

1. 实验名称及目的

1.1 实验名称

UDP直传方式发布相机以及云台数据仿真实验

1.2 实验目的

在Ubuntu下运行服务端，发送取图请求并通过UDP直传方式传输图像数据，再对回传的图像进行数据的处理，并通过订阅截图发射器视角窗口消息、控制云台消息分析处理，然后发布相机以及云台数据话题。

1.3 关键知识点

本实验主要是实现通过Python接口VisionCaptureApi.py使用（见RflySimAPIs\RflySimSDK\vision目录）发送取图请求。通过PX4MavCtrler.py（见RflySimAPIs\RflySimSDK\ctrl目录）使用创建控制接口，以此向RflySim发送控制指令，并使用UDP获取点云数据，将数据传输到定义的绘制点云数据的值，从而在虚拟机中绘制出点云图。关键代码解析如下：

本例子和其他分布式例子的区别，主要在于通过UDP直传方式传输图像数据，再对回传的图像进行数据的处理，并通过订阅截图发射器视角窗口消息、控制云台消息分析处理，然后发布相机以及云台数据话题。

关键知识点1：通过ReqCopterSim可以自动从局域网获取到仿真电脑的IP地址，从而自动建立连接，不再需要手动指定IP地址。不过，此种连接方式，可能在局域网中产生干扰（多台电脑同时打开多个CopterSim会产生误识别），不适合多个实验同时进行的场景。

1) 视觉接口使用

```
1 vis.jsonLoad() \# 加载Config.json中的传感器配置文件
2
3 isSuss = vis.sendReqToUE4(0, TargetIP) \#
4 向RflySim3D发送取图请求，发给ip为TargetIP的地址
5
6 vis.startImgCap() \# 开启取图
7
8 vis.hasData[i] \# 图片i数据是否更新
9
10 vis.Img[i] \# 图片i数据（像素矩阵）
```

2) ReqCopterSim接口使用（自动获取ip接口）

```
1 req = ReqCopterSim.ReqCopterSim() \# 获取局域网内所有CopterSim程序的电脑IP列表
2
3 TargetIP = req.getSimIpID(StartCopterID) \#
4 自动获取CopterSim的StartCopterID号程序所在电脑的IP，作为目标IP。这里获取CopterSim所在仿真电脑的IP。
5
6 vis = VisionCaptureApi.VisionCaptureApi(TargetIP) \#
7 创建一个视觉传感器实例，这个实例对应的ip号为TargetIP
8
9 req.sendReSimIP(CopterID) \# 请求mavlink数据到本电脑
```

3) 飞机控制指令

```
1 | mav = PX4MavCtrl.PX4MavCtrl(StartCopterID, TargetIP) \#创建飞机控制实例, id号为StartCopterID, ip为TargetIP
```

4) UE控制

接口详细使用方法见: [UE4CtrlAPI.py](#)

```
1 | ue = UE4CtrlAPI.UE4CtrlAPI() \# 创建UE控制实例
2
3 | ue.reqCamCoptObj(0, 0) \# 请求一个类型为0, id号为0物体。这里是请求seq_id为0的相机,
4
5 | ue.initUE4MsgRec() \# 初始化UDP数据监听
```

5) 其余代码说明

```
1 | def PTZCameraCtrl(data: CtrlCamera, seq_id: int) \#控制云台旋转,但是旋转需要有约束,当前版本VSIONCaptureAPI接口可直接控
2
3 | def CaptureCB(data: Bool, seq_id: int) \#在接收到特定的捕获指令时,从视觉传感器获取图像并保存到本地文件系统
4
5 | def PTZCameraInfoPub(event) \#发布云台想对于在载体坐标系数据,VisionCaptureAPI不能直接获取相机的属性
6
7 | ptz_camera_pub = rospy.Publisher("/rflsim/PTZ_camera_info", CameraInfo,queue_size=30) \#发布/rflsim/PTZ_camer
8
9 | ctrl_PTZCamera_sub = rospy.Subscriber("/camter_ctrl", CtrlCamera, PTZCameraCtrl,callback_args=(0), queue_size=
10
11 | rospy.Timer(rospy.Duration(1 / 30), PTZCameraInfoPub) \# 使用了 ROS的定时器功能,每隔1/30秒执行一次指定的回调函数
12
13 | rospy.spin() \# 持续执行回调函数,用于不断接收消息
```

