ORB_SLAM3 compilation and running

Compilation environment

Motherboard: Jetson NX 16G

System: ubuntu20.04

Jetpack: 5.1.1

Python: 3.8.10

CUDA: 11.4.15

System opencv: 4.5.4

ROS: noetic

Install and compile PANGOLIN

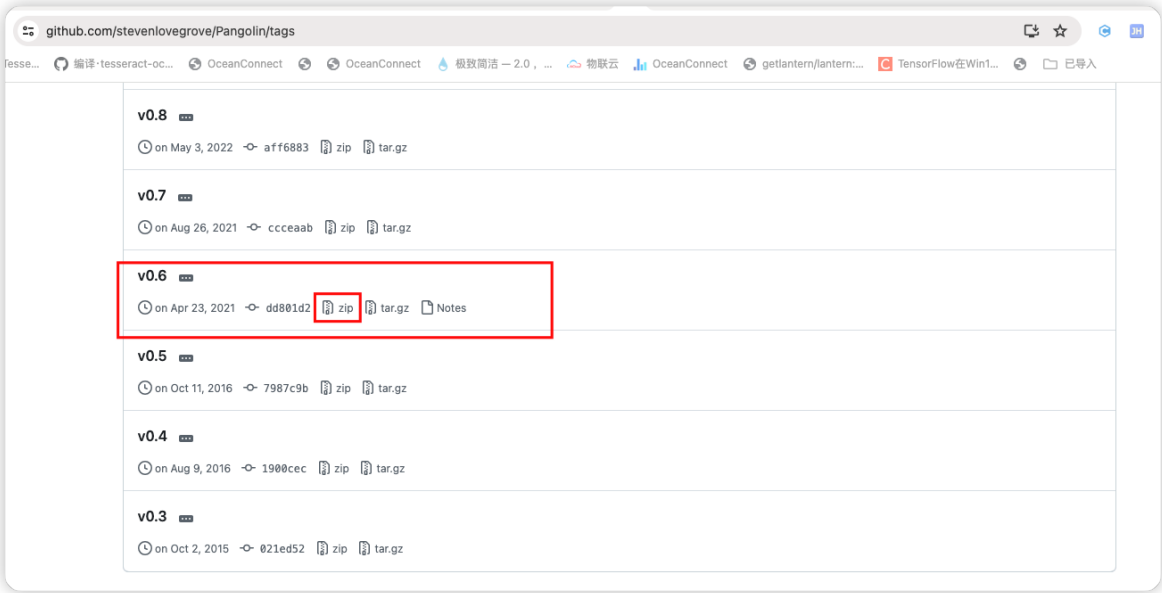
1. Download source code

```
git clone -b v0.6 https://github.com/stevenlovegrove/Pangolin.git
```

Or download the compressed package directly

```
https://github.com/stevenlovegrove/Pangolin/tags
```

Select v0.6



2. Unzip the source code

Unzip the downloaded Pangolin compressed package. Assume that the path of the Pangolin compressed package is

/home/youname/Pangolin.zip Enter the following command in the terminal:

```
unzip /home/youname/Pangolin.zip cannot be copied directly, and must be unzipped
according to your own path.
```

```
cd Pangolin
```

```
mkdir build && cd build
```

```
cmake ..
```

```
make -j8
```

```
sudo make install
```

Install and compile OpenCV

orbslam3 downloaded from GitHub requires opencv version 4.4, but the default version installed in the Jetson NX 16G image is 4.5.4. There is no problem compiling orbslam3 itself, but when compiling the ROS version, cv_bridge will be used, and the opencv version of cv_bridge based on the noetic version is 4.2, so there will be some conflicts during the compilation process.

