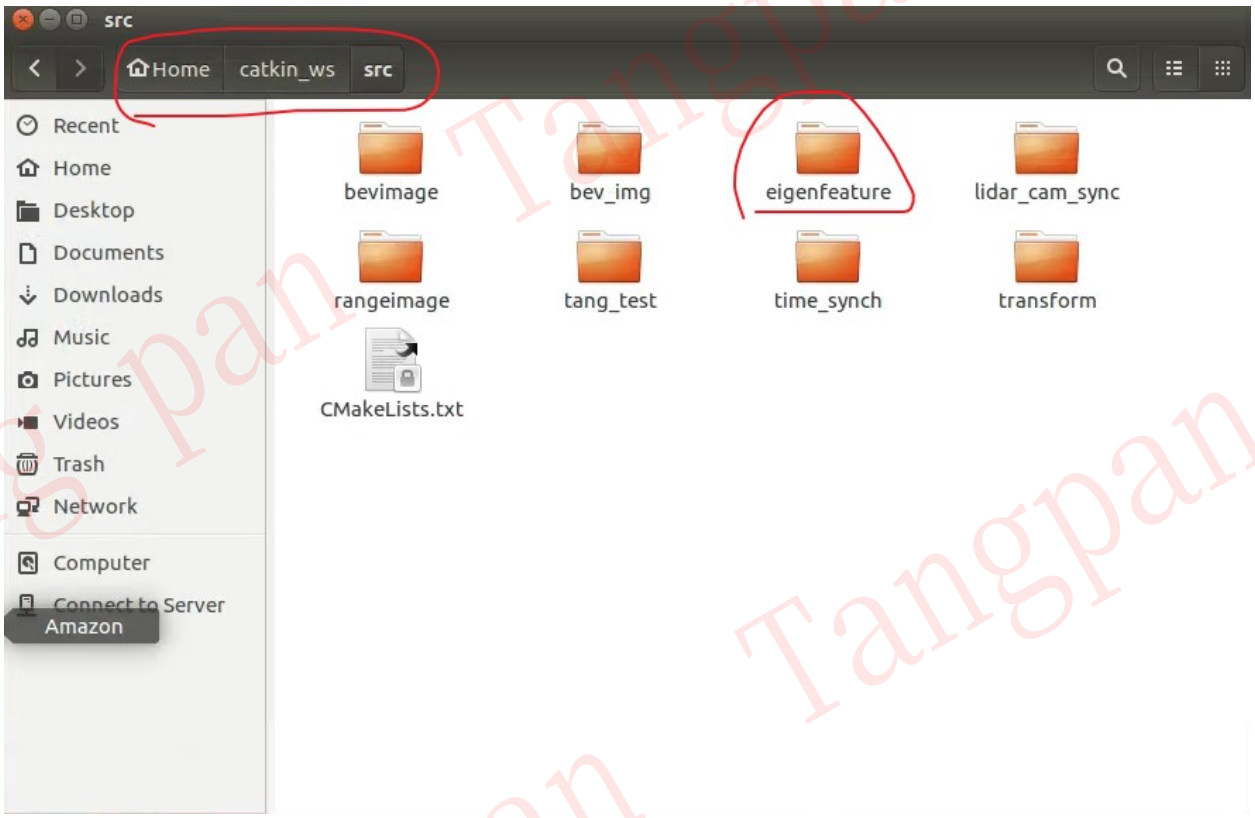
基于点云特征值的特征计算程序调用步骤：

1、把eigenfeature文件夹放到根目录catkin_ws下的src文件中，并更改各部分名字。放置位置如图所示，根据作业要求修改为自己需要的包名、topic名，节点名等，要让上述第一部分所有文件中的相关名字保持一致，作业的要求和更改如下图所示，其余部分自己按照作业要求统一修改。



提交说明:

