最近用ROS写了简单的目标检测的小程序，使用的yolov5,在实现过程中遇到了许多坑，在这里记录一下。

系统要求

ROS=noetic

opencv=4.5.5

opencv_contrib与上述opencv版本保持一致

python库yolov5

cv_bridge

可选

n卡

cuda

cudnn

这个都是根据自己系统要求来

安装相关库

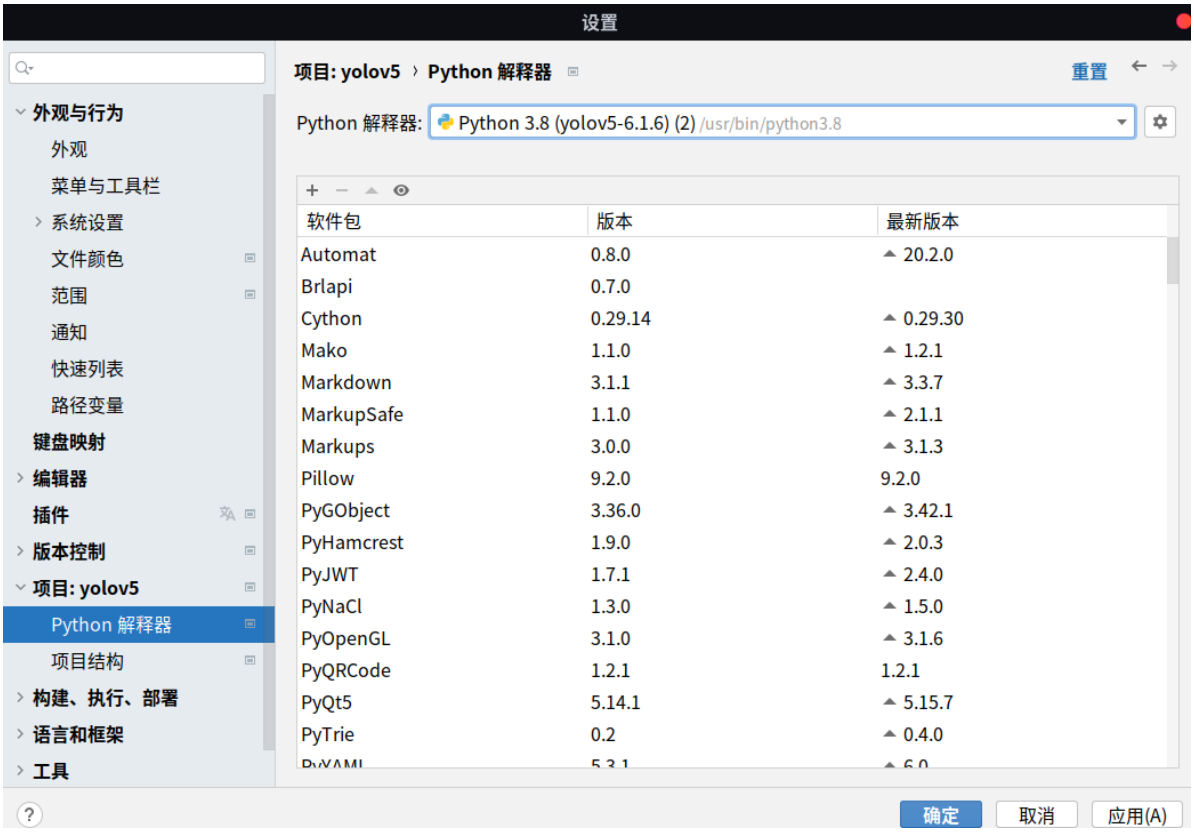
ROS

这里就不说了，在ubuntu上安装对应版本就行

opencv4.5.5+opencv_contrib4.5.5

python端

像python端的就很简单，在pycharm中新建一个项目，编译器选择系统自带的/usr/bin/python3.8



然后我们点击加号，搜索opencv-python，下载对应版本，下载时可以选择安装到系统目录。

opencv-contrib-python也是如此

C++端

这个是需要编译的，就比较复杂，尤其是自己安装过许多版本的，简直是天坑，这里来介绍一下

首先下载好opencv4.5.5和opencv_contrib4.5.5，解压到同一个目录下即可，打开opencv4.5.5，打开终端

```
mkdir build
cd build
```