Install boost library

Check boost version

```
dpkg -S /usr/include/boost/version.hpp
```

Normally, you don't need to install boost library through source code, just enter the command in the terminal:

```
sudo apt install libboost-dev
```

Install Eigen 3

```
sudo apt install libeigen3-dev
```

Install openssl-devel

```
sudo apt install libssl-dev
```

Install Obi Zhongguang camera driver

```
sudo apt install libgflags-dev ros-$ROS_DISTRO-image-geometry ros-$ROS_DISTRO-
camera-info-manager \
ros-$ROS_DISTRO-image-transport ros-$ROS_DISTRO-image-publisher libgoogle-glog-
dev libusb-1.0-0-dev libeigen3-dev \
ros-$ROS_DISTRO-diagnostic-updater ros-$ROS_DISTRO-diagnostic-msgs
```

Download the Orbslam driver in your workspace, assuming your workspace path is

```
~/orbslam3_ws
```

Enter the command in the terminal

```
mkdir -p ~/orbslam3_ws/src
cd ~/ros_ws/src
git clone https://github.com/orbbec/OrbbecSDK_ROS1.git
```

Build the package:

```
cd ~/orbslam3_ws
catkin build
```

Install udev rules:

```
cd ~/ros_ws
source ./devel/setup.bash
roscd orbbec_camera
cd scripts
sudo cp 99-obsensor-libusb.rules /etc/udev/rules.d/99-obsensor-libusb.rules
sudo udevadm control --reload && sudo udevadm trigger
```

Start the camera:

```
source ./devel/setup.bash
roslaunch orbbec_camera gemini2.launch #gemin2 camera
roslaunch orbbec_camera gemini_330_series.launch #gemin_335 camera
```

Compile SLAM3 source code

1. Download source code

```
git clone https://github.com/UZ-SLAMLab/ORB_SLAM3.git
```

2. Modify the opencv version number in CMakeList.txt

The default opencv version of the source code downloaded by Orbslam3 is 4.4, which is relatively low for Jetson NX16. However, since the opencv in notetic is opencv4.2, we still need to compile cv_bridge with the source code when compiling the ros package of orbslam3 later. It is better to configure opencv based on version 4.4.

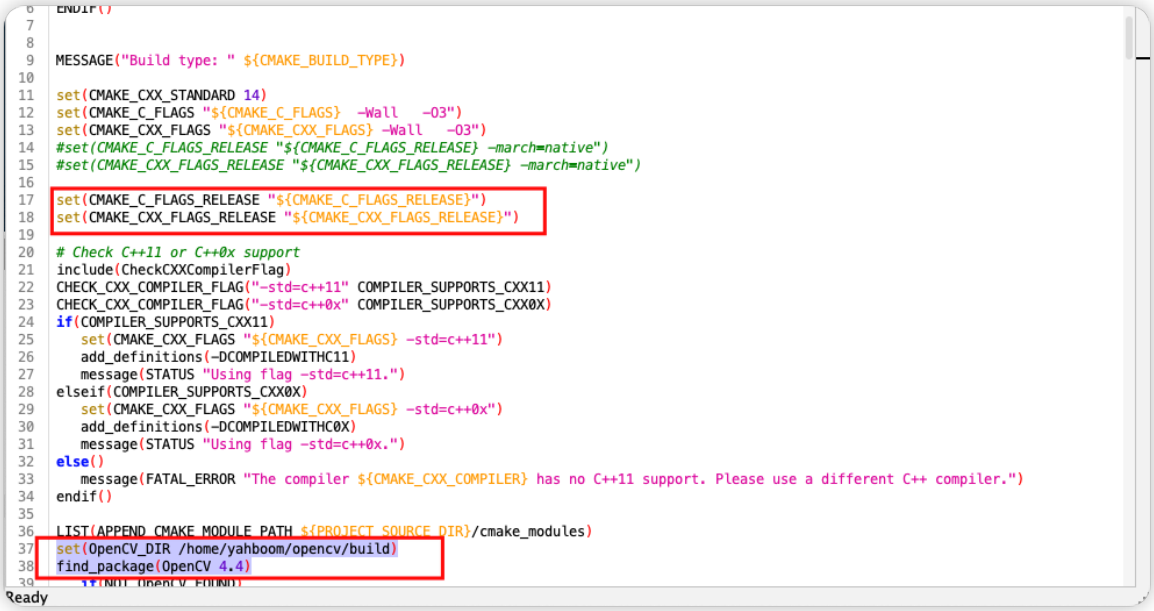
Modify the cmakelist file of orbslam3, as shown below

```
#cmakelist file path
~/your orbslam source code path/CMakeLists.txt
```

```
set(OpenCV_DIR /home/yōuname/opencv/build) ##The path here is your opencv
compilation path, which cannot be copied directly

find_package(OpenCV 4.4)
```

