
stella_vslam

Shinya Sumikura. For the changes after forking, stella-cv.

May 04, 2023

GETTING STARTED

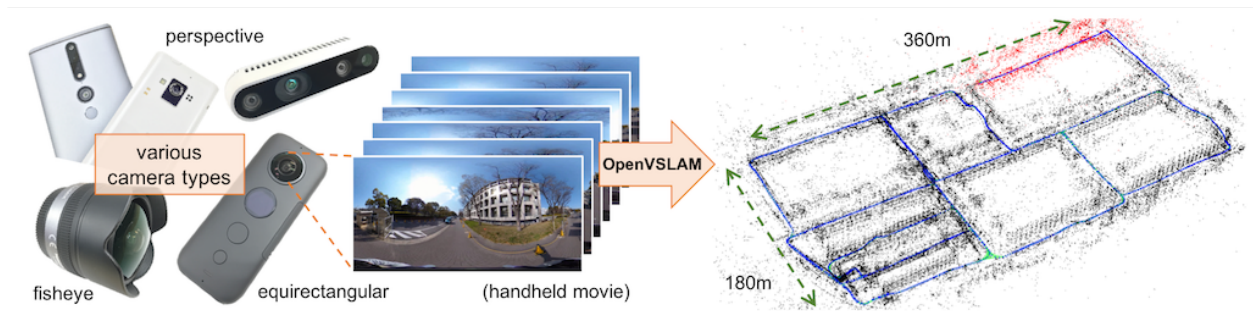
| | | |
|----------|---|-----------|
| 1 | Overview | 3 |
| 1.1 | Installation | 4 |
| 1.2 | Tutorial | 4 |
| 1.3 | Reference | 4 |
| 2 | Installation | 5 |
| 2.1 | Source code | 5 |
| 2.2 | Dependencies | 5 |
| 2.3 | Prerequisites for Unix | 6 |
| 2.4 | Build Instructions | 11 |
| 2.5 | Server Setup for SocketViewer | 12 |
| 3 | Simple Tutorial | 15 |
| 3.1 | TL; DR | 15 |
| 3.2 | Sample Datasets | 16 |
| 3.3 | Tracking and Mapping | 17 |
| 3.4 | Localization | 21 |
| 4 | Running on Docker | 25 |
| 4.1 | Instructions for PangolinViewer | 25 |
| 4.2 | Instructions for SocketViewer | 26 |
| 4.3 | Bind of Directories | 28 |
| 5 | Running on ROS | 29 |
| 5.1 | ROS Package | 29 |
| 5.2 | ROS2 Package | 31 |
| 6 | Example | 35 |
| 6.1 | SLAM with Video Files | 35 |
| 6.2 | SLAM with Image Sequences | 35 |
| 6.3 | SLAM with Standard Datasets | 36 |
| 6.4 | SLAM with UVC camera | 38 |
| 7 | Parameters | 39 |
| 7.1 | System | 39 |
| 7.2 | Camera | 39 |
| 7.3 | Feature | 40 |
| 7.4 | Preprocessing | 40 |
| 7.5 | Tracking | 40 |
| 7.6 | Mapping | 41 |
| 7.7 | StereoRectifier | 41 |

| | | |
|-----------|--|-----------|
| 7.8 | Initializer | 41 |
| 7.9 | Relocalizer | 42 |
| 7.10 | KeyframeInserter | 42 |
| 7.11 | PangolinViewer | 42 |
| 7.12 | SocketPublisher | 43 |
| 7.13 | LoopDetector | 43 |
| 7.14 | MarkerModel | 43 |
| 8 | Relocalization | 45 |
| 8.1 | What is Relocalization? Why it is needed? | 45 |
| 8.2 | Steps in Relocalization | 45 |
| 9 | Trouble Shooting | 47 |
| 9.1 | For building | 47 |
| 9.2 | For Viewer | 47 |
| 9.3 | For SLAM | 47 |
| 10 | Citation of original version of OpenVSLAM (xdspacelab/openslam) | 49 |

This is the [stella_vslam](#) documentation.

NOTE: This is a community fork of [xdspacelab/openslam](#). It was created to continue active development of Open-VSLAM on Jan 31, 2021. The original repository is no longer available. Please read README.md in [stella_vslam](#).

OVERVIEW



stella_vslam is a monocular, stereo, and RGBD visual SLAM system. The notable features are:

- It is compatible with **various type of camera models** and can be easily customized for other camera models.
- Created maps can be **stored and loaded**, then stella_vslam can **localize new images** based on the prebuilt maps.
- The system is fully modular. It is designed by encapsulating several functions in separated components with easy-to-understand APIs.
- We provided **some code snippets** to understand the core functionalities of this system.

stella_vslam is based on an indirect SLAM algorithm with sparse features, such as ORB-SLAM, ProSLAM, and UcoSLAM. One of the noteworthy features of stella_vslam is that the system can deal with various type of camera models, such as perspective, fisheye, and equirectangular. If needed, users can implement extra camera models (e.g. dual fisheye, catadioptric) with ease. For example, visual SLAM algorithm using **equirectangular camera models** (e.g. RICOH THETA series, insta360 series, etc) is shown above.

Some code snippets to understand the core functionalities of the system are provided. You can employ these snippets for in your own programs. Please see the *.cc files in ./example directory or check [Simple Tutorial](#) and [Example](#).

Also, some examples to run stella_vslam on ROS framework are provided. Please check [ROS Package](#).

If you notice any bugs, report them to [GitHub issues](#). If you have any questions, ask them at [GitHub Discussions](#).

1.1 Installation

Please see *Installation* chapter.

The instructions for Docker users are also provided.

1.2 Tutorial

Please see *Simple Tutorial*.

A sample ORB vocabulary file can be downloaded from [here](#).

Sample datasets are also provided at [here](#).

If you would like to run visual SLAM with standard benchmarking datasets (e.g. KITTI Odometry dataset), please see *SLAM with standard datasets*.

1.3 Reference

- Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. 2015. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics* 31, 5 (2015), 1147–1163.
- Raul Mur-Artal and Juan D. Tardos. 2017. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics* 33, 5 (2017), 1255–1262.
- Dominik Schlegel, Mirco Colosi, and Giorgio Grisetti. 2018. ProSLAM: Graph SLAM from a Programmer’s Perspective. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. 1–9.
- Rafael Munoz-Salinas and Rafael Medina Carnicer. 2019. UcoSLAM: Simultaneous Localization and Mapping by Fusion of KeyPoints and Squared Planar Markers. *arXiv:1902.03729*.
- Mapillary AB. 2019. OpenSfM. <https://github.com/mapillary/OpenSfM>.
- Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. 2010. A Tutorial on Graph-Based SLAM. *IEEE Transactions on Intelligent Transportation Systems Magazine* 2, 4 (2010), 31–43.
- Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. 2011. g2o: A general framework for graph optimization. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. 3607–3613.

INSTALLATION

2.1 Source code

The source code can be viewed from this [GitHub repository](#).

Cloning the repository:

```
git clone --recursive https://github.com/stella-cv/stella_vslam.git
```

If you are Windows 10 user, please install the dependencies and stella_vslam with *SocketViewer support* on [Windows Subsystem for Linux \(WSL\)](#).

Docker systems can be used instead of preparing the dependencies manually.

2.2 Dependencies

stella_vslam requires a **C++11-compliant** compiler. It relies on several open-source libraries as shown below.

2.2.1 Requirements for stella_vslam

- [Eigen](#) : version 3.3.0 or later.
- [g2o](#) : 20230223_git or later.
- [SuiteSparse](#) : Required by g2o.
- [FBoW](#) : **Please use the custom version of FBoW** released in <https://github.com/stella-cv/FBoW>.
- [yaml-cpp](#) : version 0.6.0 or later.
- [OpenCV](#) : version 3.3.1 or later. (NOTE: If ROS is installed, use the one installed with ROS.)

Note: OpenCV with GUI support is necessary for using the built-in viewer (Pangolin Viewer). OpenCV with video support is necessary if you plan on using video files (e.g. .mp4) as inputs. If your CPU has many cores, it is recommended to enable TBB.

2.2.2 Requirements for PangolinViewer

We provided an OpenGL-based simple viewer.

This viewer is implemented with [Pangolin](#). Thus, we call it **PangolinViewer**.

Please install the following dependencies if you plan on using PangolinViewer.

- [Pangolin](#) : Please use the latest release. Tested on commit ID [eab3d34](#).
- [GLEW](#) : Required by Pangolin.

Note: If Pangolin version 0.7 or higher, C++17 is required.

2.2.3 Requirements for SocketViewer

We provided an WebGL-based simple viewer running on web browsers.

The SLAM systems publish the map and the frame to the server implemented with [Node.js](#) via WebSocket. Thus, we call it **SocketViewer**.

Please install the following dependencies if you plan on using SocketViewer.

- [socket.io-client-cpp](#) : **Please use the custom version of socket.io-client-cpp** released in <https://github.com/shinsumicco/socket.io-client-cpp>.
- [Protobuf](#) : version 3 or later.

