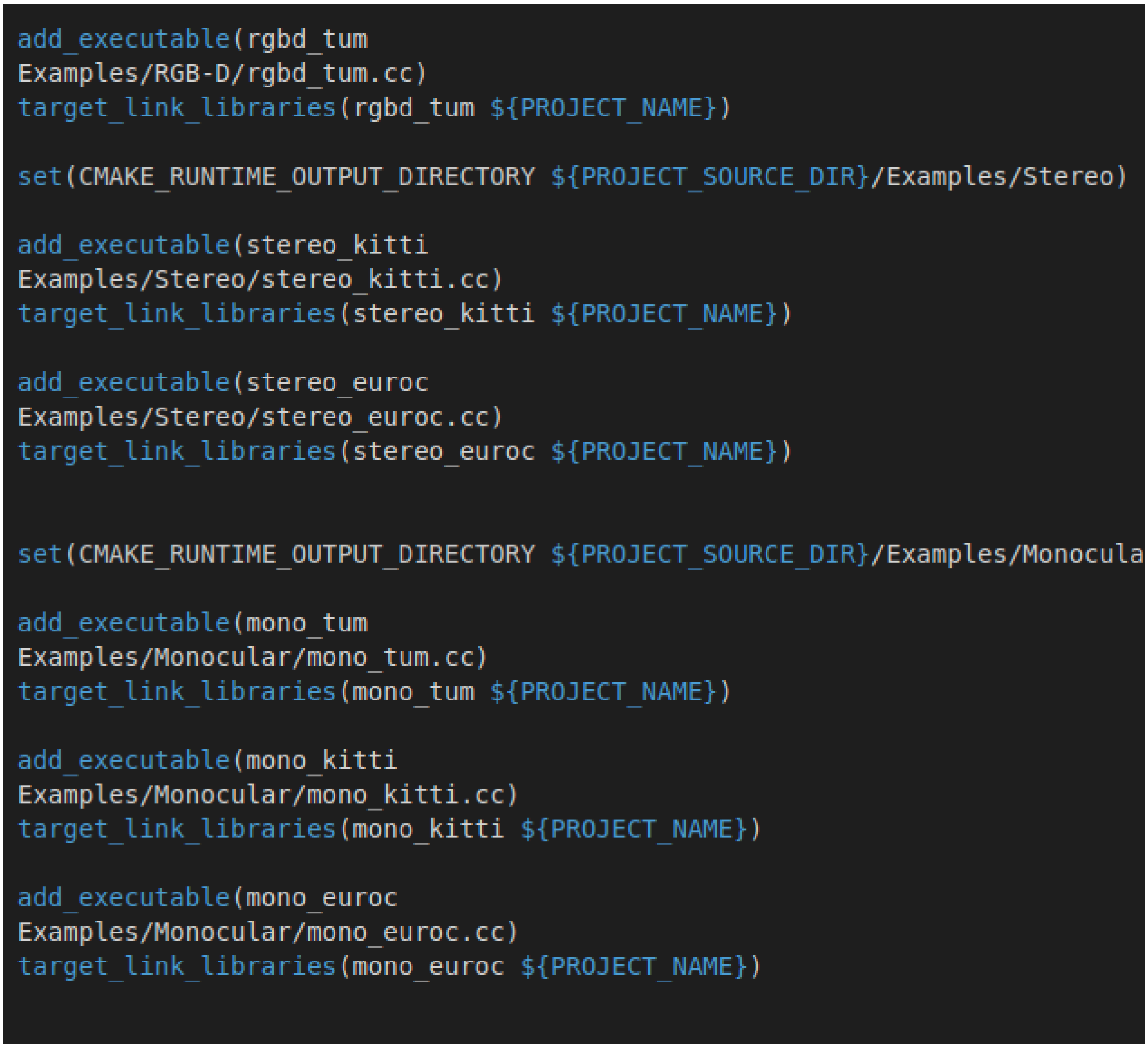
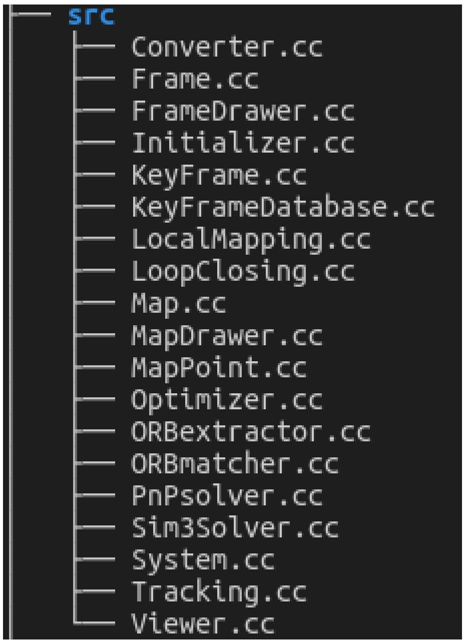


```
add_executable(rgbd_tum
Examples/RGB-D/rgbd_tum.cc)
target_link_libraries(rgbd_tum ${PROJECT_NAME})

set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${PROJECT_SOURCE_DIR}/Examples/Stereo)

add_executable(stereo_kitti
Examples/Stereo/stereo_kitti.cc)
target_link_libraries(stereo_kitti ${PROJECT_NAME})

add_executable(stereo_euroc
Examples/Stereo/stereo_euroc.cc)
target_link_libraries(stereo_euroc ${PROJECT_NAME})


set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${PROJECT_SOURCE_DIR}/Examples/Monocula

add_executable(mono_tum
Examples/Monocular/mono_tum.cc)
target_link_libraries(mono_tum ${PROJECT_NAME})

add_executable(mono_kitti
Examples/Monocular/mono_kitti.cc)
target_link_libraries(mono_kitti ${PROJECT_NAME})

add_executable(mono_euroc
Examples/Monocular/mono_euroc.cc)
target_link_libraries(mono_euroc ${PROJECT_NAME})
```

2(a).根据add_executable的信息，可以知道将生成的可执行文件:

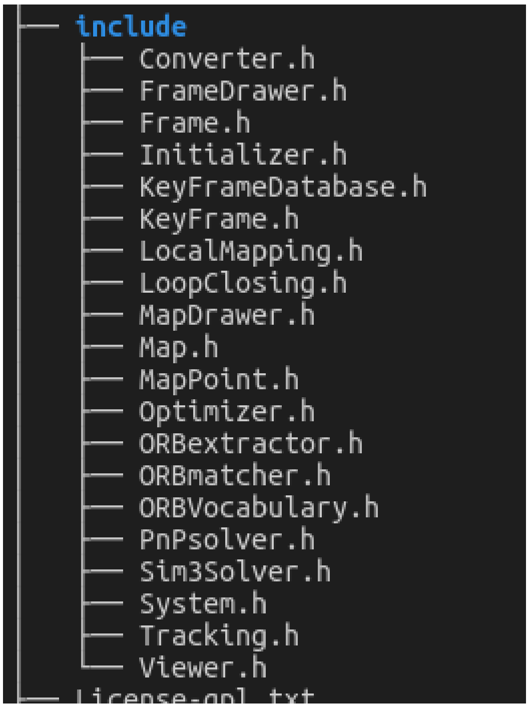
- rgbd_tum
- stereo_kitti
- stereo_euroc
- Mono_tum
- Mono_kitti
- mono_euroc

2(b)在include文件夹中，有如下文件



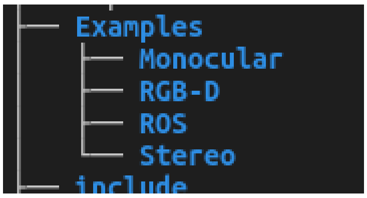
```
├── src
│   ├── Converter.cc
│   ├── Frame.cc
│   ├── FrameDrawer.cc
│   ├── Initializer.cc
│   ├── KeyFrame.cc
│   ├── KeyFrameDatabase.cc
│   ├── LocalMapping.cc
│   ├── LoopClosing.cc
│   ├── Map.cc
│   ├── MapDrawer.cc
│   ├── MapPoint.cc
│   ├── Optimizer.cc
│   ├── ORBextractor.cc
│   ├── ORBmatcher.cc
│   ├── PnPsolver.cc
│   ├── Sim3Solver.cc
│   ├── System.cc
│   ├── Tracking.cc
│   └── Viewer.cc
```

Include中有



```
├── include
│   ├── Converter.h
│   ├── FrameDrawer.h
│   ├── Frame.h
│   ├── Initializer.h
│   ├── KeyFrameDatabase.h
│   ├── KeyFrame.h
│   ├── LocalMapping.h
│   ├── LoopClosing.h
│   ├── MapDrawer.h
│   ├── Map.h
│   ├── MapPoint.h
│   ├── Optimizer.h
│   ├── ORBextractor.h
│   ├── ORBmatcher.h
│   ├── ORBVocabulary.h
│   ├── PnPsolver.h
│   ├── Sim3Solver.h
│   ├── System.h
│   ├── Tracking.h
│   └── Viewer.h
├── License-gpl.txt
```

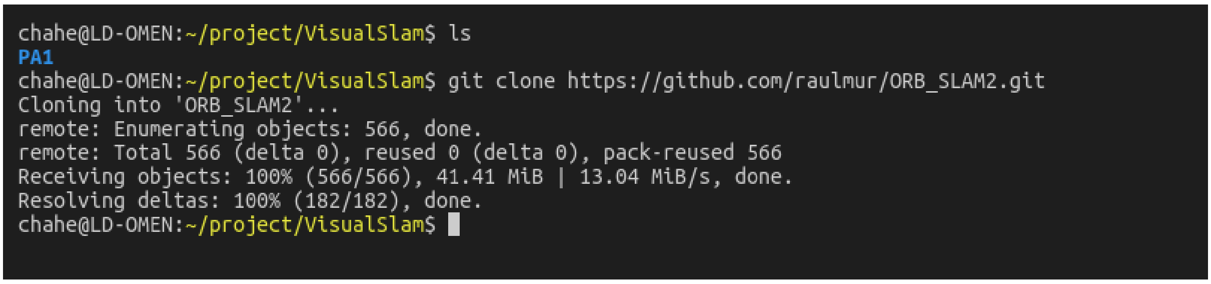
Examples中有



```
├── Examples
│   ├── Monocular
│   ├── RGB-D
│   ├── ROS
│   └── Stereo
├── include
```

2(c).ORB中的可执行文件链接到了如下库
opencv(用于图像处理)，eigen3（用于矩阵计算）,libDBoW2（用于回环检测）,Pangolin(可视化),g2o(优化问题)

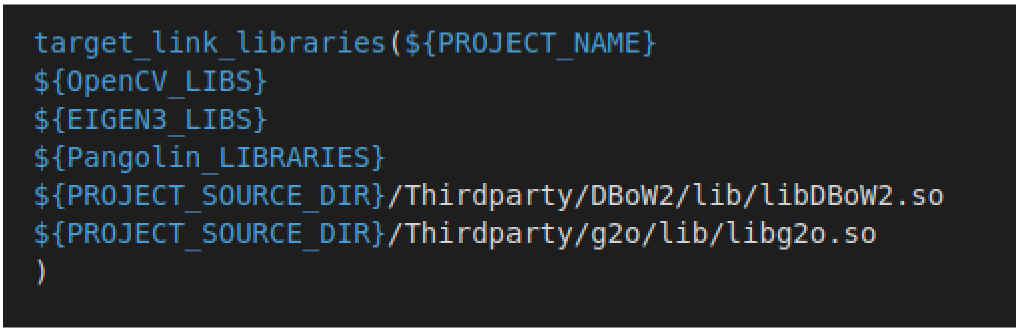
5
(1)
下载后的截图



```
chahe@LD-OMEN:~/project/VisualSlam$ ls
PA1
chahe@LD-OMEN:~/project/VisualSlam$ git clone https://github.com/raulmur/ORB_SLAM2.git
Cloning into 'ORB_SLAM2'...
remote: Enumerating objects: 566, done.
remote: Total 566 (delta 0), reused 0 (delta 0), pack-reused 566
Receiving objects: 100% (566/566), 41.41 MiB | 13.04 MiB/s, done.
Resolving deltas: 100% (182/182), done.
chahe@LD-OMEN:~/project/VisualSlam$ 
```

（2）

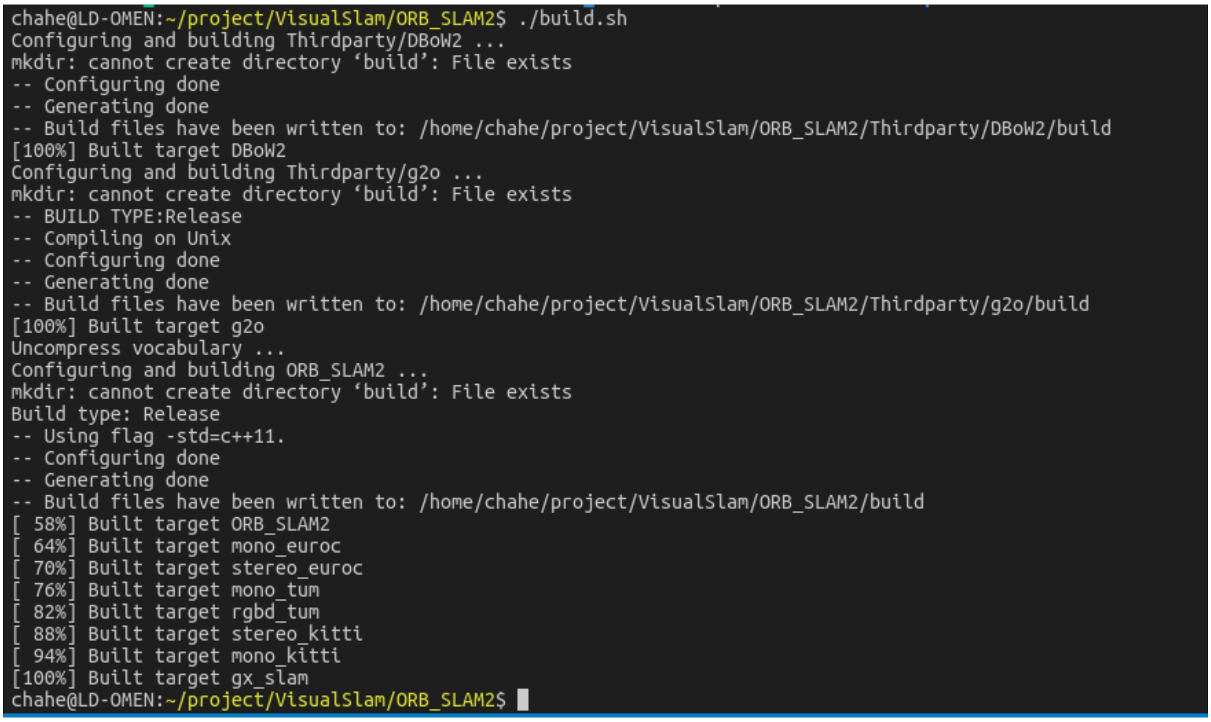
(a)编译通过后,ORB-SLAM2编译将在Examples文件夹各子目录中分别生成mono_tum, mono_kitti, mono_euroc, stereo_kitti, stereo_eruroc这些可执行文件。
此外，根据如下信息，可以知道生成的动态库文件在lib文件中，和项目同名，也就是./lib/libORB_SLAM2.so



```
target_link_libraries(${PROJECT_NAME}
${OpenCV_LIBS}
${EIGEN3_LIBS}
${Pangolin_LIBRARIES}
${PROJECT_SOURCE_DIR}/Thirdparty/DBoW2/lib/libDBoW2.so
${PROJECT_SOURCE_DIR}/Thirdparty/g2o/lib/libg2o.so
)
```

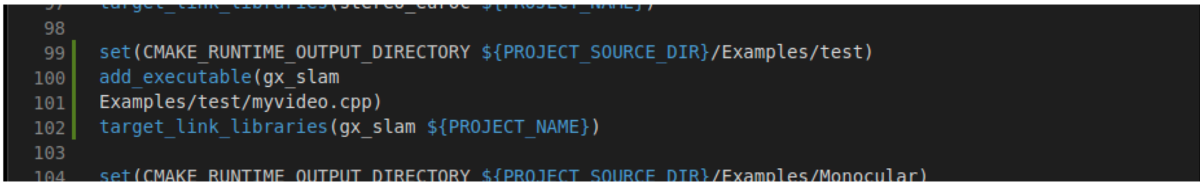
(b) include中包含头文件，src中包含.cc对应的需要被编译连接的source file ,Examples存放的是可以调用源文件执行项目的可执行文件，和相关yaml参数。
(c)可执行文件所链接的库对应的就是./lib/libORB_SLAM2.so，其所链接的就是opencv, eigen3,pangolin,libDBoW2,libg2o

6
(1). 执行build脚本直接编译



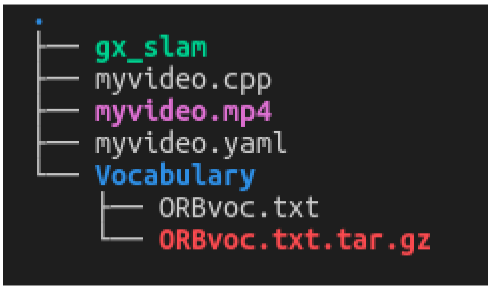
```
chahe@LD-OMEN:~/project/VisualSlam/ORB_SLAM2$ ./build.sh
Configuring and building Thirdparty/DBoW2 ...
mkdir: cannot create directory ‘build’: File exists
-- Configuring done
-- Generating done
-- Build files have been written to: /home/chahe/project/VisualSlam/ORB_SLAM2/Thirdparty/DBoW2/build
[100%] Built target DBoW2
Configuring and building Thirdparty/g2o ...
mkdir: cannot create directory ‘build’: File exists
-- BUILD TYPE:Release
-- Compiling on Unix
-- Configuring done
-- Generating done
-- Build files have been written to: /home/chahe/project/VisualSlam/ORB_SLAM2/Thirdparty/g2o/build
[100%] Built target g2o
Uncompress vocabulary ...
Configuring and building ORB_SLAM2 ...
mkdir: cannot create directory ‘build’: File exists
Build type: Release
-- Using flag -std=c++11.
-- Configuring done
-- Generating done
-- Build files have been written to: /home/chahe/project/VisualSlam/ORB_SLAM2/build
[ 58%] Built target ORB_SLAM2
[ 64%] Built target mono_euroc
[ 70%] Built target stereo_euroc
[ 76%] Built target mono_tum
[ 82%] Built target rgbd_tum
[ 88%] Built target stereo_kitti
[ 94%] Built target mono_kitti
[100%] Built target gx_slam
chahe@LD-OMEN:~/project/VisualSlam/ORB_SLAM2$ 
```

(2).如下是我的cmakelists修改方案



```
target_link_libraries(stereo_euroc ${PROJECT_NAME})

set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${PROJECT_SOURCE_DIR}/Examples/test)
add_executable(gx_slam
Examples/test/myvideo.cpp)
target_link_libraries(gx_slam ${PROJECT_NAME})

set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${PROJECT_SOURCE_DIR}/Examples/Monocular)
```

此外，我将vocabulary以及yaml等文件根据myvideo要求，进行了挪动，文件组成如下图所示。



```
.
├── gx_slam
├── myvideo.cpp
├── myvideo.mp4
├── myvideo.yaml
└── Vocabulary
    ├── ORBvoc.txt
    └── ORBvoc.txt.tar.gz
```

(3)体会，orb得到的特征相对密集，从图中效果来看，匹配效果较好，但是，得到的点云地图比较稀疏。