You can modify it by the way

```
set(CMAKE_C_FLAGS_RELEASE "${CMAKE_C_FLAGS_RELEASE} -march=native")
set(CMAKE_CXX_FLAGS_RELEASE "${CMAKE_CXX_FLAGS_RELEASE} -march=native")

#Change to
set(CMAKE_C_FLAGS_RELEASE "${CMAKE_C_FLAGS_RELEASE}")
set(CMAKE_CXX_FLAGS_RELEASE "${CMAKE_CXX_FLAGS_RELEASE}")

Also add
set(CMAKE_CXX_STANDARD 14)
```



Modify

```
~/your orbslam3 source code path/Thirdparty/DBoW2/CMakeLists.txt
```

Also change the opencv version to 4.4, and set your opencv installation path, as shown below,

```
#Also
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wall -O3 -march=native ")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -O3 -march=native")

#Change to
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wall -O3 ")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -O3 ")
```

```
  set(CMAKE_BUILD_TYPE Release)
endif()

#set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS}  -Wall  -O3 -march=native ")
#set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall  -O3 -march=native")

set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS}  -Wall  -O3 ")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall  -O3 ")

set(HDRS_DBOW2
  DBoW2/BowVector.h
  DBoW2/FORB.h
  DBoW2/FClass.h
  DBoW2/FeatureVector.h
  DBoW2/ScoringObject.h
  DBoW2/TemplatedVocabulary.h)
set(SRCS_DBOW2
  DBoW2/BowVector.cpp
  DBoW2/FORB.cpp
  DBoW2/FeatureVector.cpp
  DBoW2/ScoringObject.cpp)

set(HDRS_DUTILS
  DUtils/Random.h
  DUtils/Timestamp.h)
set(SRCS_DUTILS
  DUtils/Random.cpp
  DUtils/Timestamp.cpp)


set(OpenCV_DIR /home/yahboom/opencv/build)
find_package(OpenCV 4.4 REQUIRED)
if(NOT OpenCV_FOUND)
   find_package(OpenCV 3.0 QUIET)
   if(NOT OpenCV_FOUND)
      message(FATAL_ERROR "OpenCV > 3.0 not found.")
   endif()
endif()

set(LIBRARY_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/lib)

include_directories(${OpenCV_INCLUDE_DIRS})
add_library(DBoW2 SHARED ${SRCS_DBOW2} ${SRCS_DUTILS})
target_link_libraries(DBoW2 ${OpenCV_LIBS})
```

Modify

```
~/your orbslam3 source path/Thirdparty/g2o/CMakeLists.txt
```

As shown below:

navrobo_nx_CMakeLists(10).txt

```
# Eigen library parallelise itself, though, presumably due to performance issues
# OPENMP is experimental. We experienced some slowdown with it
FIND_PACKAGE(OpenMP)
SET(G2O_USE_OPENMP OFF CACHE BOOL "Build g2o with OpenMP support (EXPERIMENTAL)")
IF(OPENMP_FOUND AND G2O_USE_OPENMP)
  SET (G2O_OPENMP 1)
  SET(g2o_C_FLAGS "${g2o_C_FLAGS} ${OpenMP_C_FLAGS}")
  SET(g2o_CXX_FLAGS "${g2o_CXX_FLAGS} -DEIGEN_DONT_PARALLELIZE ${OpenMP_CXX_FLAGS}")
  MESSAGE(STATUS "Compiling with OpenMP support")
ENDIF(OPENMP_FOUND AND G2O_USE_OPENMP)

# Compiler specific options for gcc
#SET(CMAKE_CXX_FLAGS_RELEASE "${CMAKE_CXX_FLAGS_RELEASE} -O3 -march=native")
#SET(CMAKE_C_FLAGS_RELEASE "${CMAKE_C_FLAGS_RELEASE} -O3 -march=native")

SET(CMAKE_CXX_FLAGS_RELEASE "${CMAKE_CXX_FLAGS_RELEASE} -O3")
SET(CMAKE_C_FLAGS_RELEASE "${CMAKE_C_FLAGS_RELEASE} -O3")


# activate warnings !!!
SET(g2o_C_FLAGS "${g2o_C_FLAGS} -Wall -W")
SET(g2o_CXX_FLAGS "${g2o_CXX_FLAGS} -Wall -W")

# specifying compiler flags
SET(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} ${g2o_CXX_FLAGS}")
SET(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} ${g2o_C_FLAGS}")

# Find Eigen3
SET(EIGEN3_INCLUDE_DIR ${G2O_EIGEN3_INCLUDE})
FIND_PACKAGE(Eigen3  REQUIRED)
IF(EIGEN3_FOUND)
  SET(G2O_EIGEN3_INCLUDE ${EIGEN3_INCLUDE_DIR} CACHE PATH "Directory of Eigen3")
ELSE(EIGEN3_FOUND)
  SET(G2O_EIGEN3_INCLUDE "" CACHE PATH "Directory of Eigen3")
ENDIF(EIGEN3_FOUND)

# Generate config.h
SET(G2O_CXX_COMPILER "${CMAKE_CXX_COMPILER_ID} ${CMAKE_CXX_COMPILER}")
configure_file(config.h.in ${g2o_SOURCE_DIR}/config.h)

# Set up the top-level include directories
INCLUDE_DIRECTORIES(
${g2o_SOURCE_DIR}/core
${g2o_SOURCE_DIR}/types
${g2o_SOURCE_DIR}/stuff
${G2O_EIGEN3_INCLUDE})
```