The following libraries are the dependencies for the server.

- [Node.js](#) : version 6 or later.
- [npm](#) : Tested on version 3.5.2.

2.2.4 Recommended

- [backward-cpp](#) : Used for stack-trace logger.

2.3 Prerequisites for Unix

Note: If your PC is frozen during the build, please reduce the number of parallel compile jobs when executing `make` (e.g. `make -j2`).

2.3.1 Installing for Linux

Tested for **Ubuntu 20.04**.

Install the dependencies via apt.

```
apt update -y
apt upgrade -y --no-install-recommends
# basic dependencies
apt install -y build-essential pkg-config cmake git wget curl unzip
# g2o dependencies
apt install -y libatlas-base-dev libsuitesparse-dev
# OpenCV dependencies
apt install -y libgtk-3-dev ffmpeg libavcodec-dev libavformat-dev libavutil-dev
↪ libswscale-dev libavresample-dev libtbb-dev
# eigen dependencies
apt install -y gfortran
# backward-cpp dependencies (optional)
apt install -y binutils-dev
# other dependencies
apt install -y libyaml-cpp-dev libgflags-dev sqlite3 libsqlite3-dev

# (if you plan on using PangolinViewer)
# Pangolin dependencies
apt install -y libglew-dev

# (if you plan on using SocketViewer)
# Protobuf dependencies
apt install -y autogen autoconf libtool
# Node.js
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
apt install -y nodejs
```

Download and install Eigen from source.

```
cd /tmp
wget -q https://gitlab.com/libeigen/eigen/-/archive/3.3.7/eigen-3.3.7.tar.bz2
tar xf eigen-3.3.7.tar.bz2 && rm -rf eigen-3.3.7.tar.bz2
cd eigen-3.3.7
mkdir -p build && cd build
cmake \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=/usr/local \
  ..
make -j4 && sudo make install
```

Download, build and install OpenCV from source. (NOTE: If ROS is installed, never do the following steps.)

```
cd /tmp
# Download OpenCV
wget -q https://github.com/opencv/opencv/archive/4.5.5.zip
unzip -q 4.5.5.zip && rm -rf 4.5.5.zip
# Download aruco module (optional)
wget -q https://github.com/opencv/opencv_contrib/archive/refs/tags/4.5.5.zip -O opencv_
```

(continues on next page)

(continued from previous page)

```

↪ contrib-4.5.5.zip
unzip -q opencv_contrib-4.5.5.zip && rm -rf opencv_contrib-4.5.5.zip
mkdir -p extra && mv opencv_contrib-4.5.5/modules/aruco extra
rm -rf opencv_contrib-4.5.5
# Build and install OpenCV
cd opencv-4.5.5
mkdir -p build && cd build
cmake \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=/usr/local \
  -DBUILD_DOCS=OFF \
  -DBUILD_EXAMPLES=OFF \
  -DBUILD_JASPER=OFF \
  -DBUILD_OPENEXR=OFF \
  -DBUILD_PERF_TESTS=OFF \
  -DBUILD_TESTS=OFF \
  -DBUILD_PROTOBUF=OFF \
  -DBUILD_opencv_apps=OFF \
  -DBUILD_opencv_dnn=OFF \
  -DBUILD_opencv_ml=OFF \
  -DBUILD_opencv_python_bindings_generator=OFF \
  -DENABLE_CXX11=ON \
  -DENABLE_FAST_MATH=ON \
  -DWITH_EIGEN=ON \
  -DWITH_FFMPEG=ON \
  -DWITH_TBB=ON \
  -DWITH_OPENMP=ON \
  -DOPENCV_EXTRA_MODULES_PATH=/tmp/extra \
  ..
make -j4 && sudo make install

```

Jump to *Common Installation Instructions* for the next step.

2.3.2 Installing for macOS

Tested for **macOS High Sierra**.

Install the dependencies via brew.

```

brew update
# basic dependencies
brew install pkg-config cmake git
# g2o dependencies
brew install suite-sparse
# OpenCV dependencies and OpenCV
brew install eigen ffmpeg opencv
# other dependencies
brew install yaml-cpp glog gflags sqlite3

# (if you plan on using PangolinViewer)
# Pangolin dependencies
brew install glew

```

(continues on next page)

(continued from previous page)

```
# (if you plan on using SocketViewer)
# Protobuf dependencies
brew install automake autoconf libtool
# Node.js
brew install node
```

Jump to *Common Installation Instructions* for the next step.

2.3.3 Common Installation Instructions

Download, build and install **the custom FBoW** from source.

```
cd /tmp
git clone https://github.com/stella-cv/FBoW.git
cd FBoW
mkdir build && cd build
cmake \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=/usr/local \
  ..
make -j4 && sudo make install
```

Download, build and install g2o.

```
cd /tmp
git clone https://github.com/RainerKuemmerle/g2o.git
cd g2o
git checkout 20230223_git
mkdir build && cd build
cmake \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=/usr/local \
  -DBUILD_SHARED_LIBS=ON \
  -DBUILD_UNITTESTS=OFF \
  -DG2O_USE_CHOLMOD=OFF \
  -DG2O_USE_CSPARSE=ON \
  -DG2O_USE_OPENGL=OFF \
  -DG2O_USE_OPENMP=OFF \
  -DG2O_BUILD_APPS=OFF \
  -DG2O_BUILD_EXAMPLES=OFF \
  -DG2O_BUILD_LINKED_APPS=OFF \
  ..
make -j4 && sudo make install
```

Download, build and install backward-cpp.

```
cd /tmp
git clone https://github.com/bombela/backward-cpp.git
cd backward-cpp
git checkout 5ffb2c879ebd3b3bdb8477c671e32b1c984beaa
mkdir build && cd build
```

(continues on next page)

(continued from previous page)

```
cmake \  
-DCMAKE_BUILD_TYPE=Release \  
-DCMAKE_INSTALL_PREFIX=/usr/local \  
..  
make -j4 && sudo make install
```

(if you plan on using PangolinViewer)

Download, build and install Pangolin from source.

```
cd /tmp  
git clone https://github.com/stevenlovegrove/Pangolin.git  
cd Pangolin  
git checkout eab3d3449a33a042b1ee7225e1b8b593b1b21e3e  
mkdir build && cd build  
cmake \  
-DCMAKE_BUILD_TYPE=Release \  
-DCMAKE_INSTALL_PREFIX=/usr/local \  
-DBUILD_EXAMPLES=OFF \  
-DBUILD_PANGOLIN_DEPTHSENSE=OFF \  
-DBUILD_PANGOLIN_FFMPEG=OFF \  
-DBUILD_PANGOLIN_LIBDC1394=OFF \  
-DBUILD_PANGOLIN_LIBJPEG=OFF \  
-DBUILD_PANGOLIN_LIBOPENEXR=OFF \  
-DBUILD_PANGOLIN_LIBPNG=OFF \  
-DBUILD_PANGOLIN_LIBTIFF=OFF \  
-DBUILD_PANGOLIN_LIBUVC=OFF \  
-DBUILD_PANGOLIN_LZ4=OFF \  
-DBUILD_PANGOLIN_OPENNI=OFF \  
-DBUILD_PANGOLIN_OPENNI2=OFF \  
-DBUILD_PANGOLIN_PLEORA=OFF \  
-DBUILD_PANGOLIN_PYTHON=OFF \  
-DBUILD_PANGOLIN_TELICAM=OFF \  
-DBUILD_PANGOLIN_UVC_MEDIAFOUNDATION=OFF \  
-DBUILD_PANGOLIN_V4L=OFF \  
-DBUILD_PANGOLIN_ZSTD=OFF \  
..  
make -j4 && sudo make install
```

(if you plan on using SocketViewer)

Download, build and install socket.io-client-cpp from source.

```
cd /tmp  
git clone https://github.com/shinsumicco/socket.io-client-cpp.git  
cd socket.io-client-cpp  
git submodule init  
git submodule update
```

(continues on next page)

(continued from previous page)

```
mkdir build && cd build
cmake \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=/usr/local \
  -DBUILD_UNIT_TESTS=OFF \
  ..
make -j4
sudo make install
```

(if you plan on using SocketViewer)

Install Protobuf.