2.实验效果

在Ubuntu下运行服务端,发送取图请求并通过UDP直传方式传输图像数据,再对回传的图像进行数据的处理,并通过订阅截图发射器视角窗口消息、控制云台消息分析处理,然后发布相机以及云台数据话题。

3.文件目录

例程目录: [\[安装目录\]\RflySimAPIs\8.RflySimVision\0.ApiExps\9.CameraInfo](#)

文件夹/文件名称	说明
camer_ctrl	客户端、服务端程序文件夹
rflsim_msgs	自定义ros消息文件夹
Python38Run.bat	Windows下Python程序运行脚本
WinWSL.bat	WSL1/Ubuntu 20.04环境程序运行脚本
WslGUI.bat	WSL1/Ubuntu 20.04可视化界面脚本

4. 运行环境

4.1 软件要求

Windows 10及以上版本；RflySim工具链；Visual Studio Code；Linux（Ubuntu 20.04）；Linux（Ubuntu 20.04）。

①：若使用Pixhawk 6X飞控，平台安装时的编译命令为：px4_fmu-v6x_default，推荐PX4固件版本为：1.12.3。其他配套飞控及编译命令请见：<https://rflysim.com/doc/zh/1/Hardware.html>

4.2 硬件要求

笔记本/台式电脑① 1台；WinWSL 1台；虚拟机/视觉盒子/其他板卡 可选台。

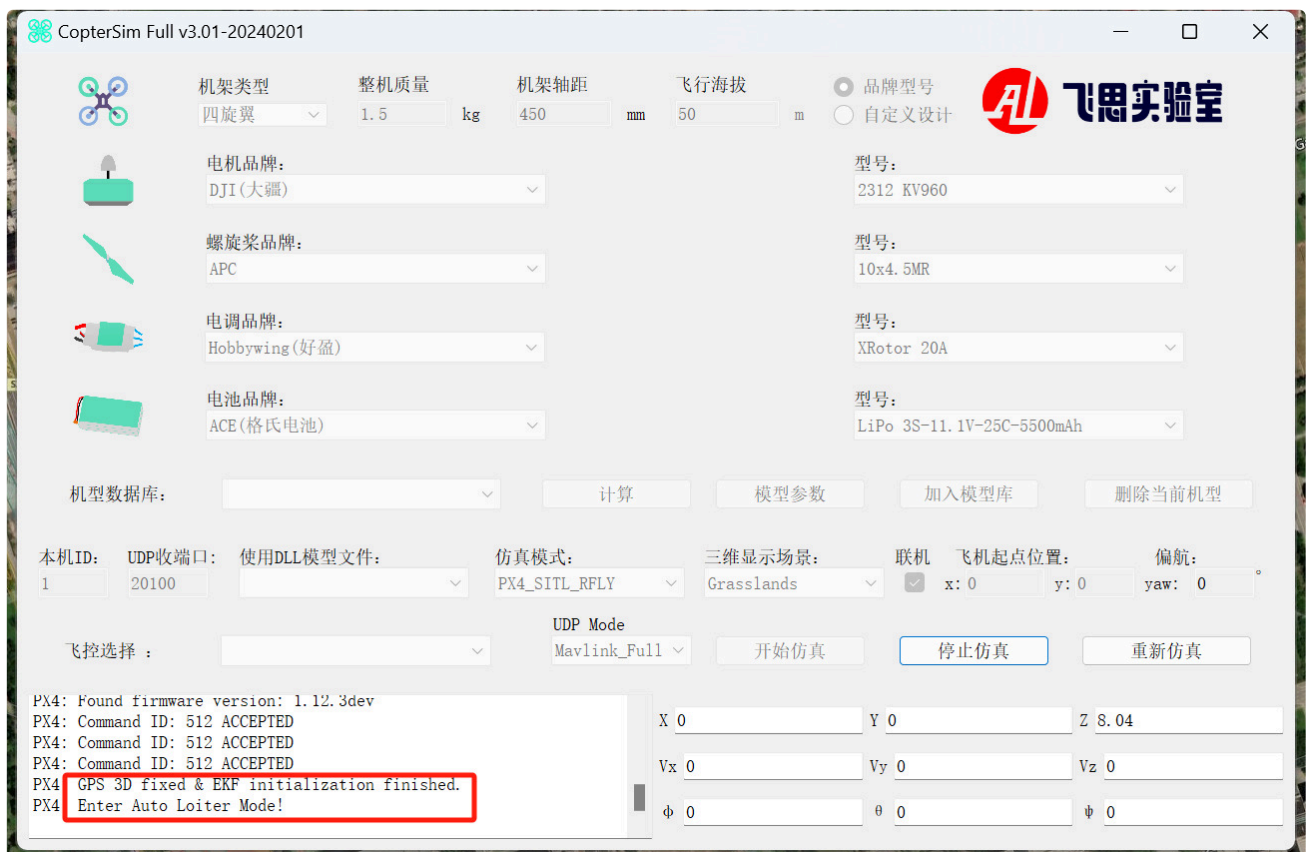
①：推荐配置请见：<https://rflysim.com/doc/zh/HowToInstall.pdf>