Ready

Modification:

```
~/your orbslam3 source code path/Thirdparty/Sophus/CMakeLists.txt
```

navrobo_nx_CMakeLists(16).txt

```
project(Sophus VERSION 1.1.0)

include(CMakePackageConfigHelpers)
include(GNUInstallDirs)

# Release by default
# Turn on Debug with "-DCMAKE_BUILD_TYPE=Debug"
if(NOT CMAKE_BUILD_TYPE)
  set(CMAKE_BUILD_TYPE Release)
endif()

set(CMAKE_CXX_STANDARD 11)

# Set compiler specific settings (FixMe: Should not cmake do this for us automatically?)
IF("${CMAKE_CXX_COMPILER_ID}" STREQUAL "Clang")
   SET(CMAKE_CXX_FLAGS_DEBUG  "-O0 -g")
   SET(CMAKE_CXX_FLAGS_RELEASE "-O3")
   SET(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -Werror -Wextra  -Wno-deprecated-register -Qunused-arguments -fcolor-diagnostics"
)
ELSEIF("${CMAKE_CXX_COMPILER_ID}" STREQUAL "GNU")
   SET(CMAKE_CXX_FLAGS_DEBUG  "-O0 -g")
   SET(CMAKE_CXX_FLAGS_RELEASE "-O3")
   SET(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS}
 -Wall -Werror -Wextra -std=c++11 -Wno-deprecated-declarations -ftemplate-backtrace-limit=0")
   SET(CMAKE_CXX_FLAGS_COVERAGE "${CMAKE_CXX_FLAGS_DEBUG} --coverage -fno-inline -fno-inline-small-functions -fno-default-inline")
   SET(CMAKE_EXE_LINKER_FLAGS_COVERAGE "${CMAKE_EXE_LINKER_FLAGS_DEBUG} --coverage")
   SET(CMAKE_SHARED_LINKER_FLAGS_COVERAGE "${CMAKE_SHARED_LINKER_FLAGS_DEBUG} --coverage")
ELSEIF(CMAKE_CXX_COMPILER_ID MATCHES "^MSVC$")
   ADD_DEFINITIONS("-D _USE_MATH_DEFINES /bigobj /wd4305 /wd4244 /MP")
ENDIF()

# Add local path for finding packages, set the local version first
list(APPEND CMAKE_MODULE_PATH "${CMAKE_CURRENT_SOURCE_DIR}/cmake_modules")

# Find Eigen 3 (dependency)
find_package(Eigen3 REQUIRED)

# Define interface library target
add_library(sophus INTERFACE)

set(SOPHUS_HEADER_FILES
  sophus/average.hpp
  sophus/common.hpp
  sophus/geometry.hpp
  sophus/interpolate.hpp
  sophus/interpolate_details.hpp
  sophus/num_diff.hpp
```

Ready

Compile source code

```
cd your orbslam3 source code path

chmod +x build.sh
./build.sh
```

Run dataset

Download the datasets MH_01 and rgdb_dataset and unzip them into the DataSets file. Place the DataSets folder in the root directory of the orbslam3 source code.

Note: When running the shell file, it must be in the orbslam3 source code path.