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D INSTALL_C_EXAMPLES=OFF \
    -D OPENCV_ENABLE_NONFREE=ON \
    -D WITH_CUDA=ON \
    -D WITH_CUDNN=ON \
    -D OPENCV_DNN_CUDA=ON \
    -D ENABLE_FAST_MATH=1 \
    -D CUDA_FAST_MATH=1 \
    -D CUDA_ARCH_BIN=7.5 \
    -D WITH_CUBLAS=1 \
    -D OPENCV_EXTRA_MODULES_PATH=~/下载/opencv_contrib-4.5.5/modules \
    -D BUILD_EXAMPLES=ON ..
```

cmake解释

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \********编译类型release
    -D CMAKE_INSTALL_PREFIX=/usr/local \*******安装目录，就是sudo make install安装的目录，如果电脑里面有其他opencv版本的可以改为/usr/local/opencv455
    -D INSTALL_PYTHON_EXAMPLES=ON \*****python的例子，不用在意
    -D INSTALL_C_EXAMPLES=OFF \*****C的例子，不用在意
    -D OPENCV_ENABLE_NONFREE=ON \不用在意
    -D WITH_CUDA=ON \*******GPU加速，GPU的话需要先安装cuda和cudnn，可以先看下面的cuda和cudnn安装在来编译
    -D WITH_CUDNN=ON \*******同上
    -D OPENCV_DNN_CUDA=ON \*******同上
    -D ENABLE_FAST_MATH=1 \******不重要
    -D CUDA_FAST_MATH=1 \*******不重要
    -D CUDA_ARCH_BIN=7.5 \*******这个是指GPU的能力
    -D WITH_CUBLAS=1 \
    -D OPENCV_EXTRA_MODULES_PATH=~/下载/opencv_contrib-4.5.5/modules \********这个根据自己解压目录来
    -D BUILD_EXAMPLES=ON ..
```

如果是安装CPU版本的

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \********编译类型release
    -D CMAKE_INSTALL_PREFIX=/usr/local \*******安装目录，就是sudo make install安装的目录，如果电脑里面有其他opencv版本的可以改为/usr/local/opencv455
```

其他的让系统默认就好了

python-yolov5

这个就简单，直接pip3 install yolov5就好。可以看一下官网

https://pypi.org/project/yolov5/

cv_bridge

cv_bridge是一个将ROS图片信息和opencv信息转换的一个库，是ROS自带的，但是它不支持opencv4.5.5,所以需要重新编译安装。

首先先卸载掉原有的cv_bridge

```
sudo apt-get remove ros-indigo-cv-bridge//***********indigo改为自己的ROS发行版
```

然后去Github的网站https://github.com/ros-perception/vision_opencv/tags，下载vision_opencv包，里面有cv_bridge，我下载的是1.14的版本，3.几的和2.多的应该是为ROS2准备的，编译通不过

下载好之后解压，进入文件，打开cv_bridge，修改cmakelist文件，主要是将里面的opencv版本改一下

```
else()
find_package(Boost REQUIRED)
endif()

set(OpenCV_DIR /usr/local/opencv455/lib/cmake/opencv4)
find_package(OpenCV 4.5.5 REQUIRED
  COMPONENTS
    opencv_core
    opencv_imgproc
    opencv_imgcodecs
  CONFIG
)

catkin_package(
  INCLUDE_DIRS include
  LIBRARIES ${PROJECT_NAME}
  CATKIN_DEPENDS rosconsole sensor_msgs
  DEPENDS OpenCV
  CFG_EXTRAS cv_bridge-extras.cmake
)
```

打开终端

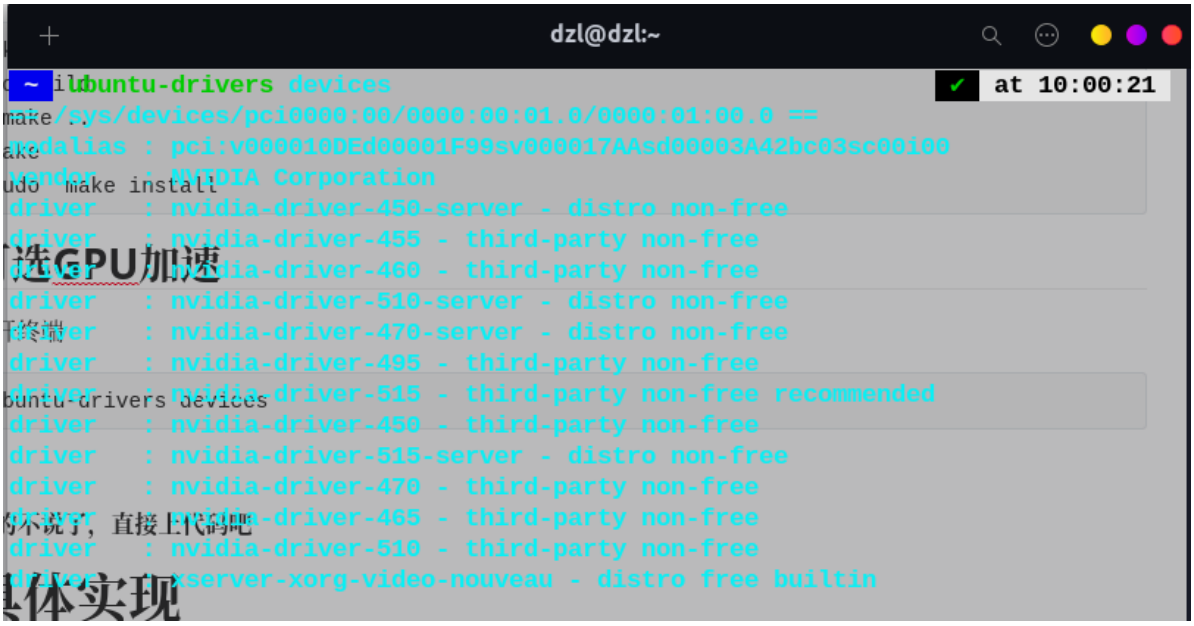
```
mkdir build
cd build
cmake ..
make
sudo  make install
```

可选GPU加速

nvidia-driver

打开终端

```
ubuntu-drivers devices
```




```
sudo apt install nvidia-driver-515
```

可以看到推荐的驱动，我的是515

安装后重启电脑

重启后输入

```
nvidia-smi
```

查看驱动信息

cuda

这里我是参考的这个网站https://medium.com/geekculture/installing-cudnn-and-cuda-toolkit-on-ubuntu-20-04-for-machine-learning-tasks-f41985fcf9b2，还有这个网站https://zhuanlan.zhihu.com/p/72298520

这个还是比较简单的，重点在下面

cudnn

首先是下载相关版本，解压安装，重点在于后面的复制移动

```
sudo cp cuda/include/cudnn.h /usr/local/cuda-10.1/include
sudo cp cuda/lib64/libcudnn* /usr/local/cuda-10.1/lib64
sudo chmod a+r /usr/local/cuda-10.1/include/cudnn.h
sudo chmod a+r /usr/local/cuda-10.1/lib64/libcudnn*
```

对于旧版本的是这样的，但是对于较新的版本

```
sudo cp cuda/include/cudnn* /usr/local/cuda-10.1/include  /************将所有的cudnn的文件全部复制过去,
sudo cp cuda/lib64/libcudnn* /usr/local/cuda-10.1/lib64
sudo chmod a+r /usr/local/cuda-10.1/include/cudnn*
sudo chmod a+r /usr/local/cuda-10.1/lib64/libcudnn*
```

不然编译opencv的GPU时候会通不过

其他的不说了，直接上代码吧

具体实现

发送端

发送端主要是将图像信息转为ROS的图像信息，然后发送出去（这里是视频），这一段比较简单是使用C++来写的

```
//
// Created by dzl on 22-6-23.
//
// http://wiki.ros.org/image_transport

#include <ros/ros.h>
#include <image_transport/image_transport.h>
#include <opencv2/highgui/highgui.hpp>
#include <cv_bridge/cv_bridge.h>

#include <stdio.h>
```

```
int main(int argc, char** argv)
{
    ros::init(argc, argv, "ros_opencv");
    // 声明节点
    ros::NodeHandle nh;
    // image_transport image订阅和发布
    // image_transport ("raw") - The default transport, sending sensor_msgs/Image
through ROS
    // 用上面声明的节点句柄初始化it，it和nh的功能基本一样使用it来发布和订阅相消息
    image_transport::ImageTransport it(nh);
    // 第一个参数是话题的名称，第二个是缓冲区的大小（消息队列的长度发布图像消息时消息队列的长度
只能是1）
    image_transport::Publisher pub = it.advertise("image", 1);


    cv::VideoCapture cap("/home/dzl/CLionProjects/-SLAM/ros-
learn/src/learn/200862413-1-64.flv");
    if (!cap.isOpened())
    {
        std::cerr << "Read video Failed !" << std::endl;
        return 0;
    }

    cv::Mat image;
    while(ros::ok() && cap.isOpened()){
        ROS_INFO("system is fine");
        cap.read(image);
        sensor_msgs::ImagePtr msg = cv_bridge::CvImage(std_msgs::Header(),
"bgr8", image).toImageMsg();
        msg->header.stamp = ros::Time::now();
        ros::Rate loop_rate(5);
        pub.publish(msg);
        ros::spinOnce();
        // 通过睡眠度过一个循环中剩下的时间
        loop_rate.sleep();
    }
    cap.release();
}
```

接收端

接收端就是接收图像信息，转为OPENCV格式，在进行识别并展示

PYTHON

```
#!/usr/bin/env python
# license removed for brevity
import rospy
from std_msgs.msg import String
import cv2
import numpy as np
from cv_bridge import CvBridge
from sensor_msgs.msg import Image
import yolov5
```

```
classesFile = "/home/dzl/CLionProjects/-SLAM/ros-learn/src/learn/yolo-
file/coco.names"
classNames = []
with open(classesFile, 'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')
model = yolov5.load('/home/dzl/CLionProjects/-SLAM/ros-learn/src/learn/yolo-
file/yolov5n6.pt')# yolov5n6.pt这个文件是在github上下载的
#
# # or load custom model
# model = yolov5.load('train/best.pt')

# set model parameters
model.conf = 0.25  # NMS confidence threshold
model.iou = 0.45  # NMS IoU threshold
model.agnostic = False  # NMS class-agnostic
model.multi_label = False  # NMS multiple labels per box
model.max_det = 1000  # maximum number of detections per image

def detector(img):
    classIds = []
    results = model(img, size = 1280 ,augment=True)
    predictions = results.pred[0]
    boxes = predictions[:, :4] # x1, y1, x2, y2
    scores = predictions[:, 4]
    categories = predictions[:, 5]
    length = boxes.shape[0]
    # print(scores)
    for i in range(length):
        x1 = int(boxes[i][0])
        y1 = int(boxes[i][1])
        x2 = int(boxes[i][2])
        y2 = int(boxes[i][3])
        cv2.rectangle(img, (x1, y1), (x2,y2), (255, 0 , 255), 2)
        classIds.append(int(categories[i]))
        # print(classNames[classIds[i]])
        cv2.putText(img,f'{classNames[classIds[i]]} {int(scores[i]*100)}%',
                    (x1, y1-10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0, 255), 2)

def callback(data):#回调函数，先转图片格式，然后检测，在然后展示
    bridge = CvBridge()
    img = bridge.imgmsg_to_cv2(data, desired_encoding="passthrough")
    detector(img)
    cv2.imshow('images',img)
    cv2.waitKey(1)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        cv2.destroyWindow('image')

def listener():
    rospy.init_node('yolo_listener', anonymous=True)# 初始化节点，订阅消息，显示成功,
不停循环
    rospy.Subscriber("image", Image, callback)
    rospy.loginfo("open succeed")
    rospy.spin()


if __name__ == '__main__':# 这里就是一个异常处理的写法
    try:
        listener()
```

```
    except rospy.ROSInterruptException:
        pass
```

C++

```
//
// Created by dzl on 22-6-24.
//
#include <ros/ros.h>
#include <image_transport/image_transport.h>
#include <opencv2/highgui/highgui.hpp>
#include <cv_bridge/cv_bridge.h>
#include <opencv2/opencv.hpp>
#include <fstream>

// Namespaces.
using namespace cv;
using namespace std;
using namespace cv::dnn;

// Constants.
const float INPUT_WIDTH = 640.0;
const float INPUT_HEIGHT = 640.0;
const float SCORE_THRESHOLD = 0.5;
const float NMS_THRESHOLD = 0.45;
const float CONFIDENCE_THRESHOLD = 0.45;

// Text parameters.
const float FONT_SCALE = 0.7;
const int FONT_FACE = FONT_HERSHEY_SIMPLEX;
const int THICKNESS = 1;

// Colors.
Scalar BLACK = Scalar(0,0,0);
Scalar BLUE = Scalar(255, 178, 50);
Scalar YELLOW = Scalar(0, 255, 255);
Scalar RED = Scalar(0,0,255);


// Draw the predicted bounding box.
void draw_label(Mat& input_image, string label, int left, int top)
{
    // Display the label at the top of the bounding box.
    int baseLine;
    Size label_size = getTextSize(label, FONT_FACE, FONT_SCALE, THICKNESS,
&baseLine);
    top = max(top, label_size.height);
    // Top left corner.
    Point tlc = Point(left, top);
    // Bottom right corner.
    Point brc = Point(left + label_size.width, top + label_size.height +
baseLine);
    // Draw black rectangle.
    rectangle(input_image, tlc, brc, BLACK, FILLED);
```

```
    // Put the label on the black rectangle.
    putText(input_image, label, Point(left, top + label_size.height), FONT_FACE,
FONT_SCALE, YELLOW, THICKNESS);
}


vector<Mat> pre_process(Mat &input_image, Net &net)
{
    // Convert to blob.
    Mat blob;
    blobFromImage(input_image, blob, 1./255., Size(INPUT_WIDTH, INPUT_HEIGHT),
Scalar(), true, false);

    net.setInput(blob);

    // Forward propagate.
    vector<Mat> outputs;
    net.forward(outputs, net.getUnconnectedOutLayersNames());

    return outputs;
}


Mat post_process(Mat &input_image, vector<Mat> &outputs, const vector<string>
&class_name)
{
    // Initialize vectors to hold respective outputs while unwrapping detections.
    vector<int> class_ids;
    vector<float> confidences;
    vector<Rect> boxes;

    // Resizing factor.
    float x_factor = input_image.cols / INPUT_WIDTH;
    float y_factor = input_image.rows / INPUT_HEIGHT;

    float *data = (float *)outputs[0].data;

    const int dimensions = 85;
    const int rows = 25200;
    // Iterate through 25200 detections.
    for (int i = 0; i < rows; ++i)
    {
        float confidence = data[4];
        // Discard bad detections and continue.
        if (confidence >= CONFIDENCE_THRESHOLD)
        {
            float * classes_scores = data + 5;
            // Create a 1x85 Mat and store class scores of 80 classes.
            Mat scores(1, class_name.size(), CV_32FC1, classes_scores);
            // Perform minMaxLoc and acquire index of best class score.
            Point class_id;
            double max_class_score;
            minMaxLoc(scores, 0, &max_class_score, 0, &class_id);
            // Continue if the class score is above the threshold.
            if (max_class_score > SCORE_THRESHOLD)
            {
                // Store class ID and confidence in the pre-defined respective
vectors.
```

```
                confidences.push_back(confidence);
                class_ids.push_back(class_id.x);

                // Center.
                float cx = data[0];
                float cy = data[1];
                // Box dimension.
                float w = data[2];
                float h = data[3];
                // Bounding box coordinates.
                int left = int((cx - 0.5 * w) * x_factor);
                int top = int((cy - 0.5 * h) * y_factor);
                int width = int(w * x_factor);
                int height = int(h * y_factor);
                // Store good detections in the boxes vector.
                boxes.push_back(Rect(left, top, width, height));
            }

        }
        // Jump to the next column.
        data += 85;
    }

    // Perform Non Maximum Suppression and draw predictions.
    vector<int> indices;
    NMSBoxes(boxes, confidences, SCORE_THRESHOLD, NMS_THRESHOLD, indices);
    for (int i = 0; i < indices.size(); i++)
    {
        int idx = indices[i];
        Rect box = boxes[idx];

        int left = box.x;
        int top = box.y;
        int width = box.width;
        int height = box.height;
        // Draw bounding box.
        rectangle(input_image, Point(left, top), Point(left + width, top +
height), BLUE, 3*THICKNESS);

        // Get the label for the class name and its confidence.
        string label = format("%.2f", confidences[idx]);
        label = class_name[class_ids[idx]] + ":" + label;
        // Draw class labels.
        draw_label(input_image, label, left, top);
    }
    return input_image;
}



void imageCallback(const sensor_msgs::ImageConstPtr& msg)
{
        // 这段代码用于显示捕捉到的图像
        // 其中cv_bridge::toCvShare(msg, "bgr8")->image
        // 用于将ROS图像消息转化为Opencv支持的图像格式采用bgr8编码方式
        // 和发布节点CvImage(std_msgs::Header(), "bgr8", image).toImageMsg()作用相反
```

```
 //******************************classnames*****************************//
        vector<string> class_list;
        ifstream ifs("/home/dzl/CLionProjects/yolo-test/coco.names");
        string line;
        while (getline(ifs, line))
        {
            class_list.push_back(line);
        }
        //***************************得到图片
***********************************//
        cv::Mat frame = cv_bridge::toCvCopy(msg,"bgr8")->image;
        //*************************load model**********************//
        Net net;
        net = readNet("/home/dzl/CLionProjects/yolo-test/models/yolov5m.onnx");
    //yolov5m.onnx这个不是作者的，作者提供的有问题，可以自己生成，不过需要改一下代码
        net.setPreferableBackend(cv::dnn::DNN_BACKEND_CUDA);//这里就是使用GPU了
        net.setPreferableTarget(cv::dnn::DNN_TARGET_CUDA_FP16);
        //*************************load result*********************//
        vector<Mat> detections;
        detections = pre_process(frame, net);
        Mat img = post_process(frame, detections, class_list);
        vector<double> layersTimes;
        double freq = getTickFrequency() / 1000;
        double t = net.getPerfProfile(layersTimes) / freq;
        string label = format("Inference time : %.2f ms", t);
        putText(img, label, Point(20, 40), FONT_FACE, FONT_SCALE, RED);
    ROS_INFO("open successfully");
        imshow("Output", img);
        waitKey(1);
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "ros_opencv_listener");//这里就是初始化和订阅消息
    ros::NodeHandle nh;
    cv::startWindowThread();
    image_transport::ImageTransport it(nh);
    image_transport::Subscriber sub = it.subscribe("image", 1, imageCallback);
    ros::spin();
    if (cv::waitKey(1)  == 27)
        cv::destroyWindow("Output");

}
```

遇到的坑

遇到的坑大多都在C++这里，一开始我是安装的opencv3.4，然后读取onnx文件时总是出错，就安装了opencv4.2，后来在网上查到yolov4至少是4.4，yolov5至少4.5，我先安装了opencv4.5.3，但是没有开GPU，并且还是读不了onnx，

谷歌一下发现yolov5的onnx是有问题的，可以看一下这个博客https://blog.csdn.net/qq_34124780/article/details/114666312，写的很清楚，

我是在https://colab.research.google.com/github/spmallick/learnopencv/blob/master/Object-Detection-using-YOLOv5-and-OpenCV-DNN-in-CPP-and-Python/Convert_PyTorch_models.ipynb网站上进行修改生成onnx文件的，

生成之后便可以读取了，但是此时还是用不了GPU，因为之前的opencv4.5.3删掉了，所以也没法卸载，不得已用了4.5.5，在编译的时候就找不到cudnn，很烦，解决方法看前面，安装的时候一开始是按照的默认的安装，然后ROS运行的时候就冲突了，不得已，只能是自己定一个文件，cmakelist文件相关部分如下所示

```
set(OpenCV_DIR /usr/local/opencv455/lib/cmake/opencv4)
find_package(OpenCV 4.5.5 REQUIRED)
set(cv_bridge_DIR /usr/local/share/cv_bridge/cmake)
find_package(catkin REQUIRED COMPONENTS
        image_transport
        roscpp
        cv_bridge
        rospy
        sensor_msgs
        std_msgs
        )
```

然后ROS编译，但还是不通过，原因是自己规定的文件没有在系统变量里面，于是又该了bash.rc，我的是zsh.rc

```
export PATH=/usr/local/cuda-11.7/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.7/lib64:$LD_LIBRARY_PATH
export CUDA_HOME=/usr/local/cuda
export LD_LIBRARY_PATH=/usr/local/opencv455/lib:$LD_LIBARARY_PATH
export LD_LIBRARY_PATH=/opt/ros/noetic/lib:$LD_LIBARARY_PATH
```

终于，成功了