5. 实验步骤

5.1. 必做实验：WinsWSL控制

Step 1：开启仿真

在Windows下双击运行CameraCtrl.bat（在9.CameraInfo\camer_ctrl\scripts\Client文件夹里）开启一个飞机的软件在环仿真。将会启动1个QGC地面站，1个CopterSim软件且其软件下侧日志栏必须打印出GPS 3D fixed & EKF initialization finished字样代表初始化完成，并且RflySim3D软件内有1架无人机。





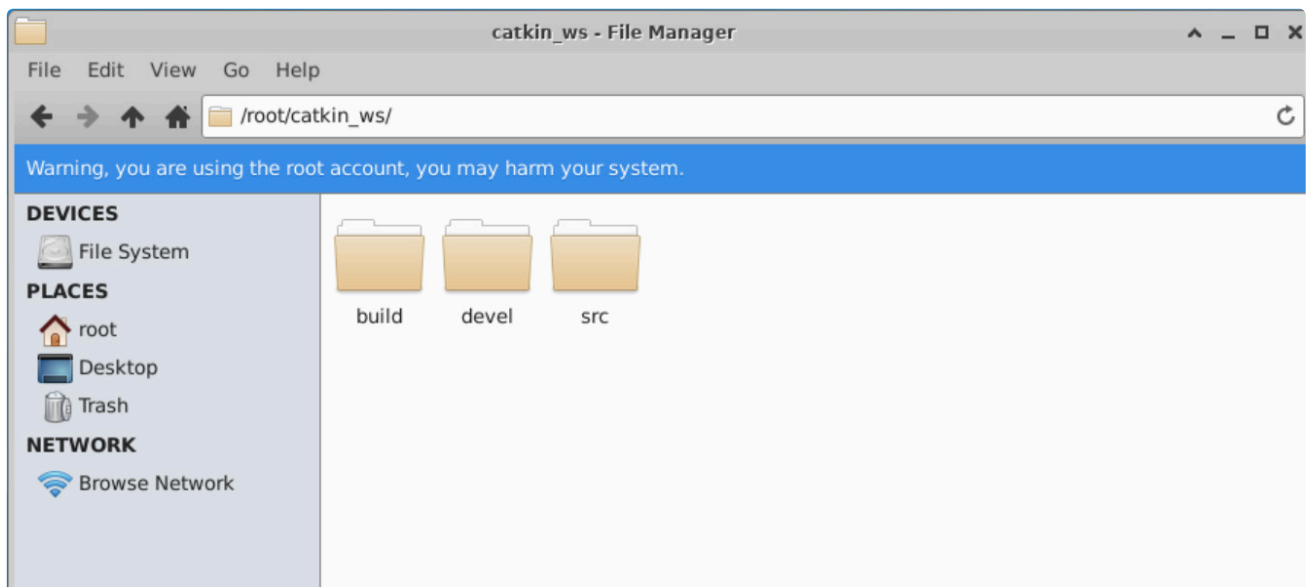
Step 2: 编译工作空间

双击打开 [WslGUI.bat](#)，启动WSL可视化界面。

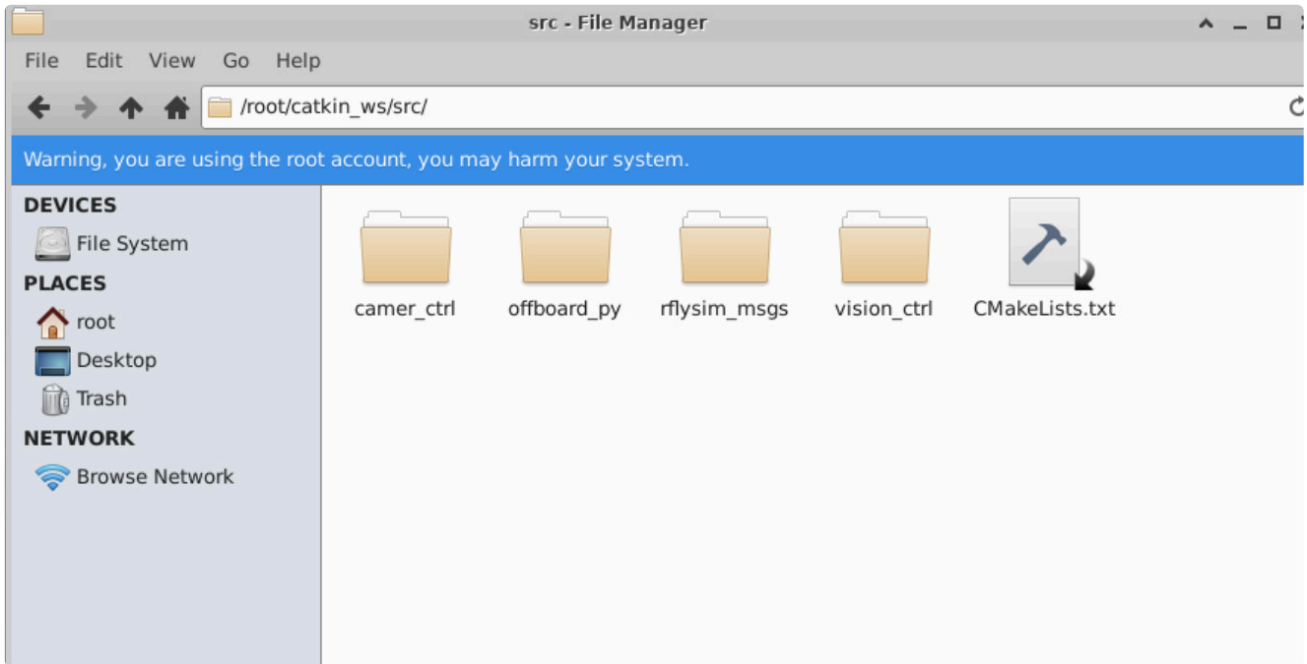
在ubuntu系统中创建一个工作空间输入如下命令

```
1 | mkdir \~/catkin_ws/src -p
2 |
3 | cd \~/catkin_ws/src
4 |
5 | catkin_init_workspace
6 |
7 | cd ..
8 |
9 | catkin_make
```

即可创建成功如下图



将本实验文件夹全拷贝到src目录下



回到~/catkin_ws目录下，右键点击空白处选择在终端打开，输入catkin_make进行编译，编译完成如下所示

```
rflsim@rflsim-virtual-machine: ~/临时/catkin_ws
Build space: /home/rflsim/临时/catkin_ws/build
Devel space: /home/rflsim/临时/catkin_ws/devel
Install space: /home/rflsim/临时/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/rflsim/临时/catkin_ws/build"
####
####
#### Running command: "make -j8 -l8" in "/home/rflsim/临时/catkin_ws/build"
####
[ 0%] Built target std_msgs_generate_messages_py
[ 0%] Built target std_msgs_generate_messages_cpp
[ 0%] Built target std_msgs_generate_messages_eus
[ 0%] Built target std_msgs_generate_messages_nodejs
[ 0%] Built target std_msgs_generate_messages_lisp
[ 0%] Built target _rflsim_msgs_generate_messages_check_deps_CtrlCamera
[ 0%] Built target _rflsim_msgs_generate_messages_check_deps_CameraInfo
[ 16%] Built target rflsim_msgs_generate_messages_cpp
[ 41%] Built target rflsim_msgs_generate_messages_py
[ 58%] Built target rflsim_msgs_generate_messages_lisp
[100%] Built target rflsim_msgs_generate_messages_nodejs
[ 83%] Built target rflsim_msgs_generate_messages_eus
[100%] Built target rflsim_msgs_generate_messages
```