1. Run the monocular euroc dataset

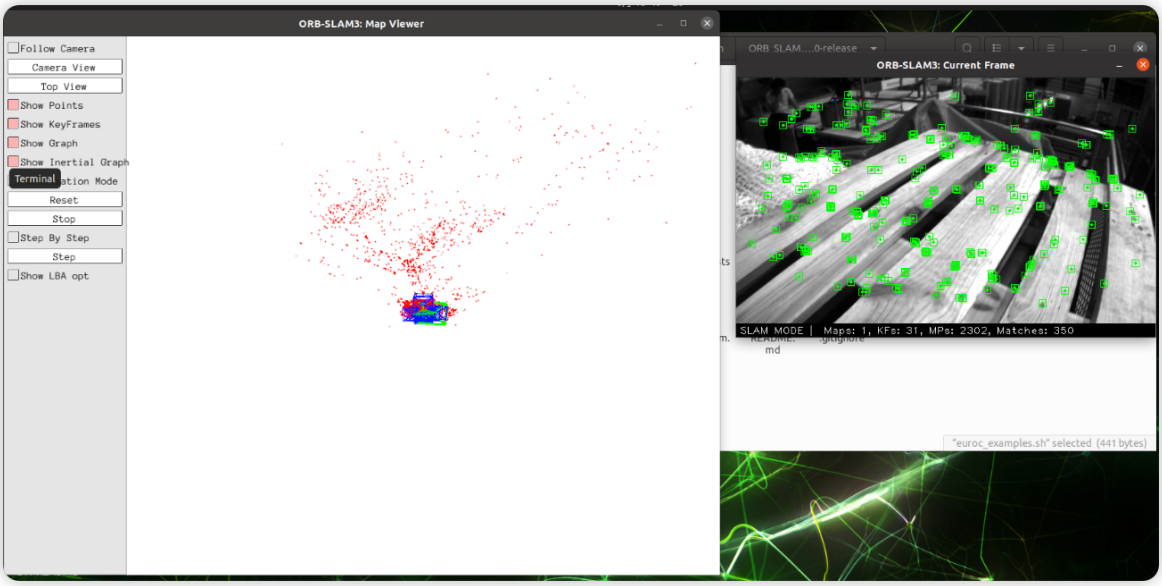
Create a file named euroc_examples.sh

Content:

```
#!/bin/bash
pathDatasetEuroc='/home/xxxxxxxx/ORB_SLAM3-1.0-release/DataSets' #Example, it is
necesary to change it by the dataset path Write your own dataset path here

#-----------------------------------
# Monocular Examples
echo "Launching MH01 with Monocular sensor"
./Examples/Monocular/mono_euroc ./Vocabulary/ORBvoc.txt
./Examples/Monocular/EuRoC.yaml "$pathDatasetEuroc"/MH_01
./Examples/Monocular/EuRoC_TimeStamps/MH01.txt dataset-MH01_mono
```

After saving, enter the command in the terminal:

```
./euroc_examples.sh
```



2. Run the stereo euroc dataset

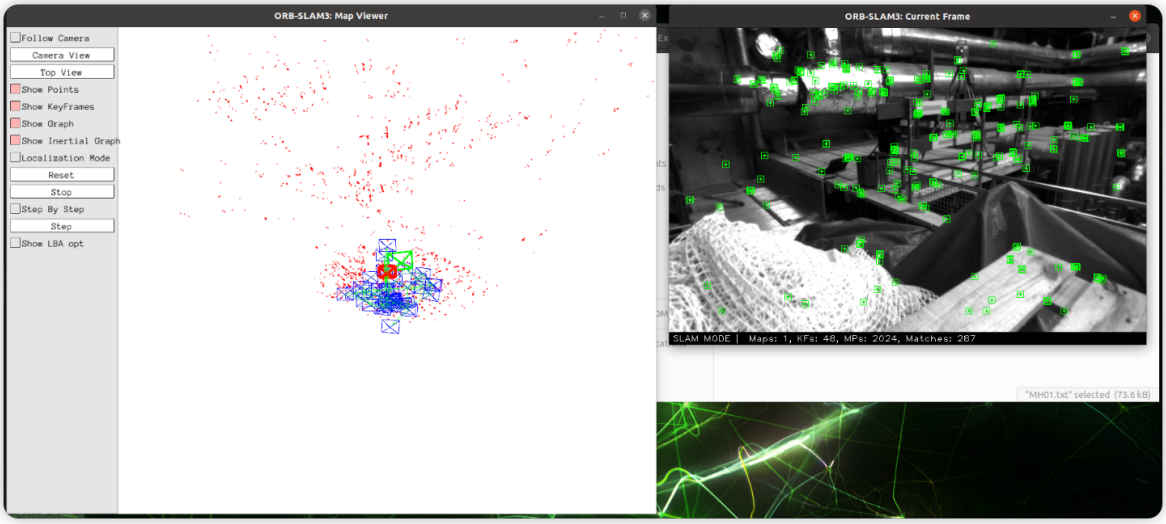
Create a file named stereo_euroc_examples.sh