If you use Ubuntu 20.04 (or later) or macOS, Protobuf 3.x can be installed via apt or brew.

```
# for Ubuntu
apt install -y libprotobuf-dev protobuf-compiler
# for macOS
brew install protobuf
```

Otherwise, please download, build and install Protobuf from source.

```
wget -q https://github.com/google/protobuf/archive/v3.6.1.tar.gz
tar xf v3.6.1.tar.gz
cd protobuf-3.6.1
./autogen.sh
./configure \
  --prefix=/usr/local \
  --enable-static=no
make -j4
sudo make install
```

2.4 Build Instructions

When building with support for PangolinViewer, please specify the following cmake options: `-DUSE_PANGOLIN_VIEWER=ON` and `-DUSE_SOCKET_PUBLISHER=OFF`.

```
cd /path/to/stella_vslam
mkdir build && cd build
cmake \
  -DUSE_STACK_TRACE_LOGGER=ON \
  -DCMAKE_BUILD_TYPE=RelWithDebInfo \
  -DUSE_PANGOLIN_VIEWER=ON \
  -DINSTALL_PANGOLIN_VIEWER=ON \
  -DUSE_SOCKET_PUBLISHER=OFF \
  -DBUILD_TESTS=OFF \
  -DBUILD_EXAMPLES=ON \
  ..
make -j4 && sudo make install
```

When building with support for SocketViewer, please specify the following cmake options:
-DUSE_PANGOLIN_VIEWER=OFF and -DUSE_SOCKET_PUBLISHER=ON.

```
cd /path/to/stella_vslam
mkdir build && cd build
cmake \
  -DUSE_STACK_TRACE_LOGGER=ON \
  -DCMAKE_BUILD_TYPE=RelWithDebInfo \
  -DUSE_PANGOLIN_VIEWER=OFF \
  -DUSE_SOCKET_PUBLISHER=ON \
  -DINSTALL_SOCKET_PUBLISHER=ON \
  -DBUILD_TESTS=OFF \
  -DBUILD_EXAMPLES=ON \
  ..
make -j4 && sudo make install
```

After building, check to see if it was successfully built by executing `./run_kitti_slam -h`.

```
$ ./run_kitti_slam -h
Allowed options:
-h, --help                produce help message
-v, --vocab arg            vocabulary file path
-d, --data-dir arg        directory path which contains dataset
-c, --config arg          config file path
--frame-skip arg (=1)     interval of frame skip
--no-sleep                not wait for next frame in real time
--auto-term               automatically terminate the viewer
--log-level arg (=info)   log level
```

2.5 Server Setup for SocketViewer

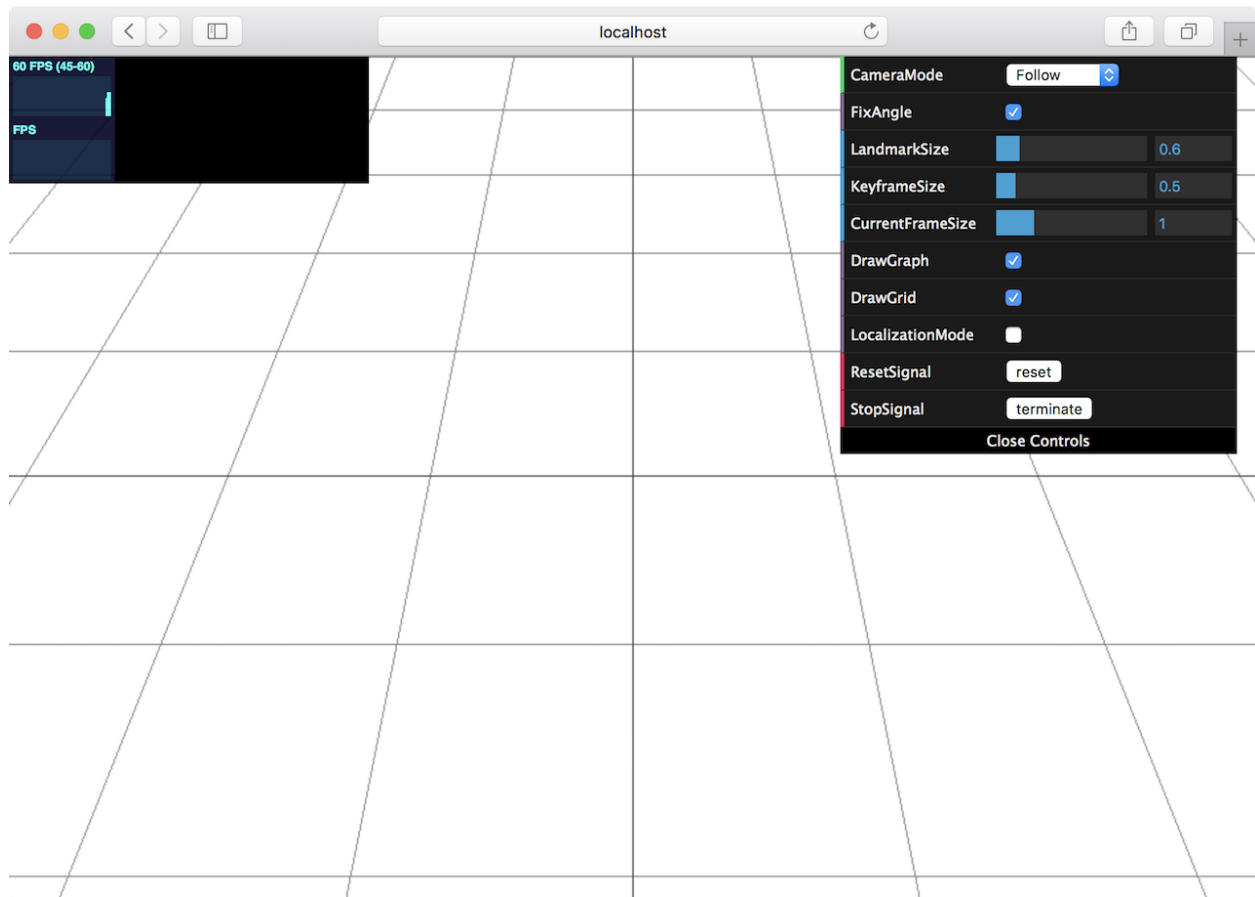
If you plan on using SocketViewer, please setup the environment for the server with npm.

```
$ cd /path/to/stella_vslam/viewer
$ ls
Dockerfile app.js package.json public views
$ npm install
added 88 packages from 60 contributors and audited 204 packages in 2.105s
found 0 vulnerabilities
$ ls
Dockerfile app.js node_modules package-lock.json package.json public views
```

Then, launch the server with `node app.js`.

```
$ cd /path/to/stella_vslam/viewer
$ ls
Dockerfile app.js node_modules package-lock.json package.json public views
$ node app.js
WebSocket: listening on *:3000
HTTP server: listening on *:3001
```

After launching, please access to `http://localhost:3001/` to check whether the server is correctly launched.



Note: When you try [the tutorial](#) and [the examples](#) with SocketViewer, please launch the server in the other terminal and access to it with the web browser **in advance**.

SIMPLE TUTORIAL

3.1 TL; DR

Note: If you use *SocketViewer*, please launch the server in the other terminal and access to it with the web browser in advance.

Running the following commands will give a feel for what *stella_vslam* can do. The later parts of this chapter explains what each of the commands do in more detail.

```
# at the build directory of stella_vslam ...
$ pwd
/path/to/stella_vslam/build/
$ ls
run_video_slam  lib/  ...

# download an ORB vocabulary from GitHub
curl -sL "https://github.com/stella-cv/FBoW_orb_vocab/raw/main/orb_vocab.fbow" -o orb_
↪vocab.fbow

# download a sample dataset from Google Drive
FILE_ID="1d8kADKWBptEqTF7jEVhKatBEaN7g0ikY"
curl -sc /tmp/cookie "https://drive.google.com/uc?export=download&id=${FILE_ID}" > /dev/
↪null
CODE="$(awk '/_warning_/ {print $NF}' /tmp/cookie)"
curl -sLb /tmp/cookie "https://drive.google.com/uc?export=download&confirm=${CODE}&id=${
↪FILE_ID}" -o aist_living_lab_1.zip
unzip aist_living_lab_1.zip

# download a sample dataset from Google Drive
FILE_ID="1TVf2D2QvMZPHsFoTb7HNxbXclPoFMGLX"
curl -sc /tmp/cookie "https://drive.google.com/uc?export=download&id=${FILE_ID}" > /dev/
↪null
CODE="$(awk '/_warning_/ {print $NF}' /tmp/cookie)"
curl -sLb /tmp/cookie "https://drive.google.com/uc?export=download&confirm=${CODE}&id=${
↪FILE_ID}" -o aist_living_lab_2.zip
unzip aist_living_lab_2.zip

# run tracking and mapping
./run_video_slam -v ./orb_vocab.fbow -m ./aist_living_lab_1/video.mp4 -c ../example/aist/
```

(continues on next page)

(continued from previous page)

```

↪ equirectangular.yaml --frame-skip 3 --no-sleep --map-db-out map.msg
# click the [Terminate] button to close the viewer
# you can find map.msg in the current directory

# run localization
./run_video_slam --disable-mapping -v ./orb_vocab.fbow -m ./aist_living_lab_2/video.mp4 -
↪ c ../example/aist/equirectangular.yaml --frame-skip 3 --no-sleep --map-db-in map.msg

```

3.2 Sample Datasets

You can use stella_vslam with various video datasets. If you want to run stella_vslam with standard benchmarking datasets, please see [this section](#).

Start by downloading some datasets you like.