输入

```
1 | source ~/.catkin_ws/devel/setup.bash
2 |
3 | source ~/.bashrc
```

也可以将 `source ~/.catkin_ws/devel/setup.bash` 手动写入 `~/.bashrc` 文件（注意通常情况下 `~/.bashrc` 文件是隐藏的，需要使用 `CTRL+H` 键显示，又或者使用 `gedit ~/.bashrc` 进行编辑）

另起一个终端，输入 `roscore` 启动 rosmaster


```

Terminal - root@RFLYSIM54: ~/catkin_ws/src/camer_ctrl/scripts/Sever
File Edit View Terminal Tabs Help
roscore http://RFLYSIM54:11311/ x root@RFLYSIM54: ~/catkin_ws/... x root@RFLYSIM54: ~/catkin_ws/... x
P: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---
header:
  seq: 20
  stamp:
    secs: 1720597627
    nsecs: 907494140
  frame_id: ''
seq_id: 0
data_width: 640
data_height: 480
FOV: 60
x: 0.0
y: 0.0
z: 0.0
yaw: 0.0
pitch: 0.0
roll: -0.0
D: []
K: [554.2562584220408, 0.0, 320.0, 0.0, 554.2562584220408, 240.0, 0.0, 0.0, 1.0]
R: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
P: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---

```

备注：可以参考 [安装目录]\RflySimAPIs\1.RflySimIntro\2.AdvExps\8.WsLVsCode\Intro.pdf 来使用VS Code开发并调试Ubuntu下python文件。

```

server_ue4.py X
Users > uavcs > Desktop > demo > 8.RflySimVision > 0.ApiExps > 0.Preparation > 5.ManModifyIPRun > server_ue4.py > ...
1
2 # import required libraries
3 # pip3 install pymavlink pyserial
4
5 import cv2
6 import numpy as np
7 import time
8 import VisionCaptureApi
9 import PX4MavCtrlV4 as PX4MavCtrl
10 import math
11
12 StartCopterID = 1 # 初始飞机的ID号
13 TargetIP = "192.168.31.141" # 手动修改为电脑主机的IP
14 # 注意：如果是本电脑运行的话，那TargetIP是127.0.0.1的本地地址；如果是远程访问，则是192打头的局域网地址。
15 # 因此本程序能同时在本机运行，也能在其他电脑运行。
16 vis = VisionCaptureApi.VisionCaptureApi(TargetIP)
17
18 # VisionCaptureApi 中的配置函数
19 vis.jsonLoad()
20 isSuss = vis.sendReqToUE4(
21     0, TargetIP
22 )
23 vis.startImgCap() # 开启取图循环，执行本语句之后，已经可以通过vis.Img[1]读取到图片了
24 print('Start Image Receiver')
25 #vis.sendImuReqCopterSim(StartCopterID, TargetIP) # 发送请求，从目标飞机CopterSim读取IMU数据，回传地址为127.0.0.1，默认频率为200Hz
26 # 执行本语句之后，会自动开启数据监听，已经可以通过vis.imu读取到IMU数据了。
27
28 VehilceNum = 1
29 MavList=[]
30 # Create MAV instance
31 for i in range(VehilceNum):
32     CopterID=StartCopterID+i # 当前配置的飞机序号
33
34     time.sleep(1)
35     MavList = MavList+[PX4MavCtrl.PX4MavCtrl(CopterID,TargetIP)] # 初始化并建立i号飞机的MAVLink通信连接
36

```

Step 4: 结束仿真

在下图“CameraCtrl.bat”脚本开启的命令提示符CMD窗口中，按下回车键（任意键）就能快速关闭CopterSim、QGC、RflySim3D等所有程序。

5.2. 选作实验

准备工作：

虚拟机或NX的配置方法是相同的。

1) Ubuntu虚拟机环境下，进行分布式联机实验。先参

考 [安装目录]\RflySimAPIs\8.RflySimVision\0.ApiExps\0.Preparation\1.VMwareUbuntu\Readme.pdf，完成虚拟机的下载与配置。

2) 用第二台Ubuntu电脑或NX板卡，实现联机实验。其他Ubuntu电脑的配置，先看

[安装目录]\RflySimAPIs\8.RflySimVision\0.ApiExps\0.Preparation\2.GenenalUbuntuConfig\Readme.pdf

；NX板卡的配置方法，先

看 [安装目录]\RflySimAPIs\8.RflySimVision\0.ApiExps\0.Preparation\3.NXwithPX4Config\Readme.pdf

。

扩展实验：

5.2.1在虚拟机/视觉板卡/另一台Ubuntu上接收图像实验

Step 1：开启仿真

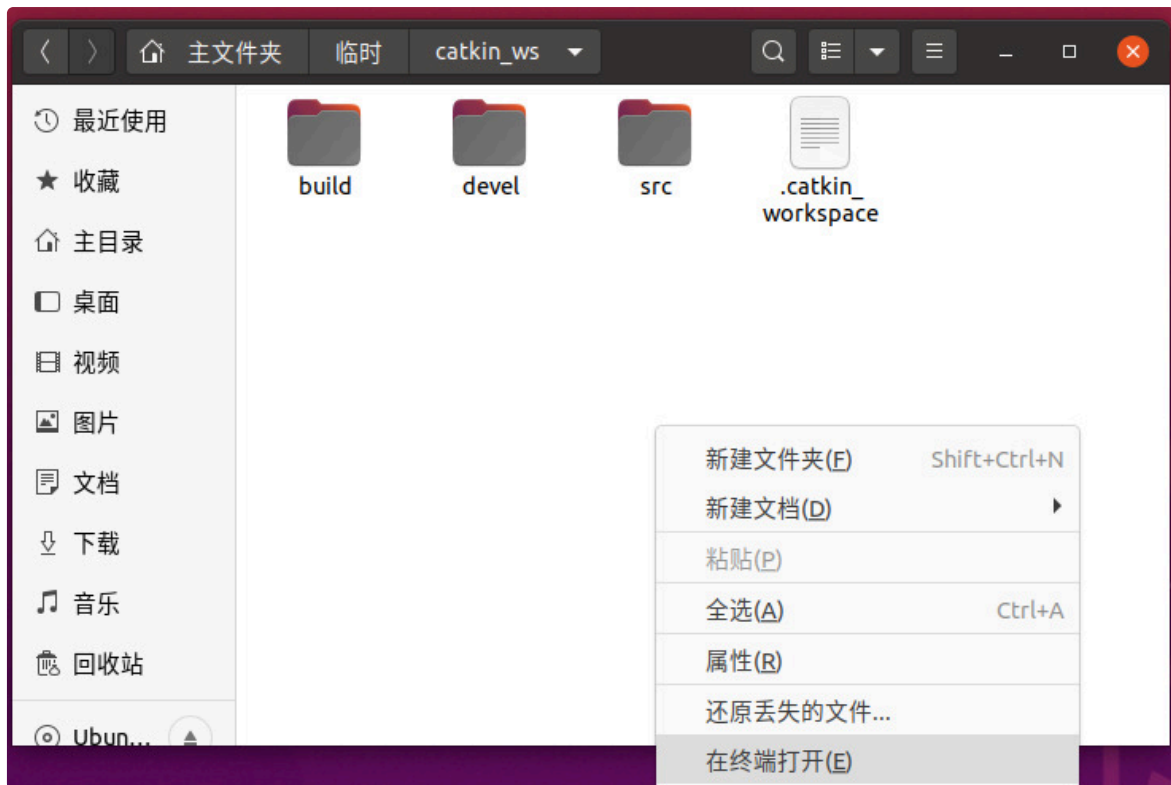
步骤1同上面的Step1步骤。

Step 2：编译工作空间

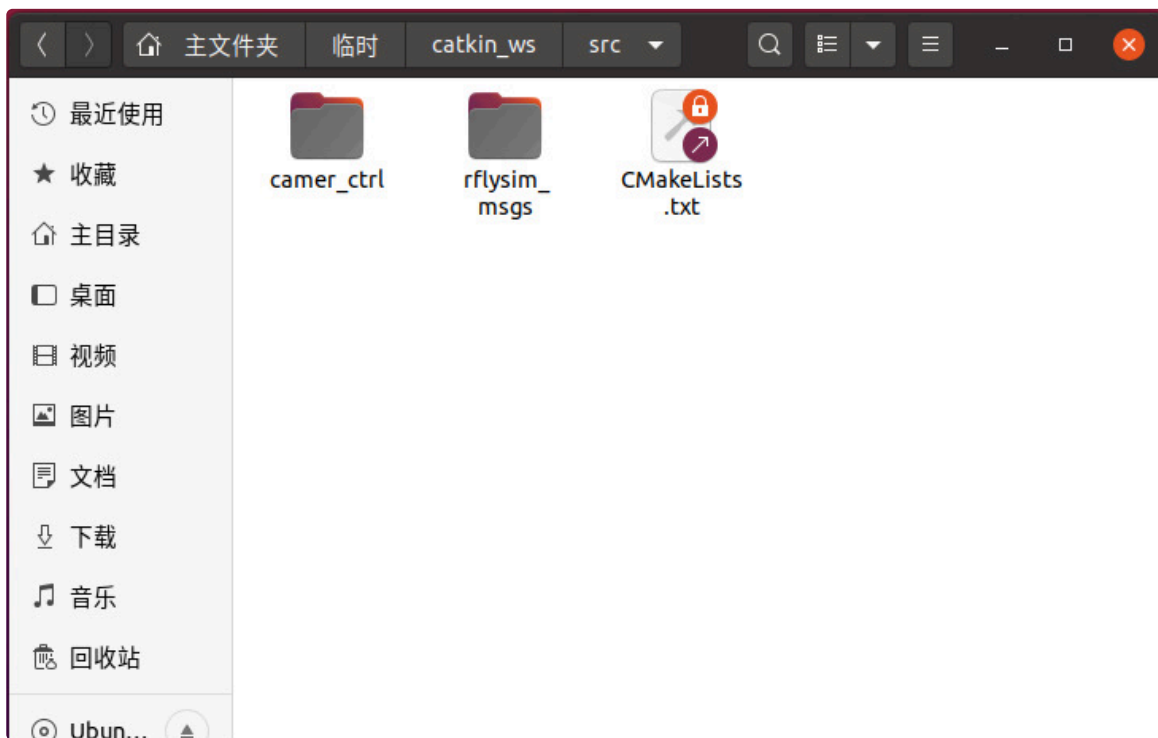
在ubuntu系统中创建一个工作空间输入如下命令

```
1 | mkdir ~/catkin_ws/src -p
2 |
3 | cd ~/catkin_ws/src
4 |
5 | catkin_init_workspace
6 |
7 | cd ..
8 |
9 | catkin_make
```

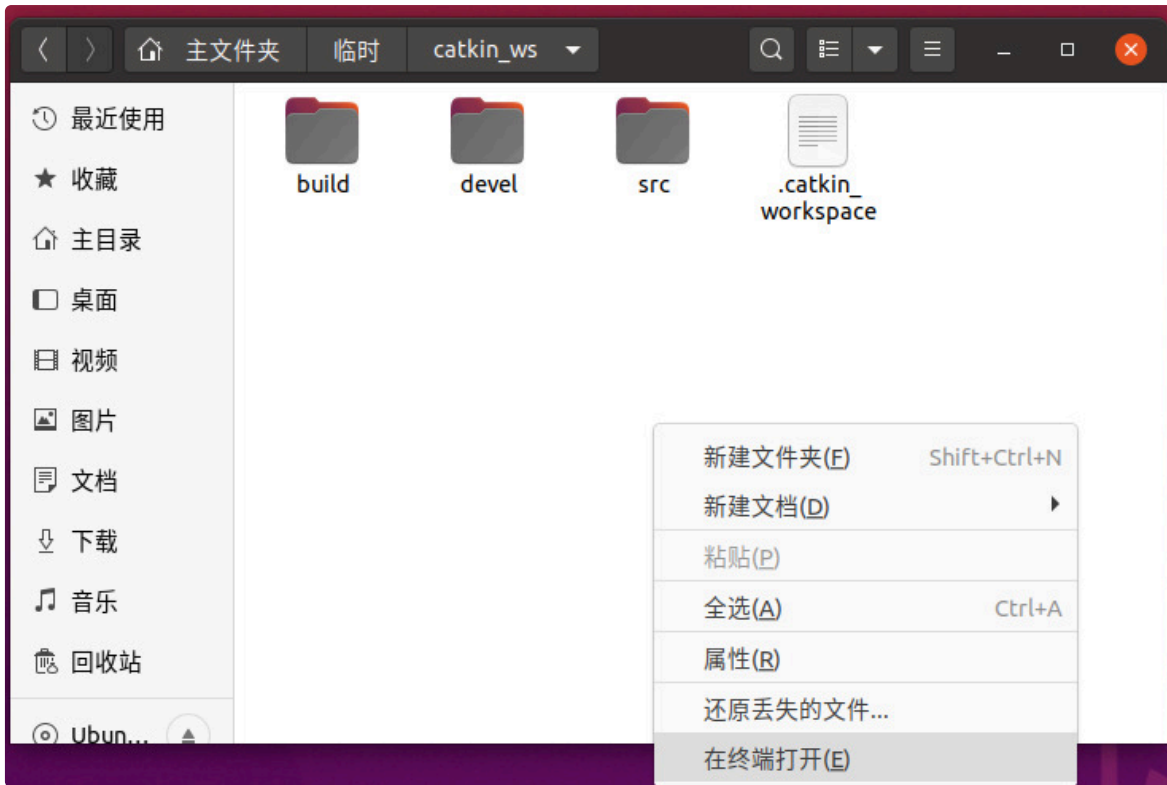
即可创建成功如下图



将本实验文件夹全拷贝到src目录下



回到~/catkin_ws目录下，右键点击空白处选择在终端打开



输入catkin_make进行编译，编译完成如下所示

```
rflsim@rflsim-virtual-machine: ~/临时/catkin_ws
Build space: /home/rflsim/临时/catkin_ws/build
Devel space: /home/rflsim/临时/catkin_ws/devel
Install space: /home/rflsim/临时/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/rflsim/临时/catkin_ws/build"
####
####
#### Running command: "make -j8 -l8" in "/home/rflsim/临时/catkin_ws/build"
####
[ 0%] Built target std_msgs_generate_messages_py
[ 0%] Built target std_msgs_generate_messages_cpp
[ 0%] Built target std_msgs_generate_messages_eus
[ 0%] Built target std_msgs_generate_messages_nodejs
[ 0%] Built target std_msgs_generate_messages_lisp
[ 0%] Built target _rflsim_msgs_generate_messages_check_deps_CtrlCamera
[ 0%] Built target _rflsim_msgs_generate_messages_check_deps_CameraInfo
[ 16%] Built target rflsim_msgs_generate_messages_cpp
[ 41%] Built target rflsim_msgs_generate_messages_py
[ 58%] Built target rflsim_msgs_generate_messages_lisp
[100%] Built target rflsim_msgs_generate_messages_nodejs
[ 83%] Built target rflsim_msgs_generate_messages_eus
[100%] Built target rflsim_msgs_generate_messages
```

输入

```
1 | source ~/catkin_ws/devel/setup.bash
2 |
3 | source ~/.bashrc
```

也可以将 `source ~/catkin_ws/devel/setup.bash` 手动写入.bashrc文件（注意通常情况下.bashrc文件是隐藏的，需要使用CTRL+H键显示，又或者使用 `gedit ~/.bashrc` 进行编辑）

另起一个终端，输入 `roscore` 启动rosmaster

```
roscore http://rfllysim-virtual-machine:11311/
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://rfllysim-virtual-machine:40695/
ros_comm version 1.16.0

SUMMARY
=====

PARAMETERS
* /rostdistro: noetic
* /rosversion: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [11371]
ROS_MASTER_URI=http://rfllysim-virtual-machine:11311/

setting /run_id to 4146dc80-c0ab-11ee-9ae6-37baca2caaa6
process[rosout-1]: started with pid [11390]
started core service [/rosout]
```

Step 3: 运行控制程序

进入到camer_ctrl\scripts\Sever路径下在终端中输入 `python3 server.py` 命令启动服务端程序进行对图像数据的处理。然后就可以监听到话题/rfllysim/PTZ_camera_info数据。

```
rfllysim@rfllysim-virtual-machine: ~
rfllysim@rfllysim-virtual-machine:~$ rostopic echo /rfllysim/PTZ_camera_info
header:
  seq: 1
  stamp:
    secs: 1706755950
    nsecs: 673694371
  frame_id: ''
seq_id: 0
data_width: 640
data_height: 480
FOV: 60
x: 0.0
y: 0.0
z: 0.0
yaw: 0.0
pitch: 0.0
roll: -0.0
D: []
K: [554.2562584220408, 0.0, 320.0, 0.0, 554.2562584220408, 240.0, 0.0, 0.0, 1.0]
R: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
P: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---
```

6.参考资料

无

7.常见问题

Q1: 无

A1: 无