VSLAM-LAB: A Comprehensive Framework for Visual SLAM Methods and Datasets

Alejandro Fontan[†] Tobias Fischer[†] Javier Civera[‡] Michael Milford[†]

https://github.com/alejandrofontan/VSLAM-LAB

***Abstract*— Visual Simultaneous Localization and Mapping (VSLAM) research faces significant challenges due to fragmented toolchains, complex system configurations, and inconsistent evaluation methodologies. To address these issues, we present VSLAM-LAB, a unified framework designed to streamline the development, evaluation, and deployment of VSLAM systems. VSLAM-LAB simplifies the entire workflow by enabling seamless compilation and configuration of VSLAM algorithms, automated dataset downloading and preprocessing, and standardized experiment design, execution, and evaluation. All of these features are accessible through a single command-line interface. The framework supports a wide range of VSLAM systems and datasets, offering broad compatibility and extendability while promoting reproducibility through consistent evaluation metrics and analysis tools. By reducing implementation complexity and minimizing configuration overhead, VSLAM-LAB empowers researchers to focus on advancing VSLAM methodologies and accelerates progress toward scalable, real-world solutions. We demonstrate the ease with which user-relevant benchmarks can be created: here, we introduce difficulty-level-based categories, but one could envision environment-specific or condition-specific categories.**

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) is a widely investigated problem within the computer vision and robotic communities [1]. It aims to estimate the trajectory of a mobile robot while concurrently constructing a representation of the environment. Visual SLAM (VSLAM) refers to the modality that uses RGB video as the main sensory input.

Despite significant research over the years, benchmarking VSLAM implementations remains highly challenging. On the one hand, VSLAM performance is highly dependent on the characteristics of the scene (*e.g.*, its texture and depth distribution) and the camera motion (*e.g.*, translation and linear and angular velocity profiles). This should motivate benchmarks in the widest variety of datasets and including as many baselines as possible. However, the lack of standardization in datasets and implementations is a key barrier in doing this. Benchmarking is now a time-consuming process for VSLAM researchers and practitioners and, as a result, experimental evaluations are typically limited to a small subset of sequences and baselines.

[†]AF, TF and MM are with the QUT Centre for Robotics, School of Electrical Engineering and Robotics, Queensland University of Technology, Brisbane, QLD 4000, Australia. [‡]JC is with the I3A, Universidad de Zaragoza, Spain. This research was partially supported by funding from ARC Laureate Fellowship FL210100156 to MM and ARC DECRA Fellowship DE240100149 to TF. The authors acknowledge continued support from the Queensland University of Technology (QUT) through the Centre for Robotics. Corresponding author email: alejandro.fontan@qut.edu.au

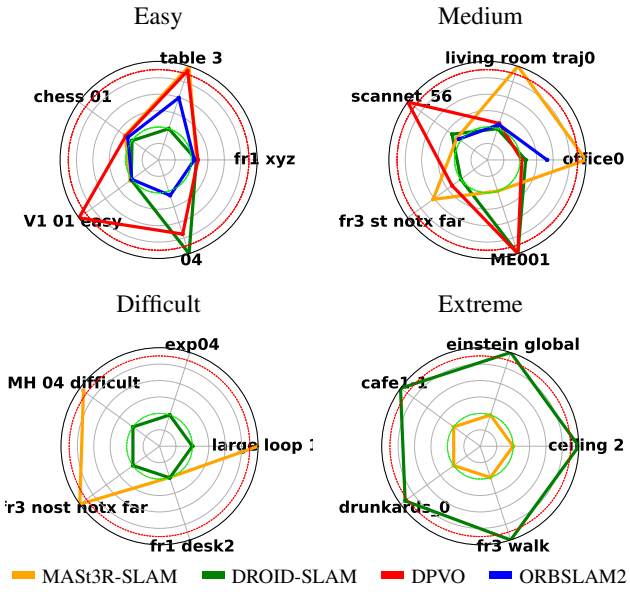


Fig. 1: Comparison of 4 state-of-the-art Visual SLAM methods across 20 sequences from 12 diverse datasets. Our evaluation highlights the strengths and weaknesses of each system, providing insights to guide future VSLAM research. Notably, the VSLAM-LAB framework ensures these experiments can be reproduced seamlessly with negligible time overhead and modified with minimal implementation effort. **Smaller area is better**.

Fragmentation is a major challenge in VSLAM research, with SLAM pipelines relying on disparate tools. Datasets differ in structure, camera calibration models, and ground truth formats, complicating direct comparisons. Inconsistencies in datasets, evaluation metrics, and benchmarking protocols further hinder reproducibility. Determining how trajectory ground truth is obtained—whether from independent motion-tracking systems like GPS or image-based reconstructions via structure-from-motion—remains time-consuming, as does automating parameter tuning [2]. Additionally, many VSLAM implementations are not easily shareable or reusable and require complex dependencies and custom configurations, slowing research progress, limiting reproducibility, hindering comparisons, and complicating deployment and integration.

An underlying issue behind these challenges is that researchers often struggle with software complexities, writing code that addresses immediate needs but is difficult to reuse or reproduce. This results in significant time spent re-implementing evaluation scripts and formatting datasets, diverting focus from advancing VSLAM research.

To address these challenges, we introduce **VSLAM-**

LAB—a powerful tool that streamlines benchmarking by automating key processes, including dataset formatting, method execution, experiment configuration, and trajectory evaluation. It enables effortless comparison of state-of-the-art VSLAM systems on standard and alternative datasets with minimal implementation effort while significantly reducing the time needed to integrate new methods and datasets. To ensure reproducibility, VSLAM-LAB employs a configuration file that standardizes the entire pipeline—from C++ and Python package installation to data formatting, execution, and evaluation—allowing experiments to be consistently replicated.

Here, we demonstrate the capabilities of **VSLAM-LAB** by evaluating 4 state-of-the-art VSLAM pipelines across 20 sequences from 12 diverse datasets. Furthermore, we introduce the first benchmark that integrates challenges from all these datasets, encompassing a wide range of scenarios, including indoor and outdoor environments, varying difficulty levels, real and synthetic data, and scenes with dynamic objects. VSLAM-LAB enables researchers to effortlessly create custom benchmarks tailored to specific research questions. While we showcase difficulty-level categorization as one example (see Figure 1), VSLAM-LAB's flexible architecture supports many other classification schemes—such as environment type, motion patterns, or lighting conditions—without requiring significant implementation effort.

II. RELATED WORK

In this section, we first briefly summarize the most popular SLAM datasets in Section II-A, followed by other works that aim toward standardised benchmarking in SLAM in Section II-B. We also review a range of localization frameworks that share our aims of standardizing dataset formatting, method execution and evaluation in Section II-C.

A. Visual SLAM Datasets

The TUM RGB-D benchmark [3] by Sturm *et al*. provides a comprehensive dataset of 39 sequences captured with a Microsoft Kinect in office and industrial environments. The benchmark includes synchronized color and depth images at 640×480 resolution with ground truth camera poses from a motion capture system.

The ETH3D SLAM benchmark by Schöps *et al*. [4], [5] features 56 training and 35 test datasets recorded with a custom camera rig. Similar to the TUM RGB-D benchmark, the ground truth annotations were obtained using a motion capture system, with some additional sequences whose ground truth was approximated via Structure-from-Motion. The ETH3D SLAM benchmark investigates performance impacts due to rolling shutter and geometric distortions in cameras. Interestingly, their benchmark contains sequences without public ground truth to avoid overfitting to specific datasets/sequences.

The KITTI vision benchmark suite [6] represents a pioneering effort in autonomous driving datasets, featuring stereo, optical flow, visual odometry/SLAM, and 3D object detection tasks. Captured using a sensor-rich autonomous driving platform, the benchmark highlighted the increased difficulty of real-world scenarios compared to controlled lab conditions.

Engel *et al*. [7], [8] presented the TUM monoVO dataset, designed to offer photometrically calibrated image sequences recorded in a variety of indoor and outdoor settings. The dataset prioritizes camera motion analysis, incorporating a significant loop closure at the end of each sequence to assess drift accumulation without relying on complete ground truth trajectories. Visual odometry (VO) performance is evaluated using the alignment error, a metric that quantifies drift across the entire sequence.

B. Visual SLAM Benchmarking Suites

The pySLAM framework [9] provides a Python implementation of visual SLAM supporting multiple camera configurations. It offers various local features, loop closing methods, and a volumetric reconstruction pipeline. While pySLAM provides implementation tools for SLAM algorithms, including a large range of local feature detectors and descriptors, VSLAM-LAB differs by focusing on the comprehensive evaluation ecosystem, addressing the fragmentation in evaluation methodologies that has hindered objective comparison of SLAM approaches. PySLAM is also limited to currently just four datasets.

The evo Python package [10] provides functionality for comparing and evaluation trajectory outputs from odometry and SLAM algorithms. It supports multiple trajectory formats including TUM, KITTI, and EuRoC MAV, as well as ROS and ROS2 bagfiles. While evo provides valuable tools for trajectory analysis, it primarily focuses on the evaluation stage.

SLAMBench [11]–[14] may be the closest to our work. It is, however, not maintained currently, with the most recent VSLAM baselines (*e.g*., DPVO, DROID-SLAM or MASt3R-SLAM) being missing. The SLAM Hive Suite [15] shares the same target. It operates using Docker containers which enables parallel benchmarking in the cloud. It does not, however, address standardization, and hence still may incur programming overhead for adapting systems, datasets and benchmarks. In this sense, it is complementary to our work, and utilizing it downstream to parallelize different runs of our VSLAM-LAB would be interesting as future work.

While these benchmarks offer valuable insights, they primarily focus on isolated use cases, limiting their generalizability to broader SLAM applications. VSLAM-LAB overcomes this issue by offering a diverse collection of datasets across indoor, outdoor, synthetic, and challenging environments. It also provides a standardized evaluation process, enabling researchers to compare SLAM methods more systematically and under various real-world conditions.

C. Localization Frameworks

Various localization benchmark efforts share our goal of standardizing evaluation methodologies, but target different technical challenges within the broader visual perception domain.

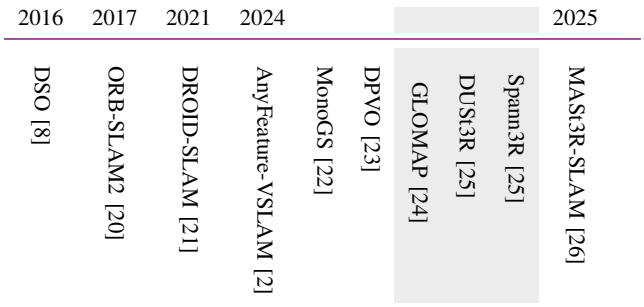


TABLE I: **VSLAM-LAB Methods**: Visual SLAM & Multi-view reconstruction methods (gray columns) for monocular cameras contained in VSLAM-LAB.

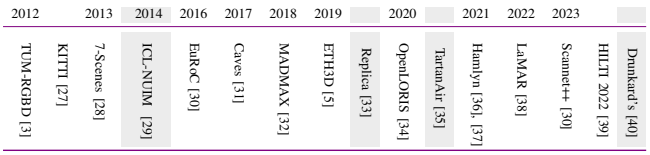


TABLE II: **VSLAM-LAB Datasets:** The white columns correspond to datasets contained in our VSLAM-LAB capturing real-world data, while the gray columns contain synthetic datasets.

The deep visual geo-localization benchmark [16] provides a framework for researchers to build, train, and test various geo-localization architectures with modular components. This framework emphasizes performance metrics such as recall@N alongside system requirements including execution time and memory consumption. While sharing our objective of standardizing evaluation protocols, this work focuses specifically on geo-localization rather than the full SLAM pipeline that VSLAM-LAB addresses.

Complementary to the deep visual geo-localization benchmark [16], the visual localization benchmark [17] provides several benchmark datasets for 6-DoF pose estimation. The datasets cover a range of appearance changes, including seasonal, viewpoint and illumination (dawn, day, sunset, night) conditions. Interestingly, the query poses are withheld to avoid overfitting of methods on specific datasets.

Several works provide comprehensive frameworks for evaluation visual place recognition (VPR), which is often used as a loop closure component for SLAM. VPR-Bench [18] offers 12 integrated datasets and benchmarks 10 VPR techniques, with particular emphasis on quantifying viewpoint and illumination invariance. The VPR methods evaluation codebase [19] provides access to many state-of-the-art VPR techniques including the model definitions and weights, and enables researchers to compare them fairly. It complements the VPR Datasets Downloader repository by the same authors, which similarly to VSLAM-LAB provides a unified way to access datasets in a standardized way, in this case for VPR datasets [16].

Despite progress in SLAM research, localization benchmarking remains a largely manual process, increasing the risk of inconsistencies and reducing reproducibility. VSLAM-LAB offers a fully integrated solution that standardizes localization evaluation, ensuring that different methods can be fairly and efficiently compared while reducing the complexity of setting up and running experiments.

III. VSLAM-LAB

VSLAM-LAB provides a flexible framework for conducting customizable experiments with minimal configuration. It offers an automated pipeline for method compilation and installation (Section III-A), supports seamless integration of various methods (Section III-B) with many datasets (Section III-C), dataset acquisition and preprocessing, experiment execution (Section III-D), and comprehensive evaluation (Section III-E).

A. Dependency management

A key aspect of VSLAM-LAB's implementation is the use of *Pixi* (`https://pixi.sh`) for dependency management. Pixi is a multi-platform and multi-language package management tool that extends the popular Conda ecosystem. In addition to seamlessly installing compilers and low-level libraries like CUDA, it provides binary packages for many popular tools like PyTorch, OpenCV, Open3D, Scikit-Learn and many more.

VSLAM-LAB specifies separate environments for each of the VSLAM methods listed in the next section, and the environments' dependencies are listed in the *pixi.toml* manifest file. Importantly for reproducibility, pixi automatically creates lock files for all dependencies, which ensures the creation of consistent environments, guaranteeing reproducibility across different setups by enforcing identical package versions and dependencies. Pixi makes running an experiment on a new machine as easy as:

```
# install pixi
curl -fsSL https://pixi.sh/install.sh | bash
# clone repo
git clone https://github.com/alejandrofontan/
    VSLAM-LAB.git
# run experiment, e.g. our demo
cd VSLAM-LAB && pixi run demo
```

B. Visual SLAM Methods

Table I (top) lists the Visual SLAM methods currently integrated into VSLAM-LAB. We include four dense methods, *i.e.*, MASt3R-SLAM [26], DPVO [23], MonoGS [22], and DROID-SLAM [21], and three sparse visual SLAM methods, *i.e.*, AnyFeature-VSLAM [2], ORB-SLAM2 [20], and DSO [8]. Among these, MASt3R-SLAM is the only approach capable of handling sequences captured with uncalibrated cameras.

A common strategy for evaluating Visual SLAM methods in the absence of ground-truth data is to generate a pseudo-ground-truth using an offline structure-from-motion technique. VSLAM-LAB integrates GLOMAP [24], enabling direct comparison with offline approaches. Furthermore, VSLAM-LAB includes multi-view stereo reconstruction methods DUSt3R [25] and Spann3R [41] (Table I, bottom).

Adding a New Method: To incorporate a new method, users must first specify the method's dependencies in the *pixi.toml* manifest. Users also need to define an *install* task, which typically runs *pip* for Python packages or builds a C++ package with *cmake*.

After the dependencies are defined and the method is installed, users then need to define a new class derived from *BaselineVSLAMLab* and in the simplest case specify the method's name, folder, and default parameter, as shown in the code listing below. The newly created class then needs to be added to the list of available methods in *baseline_utilities.py*. An example class implementing DSO is as follows:

```
class DSO_baseline(BaselineVSLAMLab):
    def __init__(self):
        baseline_name = "dso"
        baseline_folder = "dso"
        default_parameters = ["Preset: preset
            :0", "Mode: mode:1"]

        # Initialize the baseline
        super().__init__(baseline_name,
            baseline_folder,
            default_parameters)
```

VSLAM-LAB executes the script specified in the *execute* task within the *pixi.toml* manifest, which runs the method while processing the necessary inputs (see Section III-C); specifically:

```
--sequence_path  # Path Sequence
--calib_yaml     # Path calibration.yaml
--rgb_csv        # Path rgb.csv
--exp_id         # e.g. 00000
--settings_yaml  # Path method_set.yaml
--visualization  # True / False
```

The method must output a trajectory with a rigid transformation per frame in the required format (see Section III-C).

C. Visual SLAM Datasets

Table II lists all datasets available through VSLAM-LAB. Each dataset is automatically downloaded and converted into a standardized format to facilitate downstream processing. The directory structure is as follows, loosely following the TUM-RGBD Benchmark:

```
VSLAMLAB-BENCHMARK/DATASET/Sequence
|-- rgb/
|   |-- rgb_0000.jpg
|   |-- rgb_0001.jpg
|   |-- rgb_0002.jpg
|   |-- ...
|-- rgb.csv
|-- groundtruth.csv
|-- calibration.yaml
```

Trajectory Format: All trajectory files in VSLAM-LAB adhere to the format defined in [3]: `ts tx ty tz qx qy qz qw`, where `ts` represents the timestamp, `tx, ty, tz` are the translation components, and `qx, qy, qz, qw` define the orientation as a quaternion.

Image Undistortion: As part of the dataset preprocessing, all RGB images are undistorted to conform to a pinhole camera model. This ensures a consistent data format across all methods, eliminating the need for additional undistortion steps during evaluation.

Adding a New Dataset: Similar to adding a new method, to incorporate a new dataset, users must create a new class derived from *DatasetVSLAMLab* and add it to the *dataset_utilities*. The class functions should be overridden to automate data downloading and formatting according to the required structure. Specifically, the *download_sequence_data()* method should download and uncompress the dataset files, the *create_rgb_folder()* method should arrange the files according to the format specified above, the *create_calibration_yaml()* should create the *calibration.yaml* in OpenCV format, the *create_rgb_txt()* method should create the *rgb.csv* that lists all images in the correct order, and finally the *create_groundtruth_txt()* method should create the *groundtruth.csv* adhering to the trajectory format detailed above.

D. Experiment Customization

VSLAM-LAB allows for easy customization of experiments through a simple configuration process. Users can select specific methods, define the number of runs (considering non-deterministic outputs of most methods), and specify input parameters for each method to fine-tune its performance. The configuration file follows this structure:

```
# VSLAMLAB/configs/exp_config_easy.yaml

exp_config_easy_mast3rslam:
  Config: config_easy.yaml
  NumRuns: 10
  Parameters: {verbose: 0, max_rgb: 120}
  Method: mast3rslam

exp_config_easy_dpvo:
  Config: config_easy.yaml
  NumRuns: 10
  Parameters: {verbose: 0, max_rgb: 120}
  Method: dpvo
```

Additionally, users can customize the datasets used in an experiment by specifying sequences from different datasets within the configuration file:

```
# VSLAMLAB/configs/config_easy.yaml

rgbdtum:
- freiburg3_structure_texture_far
eth:
- table_3
7scenes:
- chess_seq-01
euroc:
- V1_01_easy
```

Adding a New Configuration: New configurations can be easily introduced by creating a new *config* file. This approach ensures continuous adaptability as methods evolve and datasets increase in complexity and diversity. Moreover, it

enhances reproducibility by enabling researchers to replicate specific experiments simply by sharing the corresponding configuration file.

E. Evaluation

Absolute Trajectory Error (ATE) [3]. Since monocular visual SLAM systems produce unscaled trajectories, ATE serves as the standard evaluation metric. It measures global consistency by aligning estimated and ground-truth trajectories [42], [43] and computing translational differences.

VSLAM-LAB automatically generates various plots and statistics to facilitate evaluation, with examples shown in Figures 1 and 2.

ATE Boxplots: Due to the non-deterministic nature of most VSLAM systems, multiple runs are necessary to account for random variations. To illustrate the variability in trajectory accuracy across all runs, ATE measurements are presented using boxplots. These plots provide a concise summary of absolute accuracy differences and system variability.

Cumulative ATE Plot: This plot represents the number of runs (y-axis) in which a system achieves an ATE below a given threshold (x-axis). The curve's position in the graph indicates the system's accuracy, while its width reflects variability. This visualization is particularly useful for comparing performance across different sequences [7], [8].

Radar ATE Plots: Given multiple ATE values from repeated runs of a method on a specific sequence, all ATE values are normalized using the median of the entire dataset. This normalization enables radar plots to represent the relative accuracy of different methods within a single visualization, even when applied to sequences with varying error scales. For instance, this approach allows meaningful comparisons between an indoor sequence with sub-centimeter accuracy and an outdoor sequence involving an autonomous vehicle. As shown in Figure 1, these plots effectively highlight trends in method performance.

Number of Estimated Frames: The VSLAM-LAB evaluation also includes reporting the number of frames estimated by each method, the number of frames used to compute ATE, and the total number of frames in the sequence. Since not every frame has an associated ground truth due to factors such as motion capture system limitations or GPS failures, this metric ensures a fair comparison between methods. Some systems take a conservative approach by estimating only partial trajectories, while others estimate camera positions for a subset of keyframes or provide predictions for every frame.

F. Other Tools

VSLAM-LAB provides evaluation tools to facilitate ablation studies, parameter search experiments, and analyses on specific or subsampled portions of sequences.

Parameter Ablation: The framework supports systematic experimentation with different parameter configurations, enabling seamless ablation studies:

```
# VSLAMLAB/configs/exp_ablation.yaml

exp_ablation_mast3rslam:
  Config: config_easy.yaml
  NumRuns: 100
  Parameters: {verbose: 0, max_rgb: 120}
  Ablation: configs/ablation.csv
  Method: mast3rslam
```

A sample ablation file follows:

```
# VSLAMLAB/configs/ablation.csv

exp_id  nFeatures  iniThFAST minThFAST
  0       500          20         7
  1       500          20         10
  ...     ...          ...        ...
  99      1000         40         10
```

Similar to parameter ablation, noise ablation can be conducted by specifying a file that defines the type and magnitude of noise applied to experimental runs.

RGB Sampling: The experiment configuration file allows flexible frame selection policies. Users can specify particular sequence segments to process or downsample frames to a given frequency. This functionality helps to reduce computational load, decrease execution time, evaluate specific sequence portions, and assess system robustness under low-frame-rate conditions.

	DPVO	DROID-SLAM	MASt3R-SLAM	ORB-SLAM2
V1 01 easy	51.52	75.56	59.07	54.95
chess 01	50.01	70.01	80.01	50.01
fr1 xyz	55.15	137.86	87.73	50.14
office0	60.01	55.01	55.01	45.01
table 3	59.33	59.33	67.81	50.86
Overall	55.20 ± 4.97	79.55 ± 30.68	69.93 ± 12.63	50.19 ± 3.23

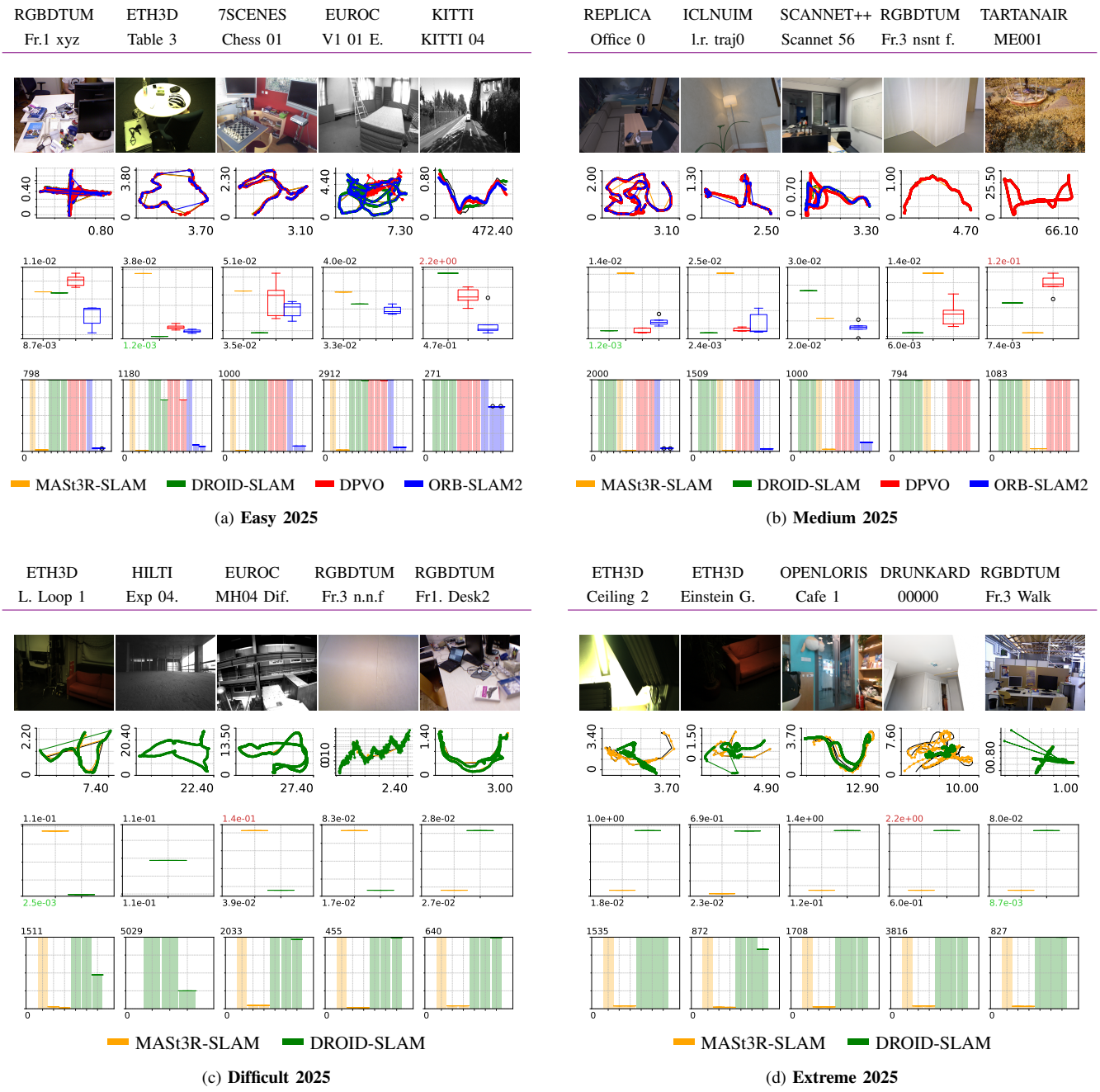
TABLE III: **Average processing time per frame (in milliseconds) for each VSLAM system.**

IV. EXPERIMENTS

Different VSLAM methods exhibit varying performance depending on a wide range of factors, including scene characteristics (*e.g.*, texture and depth distribution) and camera motion (*e.g.*, translational and rotational velocities). Additionally, each method presents distinct trade-offs between accuracy, robustness, computational efficiency, and scalability.

Here, we introduce a set of four benchmarking configurations that cluster 20 sequences from 12 diverse datasets based on the *degree of difficulty* associated with the *challenges* present in each scene. These new benchmarks enable straightforward comparison of existing techniques, and will help SLAM researchers to compare their new methods on established, standardized sequences. As methods become more robust and accurate, we can easily introduce new configurations with even more challenging and diverse datasets as described in Section III-C. Future work might also introduce domain-specific configurations, such as methods that focus on underwater SLAM or interplanetary SLAM.

Easy 2025: The top row of Figure 2a displays frames from the five sequences included in the Easy 2025 configuration. These sequences feature well-structured environments with rich image textures and gentle motion. The main exception is sequence 04 from the KITTI dataset, which introduces large



(a) **Easy 2025** (b) **Medium 2025**

(c) **Difficult 2025** (d) **Extreme 2025**

Fig. 2: **Benchmarking Configurations for VSLAM-LAB.** The figure presents four evaluation categories—Easy 2025, Medium 2025, Difficult 2025, and Extreme 2025—grouping sequences based on increasing levels of environmental complexity and motion challenges. Each configuration includes representative sequences (top row), camera trajectories (middle row), and ATE boxplots (bottom row) to facilitate performance comparisons across VSLAM methods. We note that these categories are examples only, and new categories can be easily added as described in Section III-D.

disparities due to the low frame rate (10 Hz) and vehicle movement. Aside from a distant moving car in the KITTI sequence, the scenes are predominantly static.

The ATE boxplot (third row) in Figure 2a highlights how, given the relatively low complexity of these sequences, traditional methods such as ORB-SLAM2 achieve competitive performance with state-of-the-art approaches like DROID-SLAM. Notably, ORB-SLAM2 outperforms both DROID-SLAM and MASt3R-SLAM in the KITTI dataset. This observation aligns with prior findings that these methods tend to degrade in performance in outdoor environments [21].

Medium 2025: Figure 2b includes sequences characterized by low-texture environments, such as the column in Fr3 Structure No Texture from the RGB-D TUM Benchmark and the Living Room Traj0 from the ICL-NUIM dataset. Other sequences, such as ME001 from the TartanAir dataset and a sequence from ScanNet++ feature stronger camera motion. DROID-SLAM successfully reconstructs all trajectories in

these sequences, achieving the best performance.

Difficult 2025: This configuration includes sequences characterized by strong camera motion, often leading to motion blur, such as the drone-mounted camera in sequence MH 04 Difficult from the EuRoC dataset and the handheld camera in sequence fr1 desk2 from the RGB-D TUM Benchmark. Additionally, it contains sequences with low structural complexity and poor texture, such as the construction site walls in sequence exp04 from the HILTI 2022 Challenge. It also includes sequences captured under minimal lighting conditions, such as large loop 1 from the ETH3D Benchmark.

Figure 2c presents only the performance of DROID-SLAM and MASt3R-SLAM, as DPVO and ORB-SLAM2 exhibited catastrophic accuracy degradation in these sequences. Notably, in challenging scenarios without extreme lighting variations, dynamic objects, or complete texture absence, DROID-SLAM consistently outperforms MASt3R-SLAM. It is important to note that MASt3R-SLAM is a relatively recent VSLAM system and may have undergone less fine-tuning for these datasets.

Extreme 2025: Figure 2d presents sequences that exhibit extreme conditions, including dynamic objects occupying a significant portion of the image, as seen in fr3 walking from the TUM-RGBD Benchmark. It also includes sequences with strong lighting variations, such as ceiling2 and einstein global light changes 1 from the ETH3D Benchmark, and scenes with an almost complete lack of texture, as observed in the walls of cafe1-1 from the OPENLORIS dataset or 00000 1014 level0 from the Drunkard's dataset.

In these very challenging sequences, MASt3R-SLAM demonstrates promising results, consistently achieving higher accuracy than DROID-SLAM.

The average tracking time per frame for each SLAM system is reported in Table III, highlighting the computational efficiency differences across methods.

V. CONCLUSIONS

As SLAM researchers, our focus should be on developing new methods that are robust to a wide range of scenarios and accurately track the camera's pose while building a map of the environment. While open-sourcing of SLAM methods (and computer vision / robotics papers in general) has become more common, running these methods for benchmarking purposes and evaluating them on new datasets takes considerable time and effort.

In this paper, we introduce VSLAM-LAB, the first comprehensive framework for benchmarking VSLAM systems. VSLAM-LAB addresses the lack of standardization in VSLAM implementations, datasets, and evaluation procedures, which leads to fragmented comparisons and significant programming overhead. Our framework provides tools to unify VSLAM methods, data, and benchmarks, greatly simplifying development, reproducibility, and evaluation tasks. Installing and using VSLAM-LAB can be as easy as cloning our GitHub repository and running a single command: *pixi run demo*.

Our experimental results compare 4 state-of-the-art VSLAM implementations across 20 sequences from 12 different datasets. For the first time, these comparisons highlight that "old" approaches like ORB-SLAM2 still outperform new deep-learning techniques like DROID-SLAM and MASt3R-SLAM on some outdoor sequences, while the very recent MASt3R-SLAM performs best on extremely difficult sequences where both ORB-SLAM2 and DPVO catastrophically fail. The inclusion of the recently proposed MASt3R-SLAM demonstrates VSLAM-LAB's ability to seamlessly integrate novel implementations—an effort we will continue in the future, hopefully with the help of the research community.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] A. Fontan, J. Civera, and M. Milford, "AnyFeature-VSLAM: Automating the usage of any chosen feature into visual slam," in *Robotics: Science and Systems*, 2024.

[3] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[4] T. Schops, J. L. Schonberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3260–3269.

[5] T. Schops, T. Sattler, and M. Pollefeys, "BAD SLAM: Bundle Adjusted Direct RGB-D SLAM," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 134–144.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[7] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," *arXiv preprint arXiv:1607.02555*, 2016.

[8] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[9] L. Freda, "pySLAM: An open-source, modular, and extensible framework for SLAM," *arXiv preprint arXiv:2502.11955*, 2025.

[10] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.

[11] L. Nardi, B. Bodin, M. Z. Zia, J. Mawer, A. Nisbet, P. H. Kelly, A. J. Davison, M. Luján, M. F. O'Boyle, G. Riley, *et al.*, "Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 5783–5790.

[12] B. Bodin, H. Wagstaff, S. Saeedi, L. Nardi, E. Vespa, J. Mayer, A. Nisbet, M. Luján, S. Furber, A. Davison, P. Kelly, and M. O'Boyle, "SLAMBench2: multi-objective head-to-head benchmarking for Visual SLAM," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 3637–3644.

[13] M. Bujanca, P. Gafton, S. Saeedi, A. Nisbet, B. Bodin, M. F. O'Boyle, A. Davison, P. Kelly, G. Riley, B. Lennox, M. Luján, and S. Furber, "SLAMBench 3.0: systematic automated reproducible evaluation of SLAM systems for robot vision challenges and scene understanding," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 6351–6358.

[14] M. Bujanca, X. Shi, M. Spear, P. Zhao, B. Lennox, and M. Luján, "Robust SLAM systems: Are we there yet?" in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 5320–5327.

[15] Y. Yang, B. Xu, Y. Li, and S. Schwertfeger, "The SLAM Hive benchmarking suite," in *IEEE International Conference on Robotics and Automation*, 2023, pp. 11 257–11 263.

[16] G. Berton, R. Mereu, G. Trivigno, C. Masone, G. Csurka, T. Sattler, and B. Caputo, "Deep visual geo-localization benchmark," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5396–5407.

[17] C. Toft, W. Maddern, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, T. Pajdla, *et al.*, "Long-term visual localization revisited," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 4, pp. 2074–2088, 2020.

[18] M. Zaffar, S. Garg, M. Milford, J. Kooij, D. Flynn, K. McDonald-Maier, and S. Ehsan, "VPR-bench: an open-source visual place recognition evaluation framework with quantifiable viewpoint and appearance change," *International Journal of Computer Vision*, vol. 129, no. 7, pp. 2136–2174, 2021.

[19] G. Berton, G. Trivigno, B. Caputo, and C. Masone, "EigenPlaces: Training viewpoint robust models for visual place recognition," in *IEEE/CVF International Conference on Computer Vision*, 2023, pp. 11 080–11 090.

[20] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[21] Z. Teed and J. Deng, "Droid-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 558–16 569, 2021.

[22] H. Matsuki, R. Murai, P. H. Kelly, and A. J. Davison, "Gaussian splatting SLAM," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 039–18 048.

[23] L. Lipson, Z. Teed, and J. Deng, "Deep patch visual SLAM," in *European Conference on Computer Vision*, 2025, pp. 424–440.

[24] L. Pan, D. Baráth, M. Pollefeys, and J. L. Schönberger, "Global structure-from-motion revisited," in *European Conference on Computer Vision*, 2024, pp. 58–77.

[25] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 697–20 709.

[26] R. Murai, E. Dexheimer, and A. J. Davison, "Mast3r-slam: Real-time dense slam with 3d reconstruction priors," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 16 695–16 705.

[27] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[28] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi, "Real-time RGB-D camera relocalization," in *IEEE International Symposium on Mixed and Augmented Reality*, 2013, pp. 173–179.

[29] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 1524–1531.

[30] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai, "Scannet++: A high-fidelity dataset of 3d indoor scenes," in *IEEE/CVF International Conference on Computer Vision*, 2023, pp. 12–22.

[31] A. Mallios, E. Vidal, R. Campos, and M. Carreras, "Underwater caves sonar data set," *The International Journal of Robotics Research*, vol. 36, no. 12, pp. 1247–1251, 2017.

[32] L. Meyer, M. Smíšek, A. Fontan Villacampa, L. Oliva Maza, D. Medina, M. J. Schuster, F. Steidle, M. Vayugundla, M. G. Müller, B. Rebele, *et al.*, "The MADMAX data set for visual-inertial rover navigation on mars," *Journal of Field Robotics*, vol. 38, no. 6, pp. 833–853, 2021.

[33] J. Straub *et al.*, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[34] X. Shi, D. Li, P. Zhao, Q. Tian, Y. Tian, Q. Long, C. Zhu, J. Song, F. Qiao, L. Song, *et al.*, "Are we ready for service robots? the openloris-scene datasets for lifelong slam," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 3139–3145.

[35] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor, and S. Scherer, "Tartanair: A dataset to push the limits of visual slam," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 4909–4916.

[36] D. Recasens, J. Lamarca, J. M. Fácil, J. Montiel, and J. Civera, "Endo-depth-and-motion: Reconstruction and tracking in endoscopic videos using depth networks and photometric constraints," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7225–7232, 2021.

[37] P. Mountney, D. Stoyanov, and G.-Z. Yang, "Three-dimensional tissue deformation recovery and tracking," *IEEE Signal Processing Magazine*, vol. 27, no. 4, pp. 14–24, 2010.

[38] P.-E. Sarlin, M. Dusmanu, J. L. Schönberger, P. Speciale, L. Gruber, V. Larsson, O. Miksik, and M. Pollefeys, "LaMAR: Benchmarking localization and mapping for augmented reality," in *European Conference on Computer Vision*. Springer, 2022, pp. 686–704.

[39] L. Zhang, M. Helmberger, L. F. T. Fu, D. Wisth, M. Camurri, D. Scaramuzza, and M. Fallon, "Hilti-oxford dataset: A millimeter-accurate benchmark for simultaneous localization and mapping," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 408–415, 2022.

[40] D. Recasens, M. R. Oswald, M. Pollefeys, and J. Civera, "The drunkard's odometry: Estimating camera motion in deforming scenes," in *International Conference on Neural Information Processing Systems*, 2023, pp. 48 877–48 889.

[41] H. Wang and L. Agapito, "3d reconstruction with spatial memory," *arXiv preprint arXiv:2408.16061*, 2024.

[42] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[43] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991.