The content is:

```
#!/bin/bash
pathDatasetEuroc='/home/xxxxxx/ORB_SLAM3-1.0-release/DataSets' #Example, it is
necesary to change it by the dataset path to your own dataset directory

#-----------------------------------
# Monocular Examples
echo "Launching MH01 with Monocular sensor"

./Examples/Stereo/stereo_euroc ./Vocabulary/ORBvoc.txt
./Examples/Stereo/EuRoC.yaml "$pathDatasetEuroc"/MH_01
./Examples/Stereo/EuRoC_TimeStamps/MH01.txt dataset-MH01_stere
```

After saving, enter the command in the terminal:

```
./stereo_euroc_examples.sh
```



3. Run RGBD_TUM dataset

Create a file named rgbd_tum.sh

Content:

```
#!/bin/bash
pathDatasetTum='/home/xxxxx/ORB_SLAM3-1.0-release/DataSets' #Example, it is
necesary to change it by the dataset path to your own data path

#------------------------------------
# Monocular Examples
echo "Launching MH01 with Monocular sensor"
./Examples/RGB-D/rgbd_tum Vocabulary/ORBvoc.txt Examples/RGB-D/TUM3.yaml
"$pathDatasetTum"/rgbd_dataset_freiburg3_long_office_household
"$pathDatasetTum"/rgbd_dataset_freiburg3_long_office_household/associations.txt
```