3.2.1 Equirectangular Datasets

| name | camera model | length | Google Drive | Baidu Wangpan |
|----------------------|------------------------|--------|----------------------|-----------------------------------|
| aist_entrance_hall_1 | equirectangular (mono) | 0:54 | link | link (Pass: r7r4) |
| aist_entrance_hall_2 | equirectangular (mono) | 0:54 | link | link (Pass: 4qma) |
| aist_factory_A_1 | equirectangular (mono) | 1:55 | link | link (Pass: yy2u) |
| aist_factory_A_2 | equirectangular (mono) | 1:54 | link | link (Pass: 9vey) |
| aist_factory_B_1 | equirectangular (mono) | 1:04 | link | link (Pass: gpec) |
| aist_factory_B_2 | equirectangular (mono) | 1:34 | link | link (Pass: ugrx) |
| aist_living_lab_1 | equirectangular (mono) | 2:16 | link | link (Pass: 434m) |
| aist_living_lab_2 | equirectangular (mono) | 1:47 | link | link (Pass: 549f) |
| aist_living_lab_3 | equirectangular (mono) | 2:06 | link | link (Pass: cc2p) |
| aist_stairs_A_1 | equirectangular (mono) | 2:27 | link | link (Pass: ncdt) |
| aist_stairs_B_1 | equirectangular (mono) | 2:55 | link | link (Pass: xr5t) |
| aist_store_1 | equirectangular (mono) | 1:12 | link | link (Pass: 47vq) |
| aist_store_2 | equirectangular (mono) | 1:44 | link | link (Pass: xt8u) |
| aist_store_3 | equirectangular (mono) | 1:18 | link | link (Pass: kghe) |
| ALL | equirectangular (mono) | | link | link (Pass: vsv7) |

3.2.2 Fisheye Datasets

| name | camera model | length | Google Drive | Baidu Wangpan |
|----------------------|----------------|--------|----------------------|-----------------------------------|
| aist_entrance_hall_1 | fisheye (mono) | 1:05 | link | link (Pass: u86e) |
| aist_entrance_hall_2 | fisheye (mono) | 1:06 | link | link (Pass: 9iyc) |
| aist_entrance_hall_3 | fisheye (mono) | 1:23 | link | link (Pass: qaqc) |
| aist_entrance_hall_4 | fisheye (mono) | 1:27 | link | link (Pass: em43) |
| aist_living_lab_1 | fisheye (mono) | 1:20 | link | link (Pass: wcw4) |
| aist_living_lab_2 | fisheye (mono) | 2:26 | link | link (Pass: dxns) |
| aist_living_lab_3 | fisheye (mono) | 3:43 | link | link (Pass: 7n4q) |
| nu_eng2_corridor_1 | fisheye (mono) | 2:56 | link | link (Pass: 7lws) |
| nu_eng2_corridor_2 | fisheye (mono) | 2:45 | link | link (Pass: yrtj) |
| nu_eng2_corridor_3 | fisheye (mono) | 2:04 | link | link (Pass: btpj) |
| ALL | fisheye (mono) | | link | link (Pass: gumj) |

After downloading and uncompressing a zip file, you will find a video file and a config file (old format) under the uncompressed directory.

```
$ ls dataset_name_X/
config.yaml  video.mp4
```

You can put the dataset in any directory where you have access to.

Additionally, please download a vocabulary file for FBoW from [here](#).

For the rest of this chapter, we will use aist_living_lab_1 and aist_living_lab_2 datasets for our example.

3.3 Tracking and Mapping

Here we should know how to run SLAM and create a map database file with aist_living_lab_1 dataset. You can use `./run_video_slam` to run SLAM with the video file.

```
# at the build directory of stella_vslam
$ ls
...
run_video_slam
...
$ ./run_video_slam -h
Allowed options:
  -h, --help                produce help message
  -v, --vocab arg           vocabulary file path
  -m, --video arg           video file path
  -c, --config arg          config file path
  --mask arg               mask image path
  --frame-skip arg (=1)    interval of frame skip
  --no-sleep               not wait for next frame in real time
  --auto-term              automatically terminate the viewer
  --log-level arg (=info)  log level
```

Execute the following command to run SLAM. The paths should be changed accordingly.

```
$ ./run_video_slam \  
-v /path/to/orb_vocab/orb_vocab.fbow \  
-c /path/to/stella_vslam/example/aist/equirectangular.yaml \  
-m /path/to/aist_living_lab_1/video.mp4 \  
--frame-skip 3 \  
--map-db-out aist_living_lab_1_map.msg
```

The frame viewer and map viewer should launch as well. If the two viewers are not launching correctly, check if you launched the command with the appropriate paths.



| Circumstance | Justified (%) | Not justified (%) | Don't know (%) |
|---------------------|---------------|-------------------|----------------|
| Self-defense | 100 | 0 | 0 |
| To protect property | 65 | 35 | 0 |

100

(continued from previous page)

```
- cols: 1920
- rows: 960
- color: RGB
- model: Equirectangular
ORB Configuration:
- number of keypoints: 2000
- scale factor: 1.2
- number of levels: 8
- initial fast threshold: 20
- minimum fast threshold: 7
- edge threshold: 19
- patch size: 31
- half patch size: 15
- mask rectangles:
  - [0, 1, 0, 0.1]
  - [0, 1, 0.84, 1]
  - [0, 0.2, 0.7, 1]
  - [0.8, 1, 0.7, 1]
Tracking Configuration:

[2019-05-20 17:52:41.678] [I] loading ORB vocabulary: /path/to/orb_vocab/orb_vocab.fbow
[2019-05-20 17:52:42.037] [I] startup SLAM system
[2019-05-20 17:52:42.038] [I] start local mapper
[2019-05-20 17:52:42.038] [I] start loop closer
[2019-05-20 17:52:42.395] [I] initialization succeeded with E
[2019-05-20 17:52:42.424] [I] new map created with 191 points: frame 0 - frame 2
[2019-05-20 17:53:39.092] [I] detect loop: keyframe 36 - keyframe 139
[2019-05-20 17:53:39.094] [I] pause local mapper
[2019-05-20 17:53:39.303] [I] resume local mapper
[2019-05-20 17:53:39.303] [I] start loop bundle adjustment
[2019-05-20 17:53:40.186] [I] finish loop bundle adjustment
[2019-05-20 17:53:40.186] [I] updating map with pose propagation
[2019-05-20 17:53:40.194] [I] pause local mapper
[2019-05-20 17:53:40.199] [I] resume local mapper
[2019-05-20 17:53:40.199] [I] updated map
[2019-05-20 17:55:36.218] [I] shutdown SLAM system
[2019-05-20 17:55:36.218] [I] encoding 1 camera(s) to store
[2019-05-20 17:55:36.218] [I] encoding 301 keyframes to store
[2019-05-20 17:55:37.906] [I] encoding 19900 landmarks to store
[2019-05-20 17:55:38.819] [I] save the MessagePack file of database to aist_living_lab_1_
↪map.msg
median tracking time: 0.045391[s]
mean tracking time: 0.0472221[s]
[2019-05-20 17:55:40.087] [I] clear BoW database
[2019-05-20 17:55:40.284] [I] clear map database
```

Please click the **Terminate** button to close the viewer.

After terminating, you will find a map database file aist_living_lab_1_map.msg.

```
$ ls
...
aist_living_lab_1_map.msg
```

(continues on next page)

(continued from previous page)

...

The format of map database files is [MessagePack](#), so you can reuse created maps for any third-party applications other than stella_vslam.

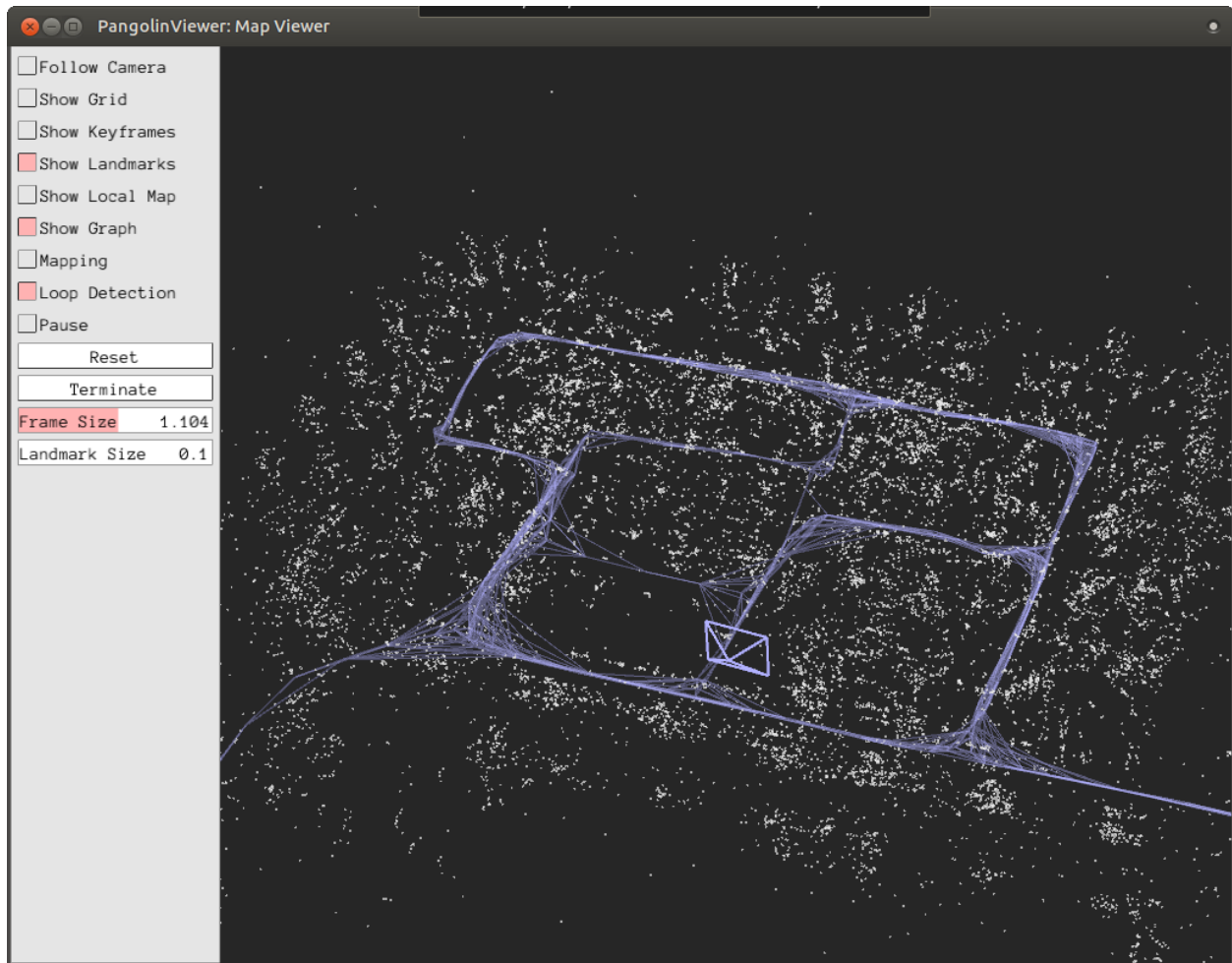
3.4 Localization

In this section, we will localize the frames in `aist_living_lab_2` dataset using the created map file `aist_living_lab_1_map.msg`. You can use `./run_video_slam` with `--map-db-in aist_living_lab_1_map.msg --disable-mapping` to run localization. Execute the following command to start localization. The paths should be changed accordingly.

```
$ ./run_video_slam --disable-mapping \  
-v /path/to/orb_vocab/orb_vocab.fbow \  
-c /path/to/stella_vslam/example/aist/equirectangular.yaml \  
-m /path/to/aist_living_lab_2/video.mp4 \  
--frame-skip 3 \  
--map-db-in aist_living_lab_1_map.msg
```

The frame viewer and map viewer should launch as well. If the two viewers are not launching correctly, check if you launched the command with the appropriate paths.

You can see if the current frame is being localized, based on the prebuild map.



```
[2019-05-20 17:58:54.728] [I] config file loaded: /path/to/stella_vslam/example/aist/
↪equirectangular.yaml
```

/ _ \ - _ _ _ _ _ \ / _ | | / _ | | \ |
 | () | ' \ -) ' \ \ v ^ _ \ | _ / - \ | \| |
 \ _ _ / | . _ ^ _ _ | | | \ / | _ _ / _ _ / \ \ | |
 | |

Copyright (C) 2019,
National Institute of Advanced Industrial Science and Technology (AIST)
All rights reserved.
For the changes after forking,
Copyright (C) 2022, stella-cv, All rights reserved.

This is free software,
and you are welcome to redistribute it under certain conditions.
See the LICENSE file.

```
Camera Configuration:
- name: RICOH THETA S 960
- setup: Monocular
- fps: 30
```

(continues on next page)

(continued from previous page)

```
- cols: 1920
- rows: 960
- color: RGB
- model: Equirectangular
ORB Configuration:
- number of keypoints: 2000
- scale factor: 1.2
- number of levels: 8
- initial fast threshold: 20
- minimum fast threshold: 7
- edge threshold: 19
- patch size: 31
- half patch size: 15
- mask rectangles:
  - [0, 1, 0, 0.1]
  - [0, 1, 0.84, 1]
  - [0, 0.2, 0.7, 1]
  - [0.8, 1, 0.7, 1]
Tracking Configuration:

[2019-05-20 17:58:54.729] [I] loading ORB vocabulary: /path/to/orb_vocab/orb_vocab.fbow
[2019-05-20 17:58:55.083] [I] clear map database
[2019-05-20 17:58:55.083] [I] clear BoW database
[2019-05-20 17:58:55.083] [I] load the MessagePack file of database from aist_living_lab_
↪ 1_map.msg
[2019-05-20 17:58:57.832] [I] decoding 1 camera(s) to load
[2019-05-20 17:58:57.832] [I] load the tracking camera "RICOH THETA S 960" from JSON
[2019-05-20 17:58:58.204] [I] decoding 301 keyframes to load
[2019-05-20 17:59:02.013] [I] decoding 19900 landmarks to load
[2019-05-20 17:59:02.036] [I] registering essential graph
[2019-05-20 17:59:02.564] [I] registering keyframe-landmark association
[2019-05-20 17:59:03.161] [I] updating covisibility graph
[2019-05-20 17:59:03.341] [I] updating landmark geometry
[2019-05-20 17:59:04.189] [I] startup SLAM system
[2019-05-20 17:59:04.190] [I] start local mapper
[2019-05-20 17:59:04.191] [I] start loop closer
[2019-05-20 17:59:04.195] [I] pause local mapper
[2019-05-20 17:59:04.424] [I] relocalization succeeded
[2019-05-20 18:01:12.387] [I] shutdown SLAM system
median tracking time: 0.0370831[s]
mean tracking time: 0.0384683[s]
[2019-05-20 18:01:12.390] [I] clear BoW database
[2019-05-20 18:01:12.574] [I] clear map database
```

If you set the `--mapping` option, the mapping module is enabled to extend the prebuild map.

RUNNING ON DOCKER

4.1 Instructions for PangolinViewer

Dockerfile.desktop can be used for easy installation. This chapter provides instructions on building and running examples with PangolinViewer support using Docker.

The instructions are tested on Ubuntu 20.04. Docker for Mac are NOT supported due to OpenGL forwarding. Please *install the dependencies manually* or use *the docker images for SocketViewer*.

Note: If you plan on using a machine with NVIDIA graphics card(s), install nvidia-docker2 and the version 390 or later of NVIDIA driver.

4.1.1 Building Docker Image

Execute the following commands:

```
git clone --recursive https://github.com/stella-cv/stella_vslam.git
cd stella_vslam
docker build -t stella_vslam-desktop -f Dockerfile.desktop .
```

You can accelerate the build of the docker image with --build-arg NUM_THREADS=<number of parallel builds> option. For example:

```
# building the docker image with four threads
docker build -t stella_vslam-desktop -f Dockerfile.desktop . --build-arg NUM_
↳ THREADS=`expr $(nproc) - 1`
```

4.1.2 Starting Docker Container

In order to enable X11 forwarding, supplemental options (-e DISPLAY=\$DISPLAY and -v /tmp/.X11-unix/:/tmp/.X11-unix:ro) are needed for docker run.

```
# before launching the container, allow display access from local users
xhost +local:
# launch the container
docker run -it --rm --privileged -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-
↳ unix:ro stella_vslam-desktop
```

Note: Additional option `--gpus all` is needed if you use NVIDIA graphics card(s).

After launching the container, the shell interface will be launched in the docker container.

```
root@ddad048b5fff:/stella_vslam/build# ls
lib                               run_image_slam                run_video_slam
run_euroc_slam                   run_kitti_slam                 run_tum_slam
```

See [Tutorial](#) to run SLAM examples in the container.

Note: If the viewer does not work, see [Trouble Shooting](#)

If you need to access to any files and directories on a host machine from the container, [bind directories](#) between the host and the container.

4.2 Instructions for SocketViewer

Dockerfile.socket and viewer/Dockerfile can be used for easy installation. This chapter provides instructions on building and running examples with SocketViewer support using Docker.

4.2.1 Building Docker Images

Docker Image of stella_vslam

Execute the following commands:

```
cd /path/to/stella_vslam
docker build -t stella_vslam-socket -f Dockerfile.socket .
```

You can accelerate the build of the docker image with `--build-arg NUM_THREADS=<number of parallel builds>` option. For example:

```
# building the docker image with four threads
docker build -t stella_vslam-socket -f Dockerfile.socket . --build-arg NUM_THREADS=`expr
↪ $(nproc) - 1`
```

Docker Image of Server

Execute the following commands:

```
cd /path/to/stella_vslam
cd viewer
docker build -t stella_vslam-viewer .
```

4.2.2 Starting Docker Containers

On Linux

Launch the server container and access to it with the web browser in advance. Please specify `--net=host` in order to share the network with the host machine.

```
$ docker run --rm -it --name stella_vslam-viewer --net=host stella_vslam-viewer
WebSocket: listening on *:3000
HTTP server: listening on *:3001
```

After launching, access to `http://localhost:3001/` with the web browser.

Next, launch the container of `stella_vslam`. The shell interface will be launched in the docker container.

```
$ docker run --rm -it --name stella_vslam-socket --net=host stella_vslam-socket
root@hostname:/stella_vslam/build#
```

See [Tutorial](#) to run SLAM examples in the container.

If you need to access to any files and directories on a host machine from the container, [bind directories](#) between the host and the container.

On macOS

Launch the server container and access to it with the web browser in advance. Please specify `-p 3001:3001` for port-forwarding.

```
$ docker run --rm -it --name stella_vslam-viewer -p 3001:3001 stella_vslam-viewer
WebSocket: listening on *:3000
HTTP server: listening on *:3001
```

After launching, access to `http://localhost:3001/` with the web browser.

Then, inspect the container's IP address and append the `SocketPublisher.server_uri` entry to the YAML config file of `stella_vslam`.

```
# inspect the server's IP address
$ docker inspect stella_vslam-viewer | grep -m 1 "IPAddress" | sed 's/ //g' | sed 's/,/↵/g'
"IPAddress": "172.17.0.2"
```

```
# config file of stella_vslam

...

#####
# SocketPublisher Parameters #
#####

# append this entry
SocketPublisher.server_uri: "http://172.17.0.2:3000"
```

Next, launch the container of `stella_vslam`. The shell interface will be launched in the docker container.

```
$ docker run --rm -it --name stella_vslam-socket stella_vslam-socket
root@hostname:/stella_vslam/build#
```

See [Tutorial](#) to run SLAM examples in the container.

Please don't forget to append `SocketPublisher.server_uri` entry to the `config.yaml` if you use the downloaded datasets in the tutorial.

If you need to access to any files and directories on a host machine from the container, [bind directories](#) between the host and the container.

4.3 Bind of Directories

If you need to access to any files and directories on a host machine from the container, bind directories between the host and the container using `--volume` or `--mount` option. (See [the docker documentataion](#).)

For example:

```
# launch a container of stella_vslam-desktop with --volume option
$ docker run -it --rm --runtime=nvidia -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-
↪unix:ro \
    --volume /path/to/dataset/dir:/dataset:ro \
    --volume /path/to/vocab/dir:/vocab:ro \
    stella_vslam-desktop
# dataset/ and vocab/ are found at the root directory in the container
root@0c0c9f115d74:/# ls /
... dataset/ vocab/ ...
```

```
# launch a container of stella_vslam-socket with --volume option
$ docker run --rm -it --name stella_vslam-socket --net=host \
    --volume /path/to/dataset/dir:/dataset:ro \
    --volume /path/to/vocab/dir:/vocab:ro \
    stella_vslam-socket
# dataset/ and vocab/ are found at the root directory in the container
root@0c0c9f115d74:/# ls /
... dataset/ vocab/ ...
```


RUNNING ON ROS

We provide ROS and ROS2 package examples to help you run `stella_vslam` on ROS framework.

5.1 ROS Package

5.1.1 Installation

Requirements

- `ROS` : `noetic`
- `stella_vslam`
- `image_transport` : Required by this ROS package examples.
- `cv_bridge` : Please build it with the same version of OpenCV used in `stella_vslam`.

Prerequisites

Tested for **Ubuntu 20.04**.

Please install the following dependencies.

- `ROS` : Please follow [Installation of ROS](#).
- `stella_vslam` : Please follow [Installation of stella_vslam](#).

Note: Please build `stella_vslam` with PangolinViewer or SocketViewer if you plan on using it for the examples.

Install the dependencies via `apt`.

```
apt update -y
apt install ros-{ROS_DISTRO}-image-transport
```

Download the source of `cv_bridge`.

```
mkdir -p ~/catkin_ws/src
git clone --branch {ROS_DISTRO} --depth 1 https://github.com/ros-perception/vision_
↪opencv.git
cp -r vision_opencv/cv_bridge ~/catkin_ws/src
rm -rf vision_opencv
```

Build Instructions

When building with support for PangolinViewer, please specify the following cmake options: `-DUSE_PANGOLIN_VIEWER=ON` and `-DUSE_SOCKET_PUBLISHER=OFF` as described in [build of stella_vslam](#). `stella_vslam` and `stella_vslam_ros` need to be built with the same options.

```
cd ~/catkin_ws/src
git clone --recursive --branch ros --depth 1 https://github.com/stella-cv/stella_vslam_
↪ros.git
cd ~/catkin_ws
catkin_make -DUSE_PANGOLIN_VIEWER=ON -DUSE_SOCKET_PUBLISHER=OFF
```

5.1.2 Examples

Run the core program required for ROS-based system in advance.

```
roscore
```

Note: Please leave the **roscore** run.

Publisher

Publish Images by a video

```
roslaunch image_publisher image_publisher ./aist_living_lab_1/video.mp4 /image_raw:=/camera/
↪image_raw
```

Publish Images of a USB Camera

For using a standard USB camera for visual SLAM or localization.

```
apt install ros-${ROS_DISTRO}-usb-cam
```

```
rosparam set usb_cam/pixel_format yuyv
roslaunch usb_cam usb_cam_node
```

Republish the ROS topic to `/camera/image_raw`.

```
roslaunch image_transport republish \
  raw in:=/usb_cam/image_raw raw out:=/camera/image_raw
```

Subscriber

Subscribers continually receive images. Please execute one of the following command snippets in the new terminal.

Note: Option arguments are the same as *the examples of stella_vslam*.

Tracking and Mapping

We provide an example snippet for visual SLAM. The source code is placed at `stella_vslam_ros/src/run_slam.cc`.

```
source ~/catkin_ws/devel/setup.bash
roslaunch stella_vslam_ros run_slam \
  -v /path/to/orb_vocab.fbow \
  -c /path/to/config.yaml \
  --map-db-out /path/to/map.msg
```

Localization

We provide an example snippet for localization based on a prebuilt map. The source code is placed at `stella_vslam_ros/src/run_slam.cc`.

```
source ~/catkin_ws/devel/setup.bash
roslaunch stella_vslam_ros run_slam \
  --disable-mapping \
  -v /path/to/orb_vocab.fbow \
  -c /path/to/config.yaml \
  --map-db-in /path/to/map.msg
```

5.2 ROS2 Package

5.2.1 Installation

Requirements

- **ROS2** : foxy or later.
- *stella_vslam*
- **image_common** : Required by this ROS package examples.
- **vision_opencv** : Please build it with the same version of OpenCV used in *stella_vslam*.
- **image_tools** : An optional requirement to use USB cameras.

Prerequisites

Tested for **Ubuntu 20.04**.

Please install the following dependencies.

- ROS2 : Please follow [Installation of ROS2](#).
- stella_vslam : Please follow [Installation of stella_vslam](#).

Note: Please build stella_vslam with PangolinViewer or SocketViewer if you plan on using it for the examples.

Download repositories of image_common and vision_opencv.

```
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws/src
git clone -b ${ROS_DISTRO} --single-branch https://github.com/ros-perception/image_
↪common.git
git clone -b ros2 --single-branch https://github.com/ros-perception/vision_opencv.git
```

For using USB cam as a image source, donload a repository of demos and pick image_tools module.

```
cd ~/ros2_ws
git clone https://github.com/ros2/demos.git
cp -r demos/image_tools src/
rm -rf demos
```

Build Instructions

When building with support for PangolinViewer, please specify the following cmake options: -DUSE_PANGOLIN_VIEWER=ON and -DUSE_SOCKET_PUBLISHER=OFF as described in [build of stella_vslam](#). stella_vslam and stella_vslam_ros need to be built with the same options.

```
cd ~/ros2_ws/src
git clone --recursive --branch ros2 --depth 1 https://github.com/stella-cv/stella_vslam_
↪ros.git
cd ~/ros2_ws
colcon build --symlink-install --cmake-args -DUSE_PANGOLIN_VIEWER=ON -DUSE_SOCKET_
↪PUBLISHER=OFF
```

5.2.2 Examples

Publisher

Publish Images by a video

```
ros2 run image_publisher image_publisher_node ./aist_living_lab_1/video.mp4 --ros-args --
↪remap /image_raw:=/camera/image_raw
```

Publish Images Captured by a USB Camera

For using a standard USB camera for visual SLAM or localization.

```
ros2 run image_tools cam2image
```

Republish the ROS topic to /camera/image_raw.

```
ros2 run image_transport republish \  
  raw in:=image raw out:=/camera/image_raw
```

Subscriber

Subscribers continually receive images. Please execute one of the following command snippets in the new terminal.

Note: Option arguments are the same as *the examples of stella_vslam*.

Tracking and Mapping

We provide an example snippet for visual SLAM. The source code is placed at `stella_vslam_ros/src/run_slam.cc`.

```
source ~/ros2_ws/install/setup.bash  
ros2 run stella_vslam_ros run_slam \  
  -v /path/to/orb_vocab.fbow \  
  -c /path/to/config.yaml \  
  --map-db-out /path/to/map.msg
```

Localization

We provide an example snippet for localization based on a prebuilt map. The source code is placed at `stella_vslam_ros/src/run_slam.cc`.

```
source ~/ros2_ws/install/setup.bash  
ros2 run stella_vslam_ros run_slam \  
  --disable-mapping \  
  -v /path/to/orb_vocab.fbow \  
  -c /path/to/config.yaml \  
  --map-db-in /path/to/map.msg
```


EXAMPLE

We provided example code snippets for running `stella_vslam` with variety of datasets.

6.1 SLAM with Video Files

We provide an example snippet for using video files (e.g. `.mp4`) for visual SLAM. The source code is placed at `./example/run_video_slam.cc`.

The camera that captures the video file must be calibrated. Create a config file (`.yaml`) according to the camera parameters.

We provided a vocabulary file for FBoW at [here](#).

You can create a map database file by running one of the `run_****_slam` executables with `--map-db-out map_file_name.msg` option.

6.2 SLAM with Image Sequences

We provided an example snippet for using image sequences for visual SLAM. The source code is placed at `./example/run_image_slam.cc`.

The camera that captures the video file must be calibrated. Create a config file (`.yaml`) according to the camera parameters.

We provided a vocabulary file for FBoW at [here](#).

You can create a map database file by running one of the `run_****_slam` executables with `--map-db-out map_file_name.msg` option.

6.3 SLAM with Standard Datasets

6.3.1 KITTI Odometry dataset

KITTI Odometry dataset is a benchmarking dataset for monocular and stereo visual odometry and lidar odometry that is captured from car-mounted devices. We provided an example source code for running monocular and stereo visual SLAM with this dataset. The source code is placed at `./example/run_kitti_slam.cc`.

Start by downloading the dataset from [here](#). Download the grayscale set (`data_odometry_gray.zip`).

After downloading and uncompressing it, you will find several sequences under the `sequences/` directory.

```
$ ls sequences/
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21
```

In addition, download a vocabulary file for FBoW from [here](#).

A configuration file for each sequence is contained under `./example/kitti/`.

If you built examples with Pangolin Viewer support, a map viewer and frame viewer will be launched right after executing the following command.

```
# at the build directory of stella_vslam
# monocular SLAM with sequence 00
$ ./run_kitti_slam \
  -v /path/to/orb_vocab/orb_vocab.fbow \
  -d /path/to/KITTI/Odometry/sequences/00/ \
  -c ../example/kitti/KITTI_mono_00-02.yaml
# stereo SLAM with sequence 05
$ ./run_kitti_slam \
  -v /path/to/orb_vocab/orb_vocab.fbow \
  -d /path/to/KITTI/Odometry/sequences/05/ \
  -c ../example/kitti/KITTI_stereo_04-12.yaml
```

6.3.2 EuRoC MAV dataset

EuRoC MAV dataset is a benchmarking dataset for monocular and stereo visual odometry that is captured from drone-mounted devices. We provide an example source code for running monocular and stereo visual SLAM with this dataset. The source code is placed at `./example/run_euroc_slam.cc`.

Start by downloading the dataset from [here](#). Download the `.zip` file of a dataset you plan on using.

After downloading and uncompressing it, you will find several directories under the `mav0/` directory.

```
$ ls mav0/
body.yaml cam0 cam1 imu0 leica0 state_groundtruth_estimate0
```

In addition, download a vocabulary file for FBoW from [here](#).

We provided the two config files for EuRoC, `./example/euroc/EuRoC_mono.yaml` for monocular and `./example/euroc/EuRoC_stereo.yaml` for stereo.

If you have built examples with Pangolin Viewer support, a map viewer and frame viewer will be launched right after executing the following command.


```
# at the build directory of stella_vslam
# monocular SLAM with any EuRoC sequence
$ ./run_euroc_slam \
  -v /path/to/orb_vocab/orb_vocab.fbow \
  -d /path/to/EuRoC/MAV/mav0/ \
  -c ../example/euroc/EuRoC_mono.yaml
# stereo SLAM with any EuRoC sequence
$ ./run_euroc_slam \
  -v /path/to/orb_vocab/orb_vocab.fbow \
  -d /path/to/EuRoC/MAV/mav0/ \
  -c ../example/euroc/EuRoC_stereo.yaml
```

6.3.3 TUM RGBD dataset

TUM RGBD dataset is a benchmarking dataset containing RGB-D data and ground-truth data with the goal to establish a novel benchmark for the evaluation of visual odometry and visual SLAM systems. The source code is placed at `./example/run_tum_rgbd_slam.cc`.

Start by downloading the various dataset from [here](#). One of many example datasets can be found from [here](#). Download the .tgz file of a dataset you plan on using.

After downloading and uncompressing it, you will find two directories and few text files under the `rgb_dataset_freiburg3_calibration_rgb_depth/` directory.

```
$ ls rgb_dataset_freiburg3_calibration_rgb_depth
accelerometer.txt depth depth.txt groundtruth.txt rgb rgb.txt
```

If you would like to preprocess dataset then you can use tool from [here](#).

In addition, download a vocabulary file for FBoW from [here](#).

We provided the config files for RGBD dataset at, `./example/tum_rgbd`.

For above specific example we shall use two config files, `./example/tum_rgbd/TUM_RGBD_mono_3.yaml` for monocular and `./example/tum_rgbd/TUM_RGBD_rgbd_3.yaml` for RGBD.

6.3.4 Tracking and Mapping

```
# at the build directory of stella_vslam
# monocular SLAM with rgb_dataset_freiburg3_calibration_rgb_depth
$ ./run_tum_rgbd_slam \
  -v /path/to/orb_vocab/orb_vocab.fbow \
  -d /path/to/rgb_dataset_freiburg3_calibration_rgb_depth/ \
  -c ../example/tum_rgbd/TUM_RGBD_mono_3.yaml \
  --no-sleep \
  --auto-term \
  --map-db-out fr3_slam_mono.msg

# RGBD SLAM with rgb_dataset_freiburg3_calibration_rgb_depth
$ ./run_tum_rgbd_slam \
  -v /path/to/orb_vocab/orb_vocab.fbow \
  -d /path/to/rgb_dataset_freiburg3_calibration_rgb_depth/ \
  -c ../example/tum_rgbd/TUM_RGBD_rgbd_3.yaml \
```

(continues on next page)

(continued from previous page)

```
--no-sleep \  
--auto-term \  
--map-db-out fr3_slam_rgbd.msg
```

6.3.5 Localization

```
# at the build directory of stella_vslam  
# monocular localization with rgb_dataset_freiburg3_calibration_rgb_depth  
$ ./run_tum_rgbd_slam --disable-mapping \  
-v /path/to/orb_vocab/orb_vocab.fbow \  
-d /path/to/rgb_dataset_freiburg3_calibration_rgb_depth/ \  
-c ../example/tum_rgbd/TUM_RGBD_mono_3.yaml \  
--no-sleep \  
--auto-term \  
--map-db-in fr3_slam_mono.msg  
  
# RGBD SLAM with rgb_dataset_freiburg3_calibration_rgb_depth  
$ ./run_tum_rgbd_slam --disable-mapping \  
-v /path/to/orb_vocab/orb_vocab.fbow \  
-d /path/to/rgb_dataset_freiburg3_calibration_rgb_depth/ \  
-c ../example/tum_rgbd/TUM_RGBD_rgbd_3.yaml \  
--no-sleep \  
--auto-term \  
--map-db-in fr3_slam_rgbd.msg
```

6.4 SLAM with UVC camera

6.4.1 Tracking and Mapping

We provided an example snippet for using a UVC camera, which is often called a webcam, for visual SLAM. The source code is placed at `./example/run_camera_slam.cc`.

Please specify the camera number you want to use by `-n` option.

The camera must be calibrated. Create a config file (`.yaml`) according to the camera parameters.

You can scale input images to the performance of your machine by `-s` option. Please modify the config accordingly.

We provided a vocabulary file for FBoW at [here](#).

PARAMETERS

7.1 System

| Name | Description |
|--------------------|--|
| map_format | msgpack or sqlite3 |
| min_num_shared_lms | minimum number of shared landmarks for covisibility graph connection. Keyframes that exceed the threshold are considered to share a sufficient number of landmarks and are used for local BA, loop detection and graph optimization. |

7.2 Camera

| Name | Description |
|--------------------|---|
| name | It is used by the camera database to identify the camera. |
| setup | monocular, stereo, RGBD |
| model | perspective, fisheye, equirectangular, radial_division (note: If you want to use stereo_rectifier, you need to specify perspective.) |
| fx, fy | Focal length (pixel) |
| cx, cy | Principal point (pixel) |
| k1, k2, p1, p2, k3 | Distortion parameters for perspective camera. When using StereoRectifier, there is no distortion after stereo rectification. |
| k1, k2, k3, k4 | Distortion parameters for fisheye camera |
| distortion | Distortion parameters for radial_division camera |
| fps | Framerate of input images |
| cols, rows | Resolution (pixel) |
| color_order | Gray, RGB, RGBA, BGR, BGRA |
| focal_x_baseline | For stereo cameras, it is the value of the baseline between the left and right cameras multiplied by the focal length fx. For RGBD cameras, if the measurement method is stereo, set it based on its baseline. If the measurement method is other than that, set the appropriate value based on the relationship between depth accuracy and baseline. |
| depth_threshold | The ratio used to determine the depth threshold. |

7.3 Feature

| Name | Description |
|--------------------|--|
| name | name of ORB feature extraction model (id for saving) |
| scale_factor | Scale of the image pyramid |
| num_levels | Number of levels of in the image pyramid |
| ini_fast_threshold | FAST threshold for try first |
| min_fast_threshold | FAST threshold for try second time |

7.4 Preprocessing

| Name | Description |
|-----------------|--|
| min_size | Size of node occupied by one feature point. The larger this value, the fewer feature points are extracted. |
| depthmap_factor | The ratio used to convert depth image pixel values to distance. |

7.5 Tracking

| Name | Description |
|---|--|
| reloc_distance_threshold | Maximum distance threshold (in meters) where close keyframes could be found when doing a relocalization by pose. |
| reloc_angle_threshold | Maximum angle threshold (in radians) between given pose and close keyframes when doing a relocalization by pose. |
| enable_auto_relocalization | If true, automatically try to relocalize when lost. |
| use_robust_matcher_for_relocalization_request | If true, use robust matcher for relocalization request. |
| max_num_local_keyfrms | Max number of local keyframes for tracking. |
| backend | g2o or gtsam |
| init_retry_threshold_time | If tracking is failed within init_retry_threshold_time sec after initialization, reset the system |
| enable_temporal_keyframe_only_tracking | If true, tracking with only temporal keyframes will not be treated as Lost |

7.6 Mapping

| Name | Description |
|--|---|
| baseline_dist_thr_ratio | For two frames of baseline below the threshold, no triangulation will be performed. In the monocular case, the scale is indefinite, so relative values are recommended. Either baseline_dist_thr or this one should be specified. If not specified, baseline_dist_thr_ratio will be used. |
| baseline_dist_thr | For two frames of baseline below the threshold, no triangulation will be performed. |
| redundant_obs_ratio_thr | |
| observed_ratio_thr | |
| num_reliable_keyfrms | |
| enable_interruption_of_landmark_generation | If true, enable interruption of landmark generation |
| enable_interruption_before_local_BA | If true, enable interruption before local BA |
| backend | g2o or gtsam |
| erase_temporal_keyframes | If true, remove keyframes past num_temporal_keyframes |
| num_temporal_keyframes | Number of temporal keyframes |

7.7 StereoRectifier

| Name | Description |
|-----------------|--|
| model | camera model type before rectification. The option is perspective or fisheye. (note: If you want to use fisheye model for stereo_rectifier, you need to specify Camera::model to perspective.) |
| K_left, K_right | Intrinsic parameters. The 3x3 matrix are written in row-major order. |
| D_left, D_right | Distortion parameters. The 5 parameters are k1, k2, p1, p2, k3. |
| R_left, R_right | Stereo-rectification parameters. The 3x3 matrix are written in row-major order. |

7.8 Initializer

| Name | Description |
|------------------------------|---|
| num_ransac_iterations | max number of iterations of RANSAC (only for monocular initializer) |
| min_num_valid_pts | min number of valid pts (It should be greater than or equal to min_num_triangulated_) |
| min_num_triangulated_pts | Minimum number of triangulated points |
| parallax_deg_threshold | min parallax (only for monocular initializer) |
| reprojection_error_threshold | reprojection error threshold (only for monocular initializer) |
| num_ba_iterations | max number of iterations of BA (only for monocular initializer) |
| scaling_factor | initial scaling factor (only for monocular initializer) |
| use_fixed_seed | Use fixed random seed for RANSAC if true |

7.9 Relocalizer

| Name | Description |
|-----------------------|-------------|
| bow_match_lowe_ratio | |
| proj_match_lowe_ratio | |
| min_num_bow_matches | |
| min_num_valid_obs | |

7.10 KeyframeInserter

| Name | Description |
|--|--|
| max_interval | max interval to insert keyframe |
| min_interval | |
| max_distance | |
| enough_lms_thr | |
| lms_ratio_thr_almost_all_lms_threshold | Threshold at which we consider that we are tracking almost all landmarks. Ratio-threshold of “the number of 3D points observed in the current frame” / “that of 3D points observed in the last keyframe” |
| lms_ratio_thr_view_changed | Threshold at which we consider the view to have changed. Ratio-threshold of “the number of 3D points observed in the current frame” / “that of 3D points observed in the last keyframe” |

7.11 PangolinViewer

| Name | Description |
|--|-------------|
| keyframe_size | |
| keyframe_line_width | |
| graph_line_width | |
| point_size | |
| camera_size | |
| camera_line_width | |
| menu_width | |
| viewpoint_x, viewpoint_y, viewpoint_z, viewpoint_f | |

7.12 SocketPublisher

| Name | Description |
|-------------------|---|
| emitting_interval | |
| image_quality | |
| server_uri | |
| max_num_keyframes | Limit the number of keyframes transferred at one time. This avoids disconnections when loading large maps. |
| max_num_landmarks | Limit the number of landmarks transferred at one time. This avoids disconnections when loading large maps. |
| publish_points | If true, pointcloud transfer is enabled. The default is true. Pointcloud transfer is slow, so disabling pointcloud transfer may be useful to improve performance of SocketViewer. |

7.13 LoopDetector

| Name | Description |
|--------------------------------|---|
| enabled | flag which indicates the loop detector is enabled or not |
| num_final_matches_threshold | the threshold of the number of mutual matches after the Sim3 estimation |
| min_continuity | the threshold of the continuity of continuously detected keyframe set |
| reject_by_graph_distance | If true, reject by distance on essential graph |
| loop_min_distance_on_graph | Minimum distance to allow for loop candidates |
| top_n_covisibilities_to_search | Top n covisibilities to search (0 means disabled) |
| num_matches_thr | Minimum number of matches to allow for loop candidates |
| num_matches_thr_brute_force | Minimum number of matches to allow for loop candidates after brute force matching. (0 means disabled) |
| num_optimized_inliers_thr | Minimum number of matches to allow for loop candidates after optimization by transform_optimizer |
| backend | g2o or gtsam |

7.14 MarkerModel

| Name | Description |
|-------------|---|
| type | Only “aruco” is a valid value |
| width | Physical size of marker |
| marker_size | 4, 5, 6, 7. See https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html . |
| max_markers | 50, 100, 250, 1000. See https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html . |

RELOCALIZATION

8.1 What is Relocalization? Why it is needed?

In Visual SLAM, the robot/camera explores its environment while

1. estimates its location using the map and the last location as prior information (Tracking), and simultaneously
2. update the map (the database that records landmarks) of environment (Mapping).

Relocalization module can estimate the location without using any prior information other than the map (with the high cost of computation). This is useful when the previous location cannot be used as prior information, for example when tracking fails.

8.2 Steps in Relocalization

1. Acquire relocalization candidate keyframes (Used as a reference keyframe for relocalization)
2. Compute matching points for each candidate by using BoW tree matcher
3. Discard the candidate if the number of 2D-3D matches is less than the threshold
4. Setup an PnP solver with the current 2D-3D matches
5. Estimate the camera pose using EPnP (+ RANSAC)
6. Apply pose optimizer
7. Apply projection match to increase 2D-3D matches
8. Re-apply the pose optimizer
9. Apply projection match again if the number of the observations is less than the threshold
10. Apply projection match again, then set the 2D-3D matches
11. Discard if the number of the observations is less than the threshold and do the pose estimation again
12. If the number of observation is greater than threshold succeed in relocalization

See the details on how to run the relocalization at [here](#).

TROUBLE SHOOTING

9.1 For building

1. stella_vslam terminates abnormally soon after **launching** or **optimization with g2o**.

Please configure and rebuild g2o and stella_vslam with `-DBUILD_WITH_MARCH_NATIVE=OFF` option for `cmake`.

9.2 For Viewer

1. (on Docker) Additional option `--gpus all` is needed if you use NVIDIA graphics card(s). Please see [here](#) for more details.
2. If the viewer does not work, try *the docker images for SocketViewer* instead.

9.3 For SLAM

CITATION OF ORIGINAL VERSION OF OPENVSLAM (XDSPACE/LAB/OPENVSLAM)

```
@inproceedings{openvslam2019,  
  author = {Sumikura, Shinya and Shibuya, Mikiya and Sakurada, Ken},  
  title = {{OpenVSLAM: A Versatile Visual SLAM Framework}},  
  booktitle = {Proceedings of the 27th ACM International Conference on Multimedia},  
  series = {MM '19},  
  year = {2019},  
  isbn = {978-1-4503-6889-6},  
  location = {Nice, France},  
  pages = {2292--2295},  
  numpages = {4},  
  url = {http://doi.acm.org/10.1145/3343031.3350539},  
  doi = {10.1145/3343031.3350539},  
  acmid = {3350539},  
  publisher = {ACM},  
  address = {New York, NY, USA}  
}
```