1. 提交 ROS package 和深度图截图或 gif 或视频(<10Mb)，参数设置#1 前视视角 360 度范围，#2 后视视角 180 度范围。

2. package 命名为 aiimooc_姓名缩写，如 aiimooc_scz。

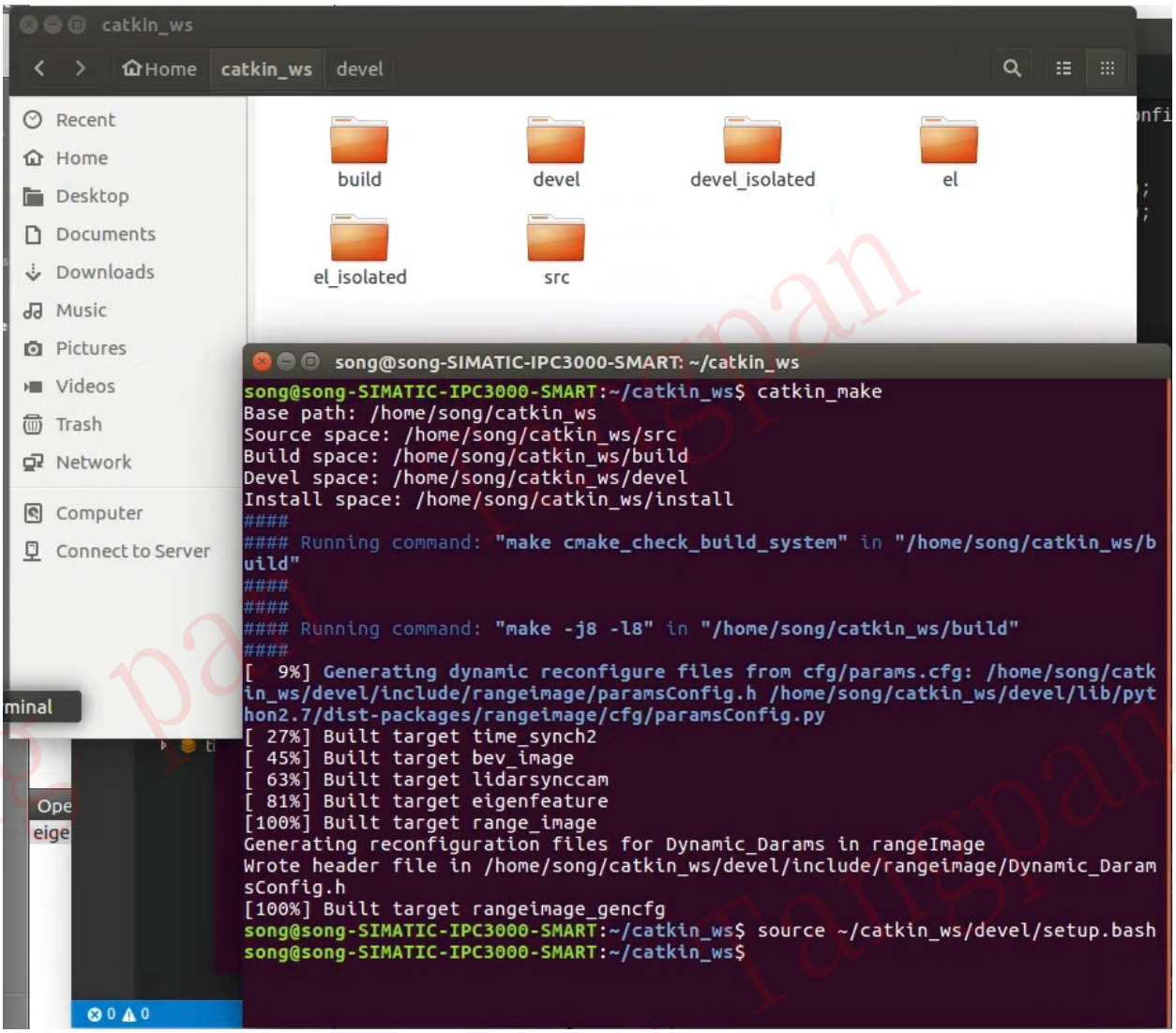
3. node 和 script 命名为 range_image，图片命名为姓名缩写_N,N=1,2,3…，如 scz_1.gif, scz_2.png。