Then enter the command in the terminal:

```
./rgbd_tum.sh
```

4. Run the MONO_TUM dataset

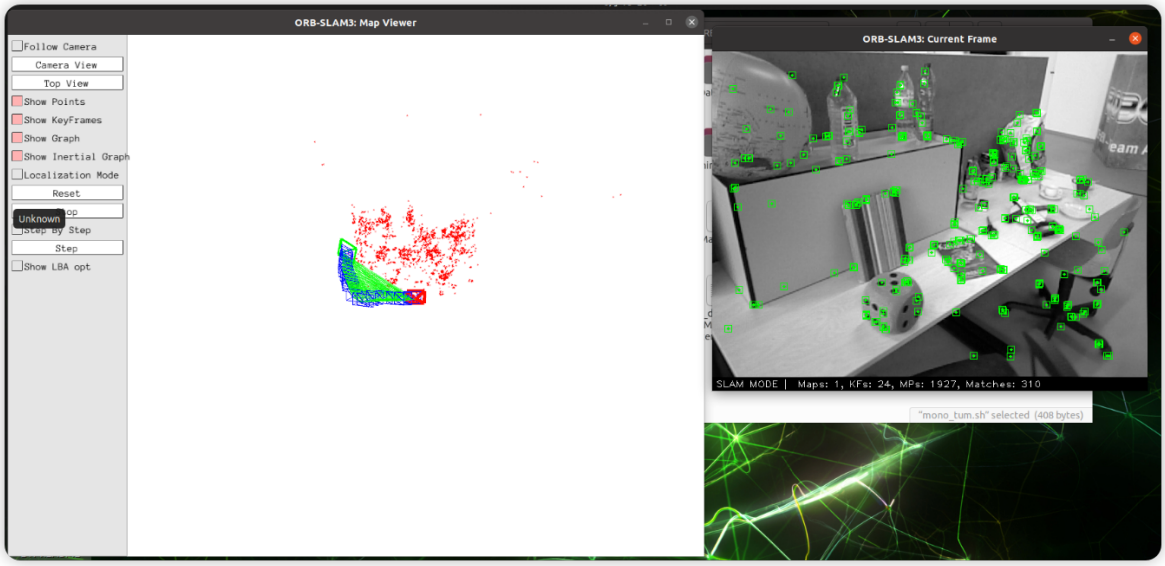
Create a file named mono_tum.sh

The content is:

```
#!/bin/bash
pathDatasetTum='/home/xxxx/ORB_SLAM3-1.0-release/DataSets' #Example, it is
necesary to change it by the dataset path to your own dataset path

#------------------------------------
# Monocular Examples
echo "Launching MH01 with Monocular sensor"

./Examples/Monocular/mono_tum ./Vocabulary/ORBvoc.txt
./Examples/Monocular/TUM3.yaml
"$pathDatasetTum"/rgbd_dataset_freiburg3_long_office_household
```

Then enter the command in the terminal:

```
./mono_tum.sh
```



Compile orbslam_ros

Download the orbslam3 source code and enter the Examples_old/ROS/ORB_SLAM3/ folder. Here is the source code of orbslam3_ros. We modify the cmakelist.txt file as shown in the following code:

```
cmake_minimum_required(VERSION 2.4.6)
include($ENV{ROS_ROOT}/core/rosbuild/rosbuild.cmake)

rosbuild_init()

IF(NOT ROS_BUILD_TYPE)
  SET(ROS_BUILD_TYPE Release)
ENDIF()

MESSAGE("Build type: " ${ROS_BUILD_TYPE})

#set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS}  -Wall  -O3 -march=native ")
#set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall  -O3 -march=native")

set(CMAKE_CXX_STANDARD 14)

set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS}  -Wall  -O3 ")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall  -O3 ")

# Check C++11 or C++0x support
include(CheckCXXCompilerFlag)
CHECK_CXX_COMPILER_FLAG("-std=c++11" COMPILER_SUPPORTS_CXX11)
CHECK_CXX_COMPILER_FLAG("-std=c++0x" COMPILER_SUPPORTS_CXX0X)
if(COMPILER_SUPPORTS_CXX11)
   set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11")
   add_definitions(-DCOMPILEDWITHC11)
   message(STATUS "Using flag -std=c++11.")
elseif(COMPILER_SUPPORTS_CXX0X)
   set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++0x")
   add_definitions(-DCOMPILEDWITHC0X)
   message(STATUS "Using flag -std=c++0x.")
```

```
else()
   message(FATAL_ERROR "The compiler ${CMAKE_CXX_COMPILER} has no C++11 support.
Please use a different C++ compiler.")
endif()

LIST(APPEND CMAKE_MODULE_PATH ${PROJECT_SOURCE_DIR}/../../../cmake_modules)



find_package(OpenCV 4.2 REQUIRED)
if(NOT OpenCV_FOUND)
   find_package(OpenCV 4.2 QUIET)
   if(NOT OpenCV_FOUND)
      message(FATAL_ERROR "OpenCV > 2.4.3 not found.")
   endif()
endif()

find_package(Eigen3 REQUIRED)
find_package(Pangolin REQUIRED)

include_directories(
${PROJECT_SOURCE_DIR}
${PROJECT_SOURCE_DIR}/../../../
${PROJECT_SOURCE_DIR}/../../../include
${PROJECT_SOURCE_DIR}/../../../include/CameraModels
${PROJECT_SOURCE_DIR}/../../../Thirdparty/Sophus
${Pangolin_INCLUDE_DIRS}
)

set(LIBS
${OpenCV_LIBS}
${EIGEN3_LIBS}
${Pangolin_LIBRARIES}
${PROJECT_SOURCE_DIR}/../../../Thirdparty/DBoW2/lib/libDBoW2.so
${PROJECT_SOURCE_DIR}/../../../Thirdparty/g2o/lib/libg2o.so
${PROJECT_SOURCE_DIR}/../../../lib/libORB_SLAM3.so

-lboost_system
)

# Node for monocular camera
rosbuild_add_executable(Mono
src/ros_mono.cc
)

target_link_libraries(Mono
${LIBS}
)

# Node for monocular camera (Augmented Reality Demo)
#rosbuild_add_executable(MonoAR
#src/AR/ros_mono_ar.cc
#src/AR/ViewerAR.h
##src/AR/ViewerAR.cc
#)
#target_link_libraries(MonoAR
#${LIBS}
#)
```

```
# Node for stereo camera
#rosbuild_add_executable(Stereo
#src/ros_stereo.cc
#)

#target_link_libraries(Stereo
#${LIBS}
#)

# Node for RGB-D camera
rosbuild_add_executable(RGBD
src/ros_rgbd.cc
)

target_link_libraries(RGBD
${LIBS}
)

# Node for monocular-inertial camera
rosbuild_add_executable(Mono_Inertial
src/ros_mono_inertial.cc
)

target_link_libraries(Mono_Inertial
${LIBS}
)

# Node for stereo-inertial camera
rosbuild_add_executable(Stereo_Inertial
src/ros_stereo_inertial.cc
)
```

