-of-the-art systems for VPR in a simple framework. VSLAM-LAB operates on a similar framework to ours that has unified visual-SLAM methods and datasets for simple and reproducible benchmarking [39], and is also implemented using Pixi.

III. METHODOLOGY

Event-LAB is implemented using Pixi, allowing for multi-platform and multi-environment package management and deployment. Pixi implements project-level environments using lockfiles that capture dependency states across supported platforms, ensuring bit-for-bit reproducibility. Additionally, it resolves dependencies using Rust, making it several times faster than conventional conda environments. As Pixi makes use of multiple package ecosystems, including conda-forge and PyPI, it provides seamless integration with several robotic architectures such as ROS Noetic (via RoboStack [40]).

A. Operation of Event-LAB

Pixi manages multiple package and dependency environments simply, allowing for baselines methods to be run within separate environments with a single `pixi.toml` file. This means that Event-LAB is able to work across Mac, Linux, and Windows operating systems and handle dependency differences seamlessly with packages like CUDA and PyTorch. Installing and running Event-LAB is as simple as:

```
# Install pixi
curl -fsSL https://pixi.sh/install.sh | sh

# Clone the Event-LAB repository
git clone
  ↪ https://github.com/EventLAB-Team/Event-LAB
  ↪ && cd Event-LAB

# Run the evaluation
pixi run eventlab lens brisbane_event sunset2
  ↪ sunset1
```

The arguments in the command line invocation use the following standardized format:

```
pixi run eventlab <baseline method> <dataset>
<reference> <query>
```

Table I highlights the currently implemented methods and datasets at the time of this work. Methods and datasets were selected on the basis of having publicly available code and data repositories. While other event-based localization methods were identified, no publicly accessible code was available to implement [9], [11], [42].

B. Baseline methods

All baseline methods (Tab. I) are implemented as an `EventBaseline` class that utilizes standardized evaluation metrics, individualized data formatting, and code execution blocks specific to the method.

```
class EventBaselineMethod(EventBaseline):
    def __init__(self, config):
        super().__init__()
        # Set config as class attribute
        self.config = config
        # Retrieve baseline method
        self.url = "https://github.com/<autho
        ↪ r>/<baseline_method>.git"
        if not os.path.exists(self.repo_path):
            clone_repo(self.url,
            ↪ destination=self.repo_path)

        # Load baseline specific configuration
        ↪ file
        self.baseline_config_path =
        ↪ './baselines/<baseline>.yaml'
        with open(self.<baseline>_config_path,
        ↪ 'r') as file:
            self.baseline_config =
            ↪ yaml.safe_load(file)

    def format_data(self, reference, query):
        # Dataset formatting for baseline

    def build_execute(self):
        # Builds executable command

    def run(self):
        # Runs the executable command

    def parse_results(self, ground_truth):
        # Performs standardized metrics

    def cleanup(self):
        # File cleanup
```

For each baseline method (Tab. I), a `baseline configuration` is used to parse specific arguments or

alter the implementation parameters. For example, in the VPR-evaluation-methods baseline [32] it is possible to alter the model parameters such as the VPR method and backbone:

```yaml
# Configuration for the vprmethods baseline
method: mixvpr
backbone: "ResNet50"
descriptors_dimension: 512
no_labels: true
```

C. Dataset management

Running the Event-LAB command triggers a series of checks and pre-processing tools, the first of which is the dataset management and formatting. Event-LAB standardizes datasets to the `.hdf5` file format, however it also provides conversion tools for `.bag` and `.txt` formats. This includes data formatting across different sensor types, *i.e.* DAVIS and Prophesee sensors. Data is downloaded from an external source based on the dataset, reference, and query choices. A configuration file is used to determine the frame generation parameters used, parsed as a list for multiple conditions to evaluate, with settings like timestamp units determined by the user.

```yaml
frame_generator: "reconstruction"
reconstruction_model: "e2vid"
# Reconstruction time in ms
timewindows: [250, 500, 750, 1000]
# Number of events per frame for reconstruction
num_events: [25000, 50000, 75000, 100000]
```

Frames can also be represented by simple event counts stored as `NumPy` arrays with or without event polarity. Frames can also be generated either a fixed time window for event collection or a maximum number of events per frame.

When a dataset is called (Tab. I), it creates an instance of an `EventDataset` class that will check the existence of the reference and query traverses with the specified parameters, download `.hdf5` raw event files, begin the frame generation process (counts or reconstruction), and generate a pseudo ground-truth. An individual dataset configuration file is also used to define download URLs, dataset formats, availability and URL to ground-truth files, and any other additional parameters that are required.

```python
class EventDataset():
    def __init__(self, config, dataset_name, sequence_name):
        super().__init__()
        # General configuration
        self.config = config
        # Name of the dataset
        self.dataset_name = dataset_name
        # Name of the sequence
        self.sequence_name = sequence_name

        # Load dataset configuration
        with open(f"./datasets/{self.dataset_name}.yaml", 'r') as file:
            self.data_config = yaml.safe_load(file)
```

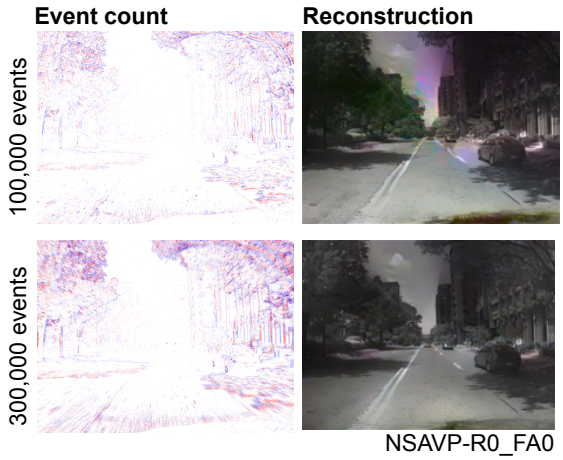

Fig. 2. Example images generated from event counts (left) or reconstruction with E2VID [22], [23] (right), with 100,000 (top) and 300,000 (bottom) events per frame specified during the image generation.

A `metadata` file is generated that contains information about the event frame generation including starting dataset and sequence names, timestamp starts and durations, and event camera resolutions. The `EventDataset` class holds all the required information to run formatting of data to the .hdf5 format.

Event images are generated either by specifying a fixed time window or the maximum number of events per frame, the latter of which will produce images of variable window length. An example of an event count and reconstructed images is shown in Fig. 2. Importantly, reconstruction-based methods like E2VID are recurrent networks that consider historic data in addition to newly incoming events, which in comparison to simple event counts provide richer detail that significantly improves performance in localization systems.

D. Pseudo ground-truth generation

Pseudo ground-truth files are auto-generated when a reference-query pair is defined in the Pixi command. The ground-truth file is a binary matrix that identifies correct reference-query matches. A ground-truth tolerance was set for each dataset based on the number of reference and query images. For small-scale, indoor datasets (QCR-Event-VPR, Fast-Slow) we set a tolerance of $\pm$ 300 msec and for the larger-scale datasets (NSAVP, Brisbane-Event-VPR), $\pm$ 10 seconds from the identified reference and query match in the pseudo ground-truth.

Matches are identified either by global positioning system (GPS) coordinates synced to event timestamps, or customized logic based on the dataset used (odometry, start and end times). We filter matches in the reverse direction of travel, and avoid matching the beginning and the end of a traverse if they overlap. References and queries can also be filtered by time to reduce the number of images in a dataset. Tolerances and filters are fully configurable parameters.

```yaml
# Signifying filter images every 60 seconds
filter_time_sec: 60
# Signifying 3 places
ground_truth_tolerance: 3
```

IV. Experimental Setup

We ran all experiments, testing, and development on an Ubuntu 24.04 system running an Nvidia RTX2080 graphics processor for CUDA acceleration. To reconstruct event frames with E2VID, we alternatively generated images in parallel using a high-performance-computing (HPC) cluster running multiple NVIDIA H100 graphics processors. Our dataset reference and query pairs for VPR evaluation were the following: Brisbane-Event-VPR dataset reference: `sunset2`, query: `sunrise` [7], NSAVP reference: `R0_FA0`, query: `R0_FS0` [19], Fast-Slow reference: `r_high1`, query: `q_high1` [18], and QCR-Event-VPR reference: `normal1`, query `fast2` [10]. To set up a reference and query dataset, we sampled places every 1000 msec and 100 msec for the long-scale (Brisbane-Event-VPR & NSAVP) and short-scale (QCR-Event-VPR & Fast-Slow) datasets, respectively. This produced on average ≈600-800 reference and query images per dataset. Importantly, this set of reference-query pairs is just an example and any combinations are possible for the available datasets implemented (Tab. I).

A. Evaluation metrics and statistics

All baseline methods and datasets are evaluated using Recall@K and the area under the precision-recall curve PR-AUC, to evaluate the overall performance and accuracy of each system. Precision and recall are calculated as follows:

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{GTP} \tag{1}$$

where TP is the number of true positives, FP is the number of false positives, and GTP is the number of ground truth positives. $Recall@K$ measures the top K matches of all references per query, measured at $K = [1, 5, 10]$.

B. Batching multiple experiments

Event-LAB provides a method for batching multiple baseline methods, datasets, and time windows by generating a bash `.sh` based on the `config.yaml` file:

```yaml
# Batch experiment parameters
batch_experiments:
  - dataset: "qcr_event"
    reference: "normal1"
    queries: ["normal2", "normal3",
    ↪  "fast2", "slow1"]
    num_events: [25000, 50000, 100000]
    frame_generator: "frames"
    frame_accumulator: "eventcount"
    baselines: ["sparse_event", "lens",
    ↪  "eventvlad"]
```

In this configuration, two main experiments will run using event-frame counts and frame reconstructions, with different baseline methods and datasets. Users can build a batch experiment of their choosing, and simply run the following to generate and execute the bash script: `pixi run batch`.

The bash script modifies the configuration file before execution of each baseline method with a dataset at the specified time window. The output of each experiment is stored to a combined spreadsheet that accumulates individual Recall@K and PR-AUC curves, as well as summary statistics of each method and baseline across frame generation parameters.

In this work, for any event reconstructed images we only used the E2VID algorithm with the E2VID checkpoint [22], [23]. All event count frames generated consider event polarity, however for baseline method formatting these are summed into a single channel. In addition to setting frames with time windows, Event-LAB allows users to match the number of events per frame from references to queries in datasets that have significant velocity differences, such as in QCR-Event-VPR [10].

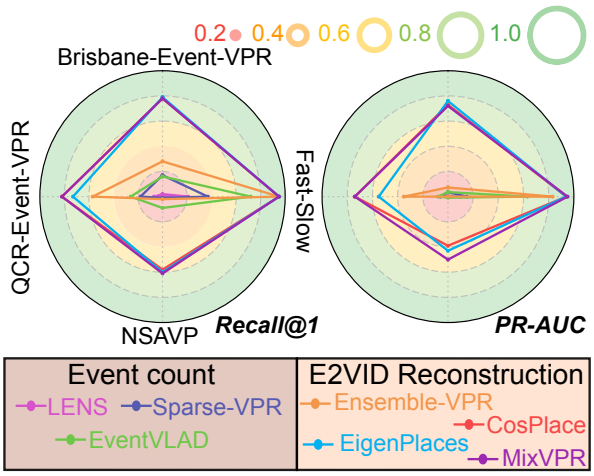

Fig. 3. Summary results from Event-LAB comparing across multiple baseline methods and datasets. All event frames were created using a fixed number of events per frame: 25,000 for QCR-Event-VPR, Brisbane-Event-VPR, and Fast-Slow, and 100,000 for NSAVP. Event counts were selected based on the camera used to collect them, a DAVIS346 and DVXplorer respectively. These event counts approximated to on average 33 msec per frame. Baseline methods that used reconstructed images performed best overall (CosPlace [30], EigenPlaces [32], MixVPR [43]).

V. Results

A. Summary results with Event-LAB

To visually compare performance of different baseline methods and datasets, we ran Event-LAB across all currently implemented methods with all datasets. A summary of the results is shown in Fig. 3. The Recall@1 and PR-AUC results highlight a diversity of outcomes for baseline methods and datasets, owing to differences in data collection, method development, and quality of reconstructed images.

Overall, the baseline methods that utilized reconstructed images performed best overall for Recall@1 and PR-AUC across the datasets tested [30], [32], [43]. Methods such as LENS and Sparse-Event-VPR did not perform as well at this smaller time window (≈33 msec) as they were both developed for longer event collection times of 1000 msec [4], [10]. For Ensemble-Event-VPR, the original implementation used a curated reference and query set that focused on finding unique features to assist in matching [7]. In Event-LAB, the focus is the ability to construct event based frames simply and easily across any time or event count domain desired. Results from Fig. 3 can easily be obtained with a single

TABLE II

FIXED TIME-WINDOW PERFORMANCE (RECALL@1 AND PR-AUC) ACROSS BASELINES FROM THE FAST-SLOW DATASET WITH R_MED1 AS THE REFERENCE AND Q_MED1 AS THE QUERY. **BOLD** IS THE BEST BASELINE METHOD FOR BOTH FRAME COUNT AND RECONSTRUCTED BASELINES. ALL TIMES ARE IN MSEC.

Baseline method	33	66	120	250	μ	PR-AUC
LENS [4]	0.05	0.16	0.53	0.77	0.38	0.29
Sparse-Event [10]	0.07	0.20	0.52	0.83	0.56	0.39
EventVLAD [8]	**0.72**	**0.84**	**0.87**	**0.95**	**0.84**	**0.76**
Ensemble-Event [7]	**0.99**	**1.0**	**1.0**	**1.0**	**0.99**	**0.99**
CosPlace [30]	0.88	0.92	0.95	**1.0**	0.94	0.84
EigenPlaces [32]	0.89	0.92	0.95	**1.0**	0.94	0.88
MixVPR [43]	0.86	0.92	0.97	**1.0**	0.94	0.93

batch script that will obtain raw data, generate images, and run baselines with a single terminal command.

B. Fixed time window performance

Creating event frames by event count generates images of varied time window length. To evaluate the performance effects of different times across fixed time window frame generation, we used the Fast-Slow dataset [18] with both frame count and reconstruction methods to evaluate Recall@1 and PR-AUC performance. The results are summarized in Table II. For the fixed time analysis, we did not subsample by time as the entire reference and query set was used to establish performance gains and losses through using different time windows.

In fixed time windows, the 33 msec condition led to results similar to those obtained in Fig. 3. For baseline methods such as LENS and Sparse-Event-VPR, performance significantly improves with increasing time window lengths, since these methods were developed and excel at processing longer event-collection durations. [4], [10].

Overall, the best performing methods were still ones that used traditional and state-of-the-art VPR feature extractors. We note however that Ensemble-Event-VPR works best in the reconstructed methods as it projects multiple windows onto the largest time, discarding all frames in between [7], leading to higher performance at lower msec time windows.

Across both event count and fixed time window evaluations, image reconstructions show the clearest advantage by being able to extract higher quality features than those available from simple event count frames.

C. Evaluating matching across collection windows

As higher time windows or event counts can lead to better performance, the question of whether equivalency from lower windows or event counts can be found to improve Recall@1 was raised. To do so, a winner-takes-all (WTA) scheme was devised such that if a percentage of the matches within a time window bin are correct and the longer-window single match for that same period is also correct, then we mark the whole period as a true positive. For example, if we consider a place with a 120 msec time window and 4×30 msec places in the same bin, if 50% of the 30

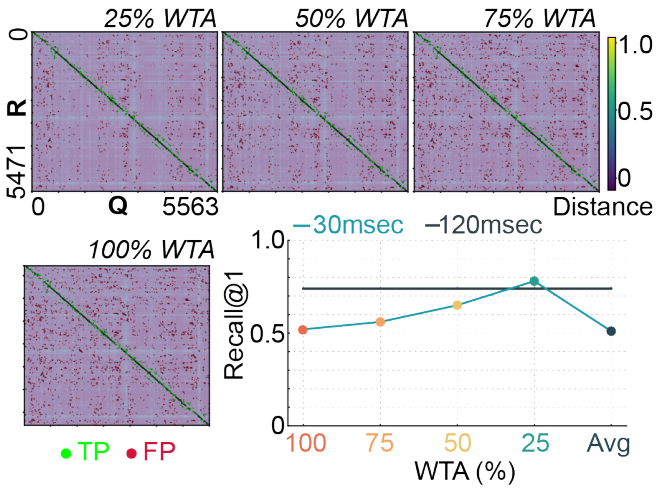

Fig. 4. Evaluation of matches from a 30 msec to 120 msec images using a winner-takes-all (WTA) approach to account for varied collections lengths. Distance matrices for the 0%, 25%, 50%, and 75% WTA condition have the true positives (TP) and false positives (FP) overlaid. The less strict 25% condition allows for a significant reduction in FP, whereas those in the 75% condition does not improve much. All WTA methods improved the Recall@1 over the 100% baseline, however averaging across references and queries in the 30 msec had no effect. The baseline method used was EventVLAD [8] and the dataset was QCR-Event-VPR [10], with `normal1` and `normal2` as the reference/query pair.

msec matches were correct and the 120 msec place was also correct, we consider all 4×30 msec places as correct. We additionally evaluated whether simply averaging over smaller time windows to match larger ones could achieve a similar outcome. Fig. 4 highlights this and shows the changes in performance using EventVLAD [8] and QCR-Event-VPR [10] as an example. All WTA implementations improved the Recall@1 performance, with the more relaxed 25% WTA condition exceeding performance of the 120 msec condition (Fig. 4. Averaging across 30 msec time windows failed to improve Recall@1 at all.

Quantitatively, the equivalent is also true for generating frames through event counts, with a 14% increase in Recall@1 applying a 50% WTA rule to event frames generated with 25,000 events when compared to frames generated with 100,000 events. Similar to using ground truth tolerances, a WTA approach for event based frames could be considered a means of easing overly strict matching criteria.

D. Event-based Simultaneous Localization and Mapping

In order to evaluate SLAM baseline methods for Event-LAB, we measured the Root Mean Squared Error Absolute Trajectory Error (RMSE-ATE) across various motions and scenes from the Event-Camera Dataset and Simulator [37]. The estimated and ground-truth trajectories were aligned with a 6-DoF transformation in $\mathbf{SE}(3)$, using the entire trajectory. Fig. 5 summarizes the main results, showing the RMSE-ATE and accuracy across five different scenes. PL-EVIO [14] was found to perform the best with the lowest degree of trajectory error across all scenes. Fig. 6 shows example trajectories from the best performing scene (Poster).

Both U-SLAM and PL-EVIO use event frames to perform feature extraction and tracking to estimate trajectories, addi-

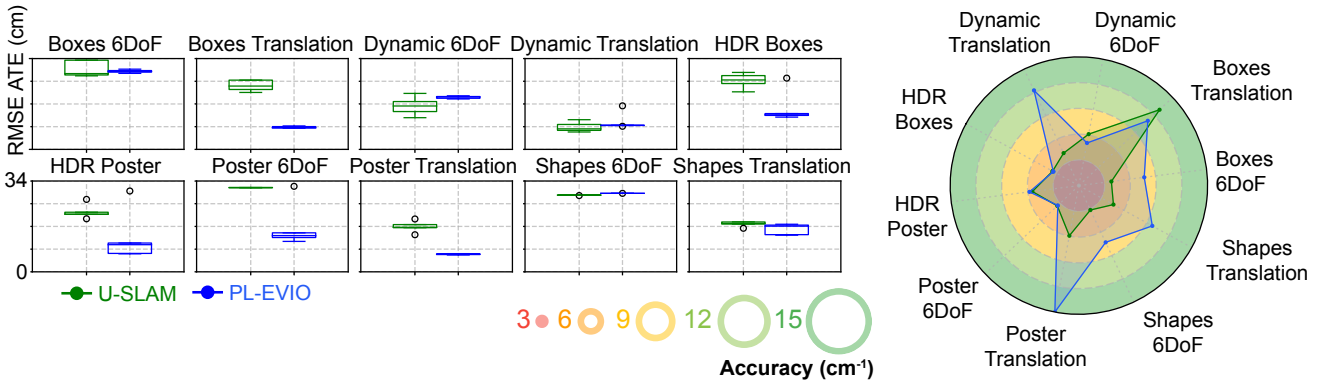

Fig. 5. Metrics for the SLAM baseline methods, U-SLAM [13] and PL-EVIO [14], using 5 different scenes and motions from the Event-Camera Dataset and Simulator [37]. The box plots (left) show the RMSE-ATE in cm across 5 individual trials. The radar plot (right) shows the Accuracy in cm^{-1}, which is the inverse of RMSE-ATE.

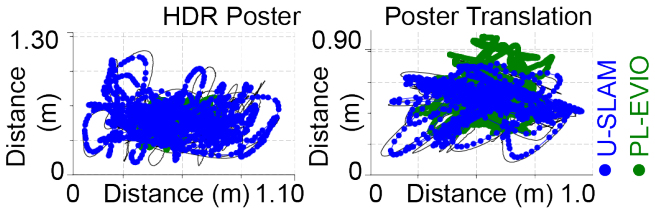

Fig. 6. Example SLAM trajectories from the Poster scene from the Event-Camera Dataset and Simulator [37]. Black is the ground truth trajectory.

TABLE III
SETUP TIME FOR EVENT-LAB FOR VARIOUS BASELINES AND DATASETS

Method	Dataset	Query	Ref.	Type	Events	Time (mm:ss)
Sparse [10]	QCR [10]	fast1	normal1	Count	25 000	01:01
MixVPR [43]	Fast-Slow [18]	q_low1	r_low1	Recons.	25 000	06:35
EventVLAD [8]	NSAVP [19]	R0_RN0	R1_DA0	Count	250 000	47:42

tionally integrating traditional RGB frame based images and inertial measurement unit (IMU) data for enhanced performance. Both SLAM methods ran in real time and covered 100% of the recording without any failures. The promising initial results with the simple Event-LAB implementation provide the platform for integration with additional datasets and methods in the future.

E. Setup time and metrics

An important component of Event-LAB is the ease of setup and the rapid formatting and output of recall and precision data. To evaluate benchmarking capability and times, we set up three independent experiments and measured the time taken to 1) obtain the data files and baseline method project repositories, 2) generate the event frames, and 3) run and obtain the results, including the installation of Pixi and environment building. The details and results are summarized in Table III. Simple VPR methods and small-scale datasets like Sparse-Event-VPR [10] and the Fast-Slow dataset [18] are very quick to download and run, whereas larger-scale and computationally expensive processes like EventVLAD [8] and NSAVP [19] take more time to generate frames and run the baseline method.

We note that with high performance computing (HPC) clusters, this process becomes entirely parallelizable, running multiple baselines and event frame generation simultaneously. The main bottleneck in the Event-LAB process is the event frame generation, particularly for image reconstruction, and in some cases downloading data files from host servers. Once frames have been generated and stored, the runtime for experiments significantly decreases.

VI. DISCUSSION AND CONCLUSIONS

We present Event-LAB, a simple and reproducible means to run and contribute event-based localization methods for the greater research community. The goal was to provide a community-based tool that continues to have new methods and datasets added and to create a platform allowing for simple additions. Event-LAB will expand to additional localization and navigational methods such optical flow and classification tasks such as object detection and recognition. Here, we focused only on accuracy metrics however we do note that method latencies and power consumption are also important considerations in event-based systems, but was outside the scope for investigation and discussion.

In this work, we identify several potential future directions. We propose a new metric for averaging the varied performance across different event frame collection parameters through a winner-takes-all strategy. Future work will focus on developing a robust matching analysis pipeline to assist in reducing the computational overhead of event-based localization methods by minimizing the generation of thousands of low-event-count or short-time-window places. An additional focus in this regard is to develop the ability for asynchronous localization streaming, without the dependence on the creation of event frames for evaluation. We will additionally focus on including SNN-based localization pipelines for asynchronous processing, streaming directly from event files. This future direction directly addresses a common issue in the literature for event-based methodologies that rely on conventional representations of event streams for localization tasks. Finally, we propose that Event-LAB could in the future be used to fuse features and output from multiple localization methods for accurate, event-based navigation pipelines.

REFERENCES

[1] C. Masone and B. Caputo, "A survey on deep visual place recognition," *IEEE Access*, vol. 9, p. 19516–19547, 2021.

[2] J. A. Placed, J. Strader, H. Carrillo *et al.*, "A survey on active simultaneous localization and mapping: State of the art and new frontiers," *Transactions on Robotics*, vol. 39, no. 3, p. 1686–1705, 2023.

[3] F. Yu, J. Shang, Y. Hu *et al.*, "NeuroSLAM: a brain-inspired SLAM system for 3D environments," *Biological Cybernetics*, vol. 113, no. 5, pp. 515–545, 2019.

[4] A. D. Hines, M. Milford, and T. Fischer, "A compact neuromorphic system for ultra–energy-efficient, on-device robot localization," *Science Robotics*, vol. 10, no. 103, p. eads3968, 2025.

[5] R. Roy and R. Shajan, "Survey of neuromorphic computing and neural networks in hardware," *International Journal of Scientific Development and Research*, vol. 8, no. 3, pp. 1462–1465, 2023.

[6] G. Gallego, T. Delbruck, G. Orchard *et al.*, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 44, no. 01, p. 154–180, 2022.

[7] T. Fischer and M. Milford, "Event-based visual place recognition with ensembles of temporal windows," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6924–6931, 2020.

[8] A. J. Lee and A. Kim, "EventVLAD: Visual place recognition with reconstructed edges from event cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 2247–2252.

[9] D. Kong, Z. Fang, K. Hou *et al.*, "Event-VPR: End-to-end weakly supervised deep network architecture for visual place recognition using event-based vision sensor," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–18, 2022.

[10] T. Fischer and M. Milford, "How many events do you need? event-based visual place recognition using sparse but varying pixels," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 275–12 282, 2022.

[11] H. Lee and H. Hwang, "Ev-ReconNet: Visual place recognition using event camera with spiking neural networks," *IEEE Sensors Journal*, vol. 23, no. 17, pp. 20 390–20 399, 2023.

[12] H. Rebecq, T. Horstschaefer, G. Gallego *et al.*, "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.

[13] A. R. Vidal, H. Rebecq, T. Horstschaefer *et al.*, "Ultimate SLAM? combining events, images, and imu for robust visual slam in hdr and high speed scenarios," in *IEEE Robotics and Automation Letters*, vol. 3, 2018, pp. 994–1001.

[14] W. Guan, P. Chen, Y. Xie *et al.*, "PL-EVIO: Robust monocular event-based visual inertial odometry with point and line features," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 4, pp. 6277–6293, 2023.

[15] S. Guo and G. Gallego, "CMax-SLAM: Event-based rotational-motion bundle adjustment and SLAM system using contrast maximization," *IEEE Transactions on Robotics*, vol. 40, pp. 2442–2461, 2024.

[16] A. Z. Zhu, D. Thakur, T. Özaslan *et al.*, "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.

[17] Y. Hu, J. Binas, D. Neil *et al.*, "DDD20 end-to-end event camera driving dataset: Fusing frames and events with deep learning for improved steering prediction," in *IEEE International Conference on Intelligent Transportation Systems*, 2020, pp. 1–6.

[18] G. B. Nair, M. Milford, and T. Fischer, "Enhancing visual place recognition via fast and slow adaptive biasing in event cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024, pp. 3356–3363.

[19] S. Carmichael, A. Buchan, M. Ramanagopal *et al.*, "Dataset and benchmark: Novel sensors for autonomous vehicle perception," 2024.

[20] T. Pan, J. He, C. Chen *et al.*, "NYC-Event-VPR: A large-scale high-resolution event-based visual place recognition dataset in dense urban environments," in *IEEE International Conference on Robotics and Automation*, 2025.

[21] J. Yik, K. Van den Berghe, D. den Blanken *et al.*, "The neurobench framework for benchmarking neuromorphic computing algorithms and systems," *Nature Communications*, vol. 16, no. 1, p. 1545, 2025.

[22] H. Rebecq, R. Ranftl, V. Koltun *et al.*, "High speed and high dynamic range video with an event camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[23] H. Rebecq, R. Ranftl, V. Koltun *et al.*, "Events-to-video: Bringing modern computer vision to event cameras," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[24] X. Lagorce, G. Orchard, F. Galluppi *et al.*, "HOTS: a hierarchy of event-based time-surfaces for pattern recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1346–1359, 2017.

[25] A. Sironi, M. Brambilla, N. Bourdis *et al.*, "Hats: Histograms of averaged time surfaces for robust event-based object classification," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1731–1740.

[26] R. W. Baldwin, R. Liu, M. Almatrafi *et al.*, "Time-ordered recent event (TORE) volumes for event cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 2519–2532, 2023.

[27] R. Arandjelović, P. Gronat, A. Torii *et al.*, "NetVLAD: CNN architecture for weakly supervised place recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1437–1451, 2018.

[28] S. Garg, M. Vankadari, and M. Milford, "Seqmatchnet: Contrastive learning with sequence matching for place recognition & relocalization," in *Conference on Robot Learning*, vol. 164, 2022, pp. 429–443.

[29] N. Keetha, A. Mishra, J. Karhade *et al.*, "Anyloc: Towards universal visual place recognition," *IEEE Robotics and Automation Letters*, pp. 1286–1293, 2023.

[30] G. Berton, C. Masone, and B. Caputo, "Rethinking visual geo-localization for large-scale applications," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4878–4888.

[31] S. Izquierdo and J. Civera, "Optimal transport aggregation for visual place recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[32] G. Berton, G. Trivigno, B. Caputo *et al.*, "Eigenplaces: Training viewpoint robust models for visual place recognition," in *IEEE/CVF International Conference on Computer Vision*, 2023, pp. 11 080–11 090.

[33] T. Joseph, T. Fischer, and M. Milford, "Ensemble-based event camera place recognition under varying illumination," in *arXiv*, 2025.

[34] W. Guan and P. Lu, "Monocular event visual inertial odometry based on event-corner using sliding windows graph-based optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 2438–2445.

[35] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," in *European Conference on Computer Vision*, 2016, pp. 349–364.

[36] S. Chiavazza, S. M. Meyer, and Y. Sandamirskaya, "Low-latency monocular depth estimation using event timing on neuromorphic hardware," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023, pp. 4071–4080.

[37] E. Mueggler, H. Rebecq, G. Gallego *et al.*, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, 2017.

[38] M. Zaffar, S. Garg, M. Milford *et al.*, "VPR-Bench: An open-source visual place recognition evaluation framework with quantifiable viewpoint and appearance change," *International Journal of Computer Vision*, pp. 1–39, 2021.

[39] A. Fontan, T. Fischer, J. Civera *et al.*, "VSLAM-LAB: A comprehensive framework for visual slam methods and datasets," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2025.

[40] T. Fischer, W. Vollprecht, S. Traversaro *et al.*, "A RoboStack Tutorial: Using the Robot Operating System Alongside the Conda and Jupyter Data Science Ecosystems," *IEEE Robotics & Automation Magazine*, vol. 29, no. 2, pp. 65–74, 12 2022.

[41] H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization," in *British Machine Vision Conference*, 2017.

[42] X. Ji, J. Wei, Y. Wang *et al.*, "Cross-modal place recognition in image databases using event-based sensors," in *arXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2307.01047

[43] A. Ali-bey, B. Chaib-draa, and P. Giguère, "MixVPR: Feature mixing for visual place recognition," in *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 2998–3007.