package.xml

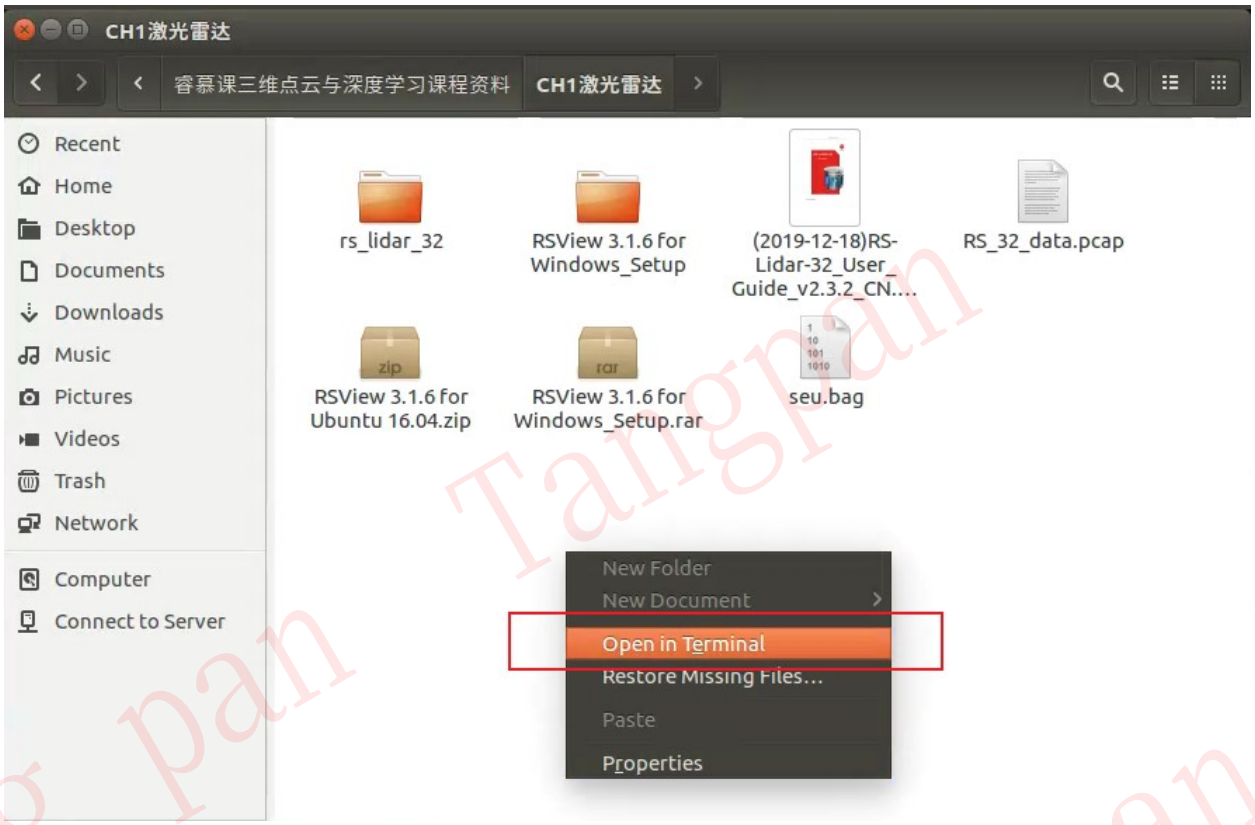
```xml
<?xml version="1.0"?>
<package>
    <name>aiimooc_tangpan</name>
    <version>0.0.0</version>
    <description>rangeimage</description>
    <maintainer email="11111111@qq.com">ccf</maintainer>
    <license>TODO</license>

    <buildtool_depend>catkin</buildtool_depend>

    <build_depend>roscpp</build_depend>
    <run_depend>roscpp</run_depend>

    <build_depend>std_msgs</build_depend>
    <run_depend>std_msgs</run_depend>

    <build_depend>sensor_msgs</build_depend>
    <run_depend>sensor_msgs</run_depend>



    <export> </export>

</package>
```

自己的包名

2、启动节点管理器，编译和更新环境配置。roscore 命令启动节点管理器，并在根目录下执行：catkin_make 命令编译源码，等到100%编译完成再执行：source ~/catkin_ws/devel/setup.bash 命令更新一下环境配置（source ~/catkin_ws/devel/setup.bash 中的 catkin_ws 是我的根目录，一般根据教程配置好环境之后都是这个根目录）。