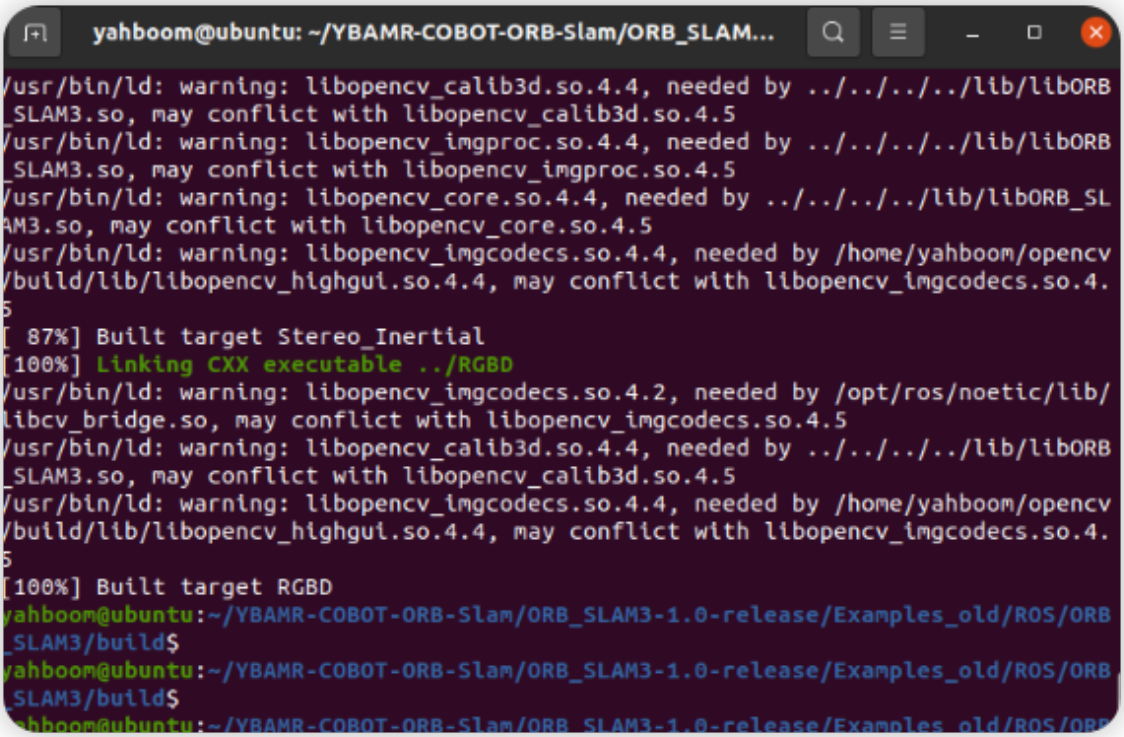
Then enter the command in the terminal:

```
mkdir build && cd build
```

```
cmake .. -DPYTHON_EXECUTABLE=/usr/bin/python3
```

```
make -j8
```

After the compilation is completed, the following figure is shown. Warnings can be ignored:



Enter the following command in two terminals:

```
#Take astra_pro2 camera as an example
roslaunch orbbec_camera astra_pro2.launch
```

```
rosrun ORB_SLAM3 RGBD /home/yahboom/YBAMR-COBOT-ORB-Slam/ORB_SLAM3-1.0-release/Vocabulary/ORBvoc.txt /home/yahboom/YBAMR-COBOT-ORB-Slam/ORB_SLAM3-1.0-release/Examples_old/ROS/ORB_SLAM3/Asus.yaml
```

The result is as follows: