VSLAM-LAB

A Comprehensive Framework for Visual SLAM Methods and Datasets

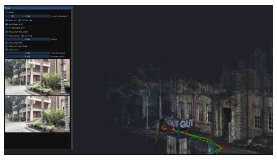
Alejandro Fontan, Tobias Fischer, Javier Civera and Michael Milford

https://github.com/VSLAM-LAB/VSLAM-LAB https://www.youtube.com/@VSLAM-LAB

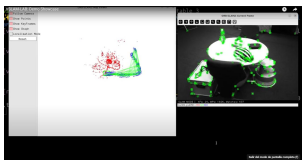
Compile and configure VSLAM systems, download and process datasets, and design, run, and evaluate experiments — **all from a single command line!** 📦 Sensor setups: Mono, RGBD, Stereo & mono-VI

```
# pixi run demo <baseline> <dataset> <seq> <sensor>
$ pixi run demo orbslam3 euroc MH_easy_01 mono-vi
```

MASt3R-SLAM in casual video

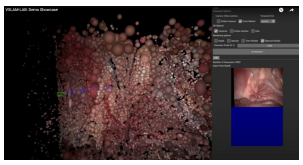


ORB-SLAM2 in ETH3D Dataset

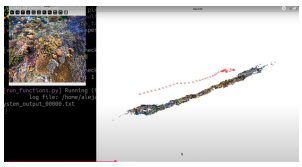


DPVO in 7 Scenes Dataset

MonoGS in Hamlyn Dataset



DROID-SLAM in Sweet Corals



Seamlessly execute complex experiments with minimal setup.

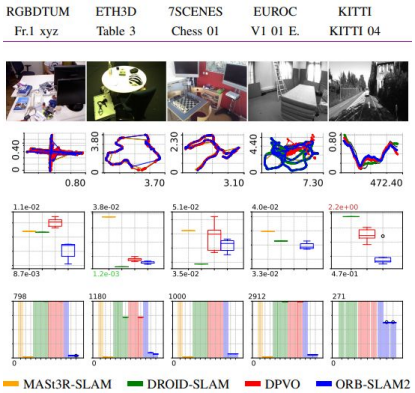
```
# VSLAMLAB/configs/exp_config_easy.yaml

exp_mast3rslam:
  Config: config_easy.yaml
  NumRuns: 5
  Parameters: {verbose: 0, max_rgb: 1000}
  Module: mast3rslam

exp_droidslam:
  Config: config_easy.yaml
  NumRuns: 5
  Parameters: {verbose: 0, max_rgb: 1000}
  Module: droidslam
```

```
# VSLAMLAB/configs/config_easy.yaml

rgbdtum:
  - rgbd_dataset_freiburg1_xyz
eth:
  - table_3
7scenes:
  - chess_seq-01
euroc:
  - V1_01_easy
kitti:
  - 04
```



… and easily include your own methods and/or datasets.

Define a new class derived from BaselineVSLAMLab

```
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)
```

Method necessary inputs

```
--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 dataset structure is as follows

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

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.

To incorporate a new method, users must specify the method's dependencies in the pixi.toml manifest.

```
# orbslam2 begin
[feature.orbslam2]
channels = ["https://fast.prefix.dev/conda-forge", "fontan"]
platforms = ["linux-64"]

[feature.orbslam2.tasks]
git-clone = "git-clone.sh VSLAM-LAB ORB_SLAM2 Baselines"
install = {cmd = "./build.sh", cwd = "Baselines/ORB_SLAM2"}
execute_mono = {cmd = "./bin/mono",  cwd = "Baselines/ORB_SLAM2"}
execute_rgbd = {cmd = "./bin/rgbd",  cwd = "Baselines/ORB_SLAM2"}

[feature.orbslam2.dependencies]
libgl-devel = "*"
libegl-devel = "*"
libglvnd-devel = "*"
libopengl-devel = "*"
xorg-libxext ="*"
xorg-libx11 ="*"
xorg-libxfixes="*"
xorg-libxrender="*"
xorg-libice="*"
xorg-libsm="*"
xorg-libxdmcp="*"
xorg-libxau="*"

compilers = "*"
ninja = ">=1.12.1 ,<1.13"
cmake = ">=3.31.1 ,<3.32"
pkg-config = ">=0.29.2 ,<0.30"
opencv = ">=4.10.0 ,<5"
eigen = ">=3.4.0 ,<3.5.0"
yaml-cpp = ">=0.8.0,<0.9"
pangolin = {version=">=2024.07.03", build="h3c9d7id_0", channel="fontan"}
# orbslam2 end
```