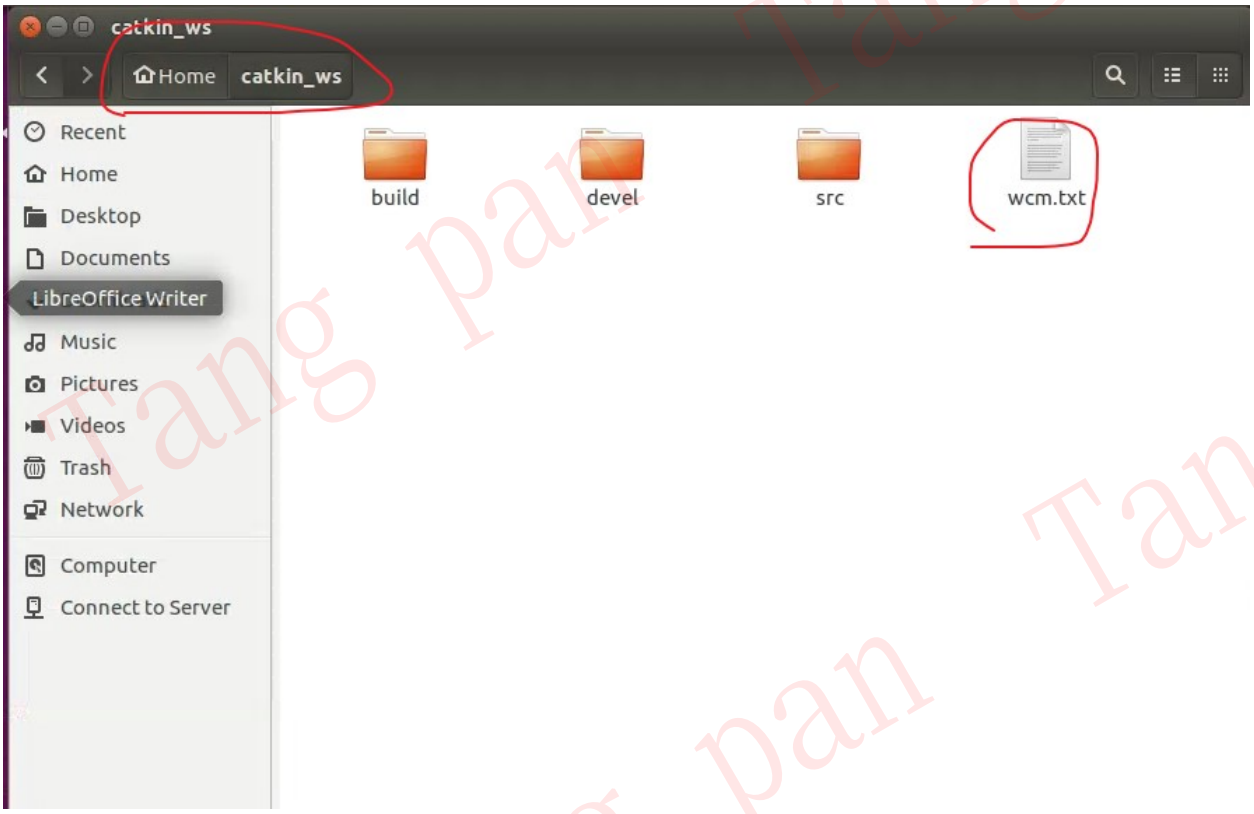


3、循环播放录制好的数据包seu.bag。 在数据包所在的文件夹内，右键选择open in Terminal命令打开终端（只有在该路径下打开终端才可以播放数据包的数据），在终端中输入：rosbag play seu.bag -l 循环播放录制好的seu.bag



4、rosrun命令启动eigenfeature.cpp转化的可执行文件。依然在工作空间根目录（我的根目录是catkin_ws）下使用rosrun命令启动可执行文件：rosrun eigenfeature eigenfeature（rosrun+包名+可执行文件名）。当显示cal done的时候就说明运行完成，存储特征值数据的txt文件已经生成，位置如图在根目录 catkin_ws 下，名字为wcm.txt，可以在 .cpp 文件中随意改想要存储的txt文件名。

```
song@song-SIMATIC-IPC3000-SMART: ~/catkin_ws
song@song-SIMATIC-IPC3000-SMART:~/catkin_ws$ catkin_make
Base path: /home/song/catkin_ws
Source space: /home/song/catkin_ws/src
Build space: /home/song/catkin_ws/build
Devel space: /home/song/catkin_ws/devel
Install space: /home/song/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/song/catkin_ws/b
uild"
####
####
#### Running command: "make -j8 -l8" in "/home/song/catkin_ws/build"
####
[  6%] Generating dynamic reconfigure files from cfg/params.cfg: /home/song/catk
in_ws/devel/include/rangeimage/paramsConfig.h /home/song/catkin_ws/devel/lib/pyt
hon2.7/dist-packages/rangeimage/cfg/paramsConfig.py
[ 20%] Built target time_synch2
[ 40%] Built target transform
[ 46%] Built target lidarsynccam
[ 60%] Built target bev_image
[ 73%] Built target eigenfeature
[ 86%] Built target bev_img
[100%] Built target range_image
Generating reconfiguration files for Dynamic_Darams in rangeImage
Wrote header file in /home/song/catkin_ws/devel/include/rangeimage/Dynamic_Daram
sConfig.h
[100%] Built target rangeimage_gencfg
song@song-SIMATIC-IPC3000-SMART:~/catkin_ws$ source ~/catkin_ws/devel/setup.bash
song@song-SIMATIC-IPC3000-SMART:~/catkin_ws$ rosrun eigenfeature eigenfeature
cal done
song@song-SIMATIC-IPC3000-SMART:~/catkin_ws$
```



eigenfeature.cpp - Qt Creator

```cpp
    std::vector<int> indices;
    pcl::KdTreeFLANN<Point> kdtree; // 用于搜索最近点


    // sensor_msgs::PointCloud2 -> pcl::PointCloud
    pcl::fromROSMsg(*msg, *pc);

    // is_dense 用于表示点云中的所有数据是否合法
    // 设置为 否，让后续的函数再检查一遍，把 Nan (not a number)
    // 这种不合法的数值全部去掉
    pc->is_dense = false;
    pcl::removeNaNFromPointCloud(*pc, *pc, indices);

    // 初始化 kd 树
    kdtree.setInputCloud(pc);

    // 提前将需要在循环中用到的变量初始化好，放置在循环中重复构造变量与析构，拖慢程序运行速度
    const int k = 20;                           // 临近点数量，根据作业要求设置为 20
    std::vector<int> point_idx(k);              // 用来保存临近点再原来点云中的下标
    std::vector<float> point_sq_dis(k);         // 用来保存临近点到目标点距离的平方
    std::vector<float> features(6);             // 用来保存六种点云特征
    std::vector<float> e(3);                    // 用来保存 k+1 个点经过 PCA 分析后得到的三个特征值计算得到的 e，从大到小排序
    std::ofstream file;                         // 输出计算结果的目标文件
    Eigen::Matrix<float, 3, 21> nearest_points; // 3x(k+1) 维的矩阵，用来保存点云中的点
    Eigen::Matrix3f covariance;                 // 用来保存协方差矩阵
    Eigen::Vector3f m, eigen_value;             // m 为 k+1 个点的质心，eigen_value 用来保存计算好的计算好的特征值

    // 打开文件，没有就凭空创建一个，如果有就删掉里面的内容，再写入新的
    // 一般不会出错
    file.open("wcm.txt");

    // pcl::PointCloud 中保存点的对象，我们用引用单独给他拿出来
    // 方便后续写代码
    auto& points = pc->points;
    for (size_t i = 0; i < pc->size(); i++)
    {
        // 每隔五个点计算一次特征值，作业没有要求这么做
        // 只是想这么做，希望能快点
        if(i%5 != 0) continue;

        // 重置 m，因为 m 需要累加，而其他的变量只需要赋值
        m = m.Zero();

        // 搜索目标点最近的几个点
        // https://pointclouds.org/documentation/classpcl_1_1_organized_neighbor_search.html#a3c18f38a4aad5fe6c05179906faf14cb
        kdtree.nearestKSearch(points[i], k, point_idx, point_sq_dis);

        // 累加搜索后的数据
        for (size_t j = 0; j < k; j++)
        {
            // 矩阵的块操作，将每个点作为列向量存入 nearest_points
            // http://eigen.tuxfamily.org/dox/group__TutorialBlockOperations.html
            nearest_points.col(j) << points[point_idx[j]].x, points[point_idx[j]].y, points[point_idx[j]].z;
            m[0] += points[point_idx[j]].x;
            m[1] += points[point_idx[j]].y;
            m[2] += points[point_idx[j]].z;
        }
        nearest_points.col(k) << points[i].x, points[i].y, points[i].z;
        m[0] += points[i].x;
        m[1] += points[i].y;
        m[2] += points[i].z;

        // 矩阵的广播操作，将每一列减去 k+1 个点的质心
        // http://eigen.tuxfamily.org/dox/group__TutorialReductionsVisitorsBroadcasting.html
```

存储特征值的文件名

CMakeLists.txt (~/catkin_ws/src/eigenfeature) - gedit

```cmake
cmake_minimum_required(VERSION 3.0.2)
project(eigenfeature)

## Compile as C++11, supported in ROS Kinetic and newer
# add_compile_options(-std=c++11)

execute_process(COMMAND "lsb_release" -rs
    OUTPUT_VARIABLE SYSTEM_RELEASE
    OUTPUT_STRIP_TRAILING_WHITESPACE
)
if(SYSTEM_RELEASE MATCHES "20.04")
  message("Using c++14")
  add_compile_options(-std=c++14)
else()
  message("Using c++11")
  add_compile_options(-std=c++11)
endif()

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  pcl_conversions
  pcl_ros
  roscpp
  sensor_msgs
)
find_package(PCL REQUIRED)

## System dependencies are found with CMake's conventions
# find_package(Boost REQUIRED COMPONENTS system)


## Uncomment this if the package has a setup.py. This macro ensures
## modules and global scripts declared therein get installed
```

包名

CMakeLists.txt (~/catkin_ws/src/eigenfeature) - gedit

Open Save

```
# add_library(${PROJECT_NAME}
#   src/${PROJECT_NAME}/aiimooc_wcm.cpp
# )

## Add cmake target dependencies of the library
## as an example, code may need to be generated before libraries
## either from message generation or dynamic reconfigure
# add_dependencies(${PROJECT_NAME} ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})

## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
add_executable(eigenfeature src/eigenfeature.cpp)

## Rename C++ executable without prefix
## The above recommended prefix causes long target names, the following renames the
## target back to the shorter version for ease of user use
## e.g. "rosrun someones_pkg node" instead of "rosrun someones_pkg someones_pkg_node"
# set_target_properties(${PROJECT_NAME}_node PROPERTIES OUTPUT_NAME node PREFIX "")

## Add cmake target dependencies of the executable
## same as for the library above
# add_dependencies(${PROJECT_NAME}_node CalFeatureLib ${catkin_EXPORTED_TARGETS})

## Specify libraries to link a library or executable target against
target_link_libraries(eigenfeature
  ${catkin_LIBRARIES}
  ${PCL_LIBRARIES}
)

#############
## Install ##
#############

# all install targets should use catkin DESTINATION variables
```

实现特征值提取的cpp文件名

可执行文件名

CMake Tab Width: 8 Ln 145, Col 31 INS

注意事项：

CMakeLists.txt (~/Desktop/SLAMResearch/CH2_PointCloudFeature/eigenfeature) - gedit

Open Save

```
cmake_minimum_required(VERSION 3.0.2)
project(eigenfeature LANGUAGES CXX)

## Compile as C++11, supported in ROS Kinetic and newer
# add_compile_options(-std=c++11)

execute_process(COMMAND "lsb_release" -rs
    OUTPUT_VARIABLE SYSTEM_RELEASE
    OUTPUT_STRIP_TRAILING_WHITESPACE
)
if(SYSTEM_RELEASE MATCHES "20.04")
  message("Using c++14")
  add_compile_options(-std=c++14)
else()
  message("Using c++11")
  add_compile_options(-std=c++11)
endif()

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  pcl_conversions
  pcl_ros
  roscpp
  sensor_msgs
)
find_package(PCL REQUIRED)

## System dependencies are found with CMake's conventions
# find_package(Boost REQUIRED COMPONENTS system)


## Uncomment this if the package has a setup.py. This macro ensures
## modules and global scripts declared therein get installed
## See http://ros.org/doc/api/catkin/html/user_guide/setup_dot_py.html
```

此处LANGUAGES CXX用来标识语言，如果没装cuda可能编译报错，可以把它删掉和package.xml中包名保持一致，然后重新编译试一下。

CMake Tab Width: 8 Ln 1, Col 1 INS