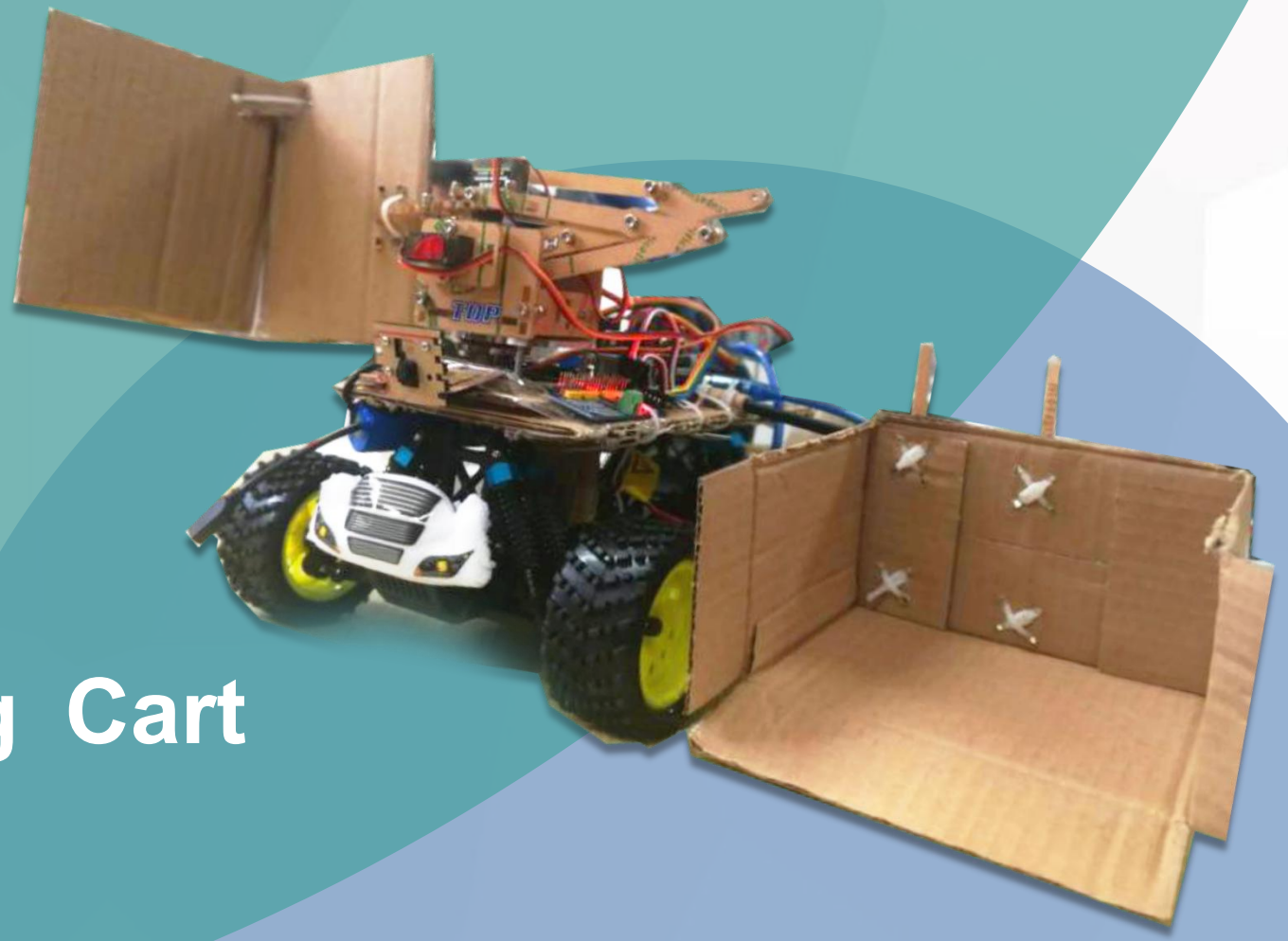
垃圾分拣小车

The Garbage Sorting Cart

小车驾驶与机械臂控制
模型训练与串口通信



目录

CONTENTS

PART 1

背景与意义

背景与意义

- **全国人均每天产生生活垃圾约1.16kg**
- **13.7亿人*1.16kg/人=160万吨**

PART 2

主要研究内容

研究主要内容

- 摄像头对准路面
- 检测到垃圾后靠近
- 到垃圾前停止
- 拍照分类
- 机械臂铲起垃圾
- 驶向垃圾站点
- 分类投放垃圾

PART 3

总体架构及主要步骤

设计思路—三个模块

寻找垃圾

小车根据摄像头的拍摄内容内是否存在垃圾来选择行为

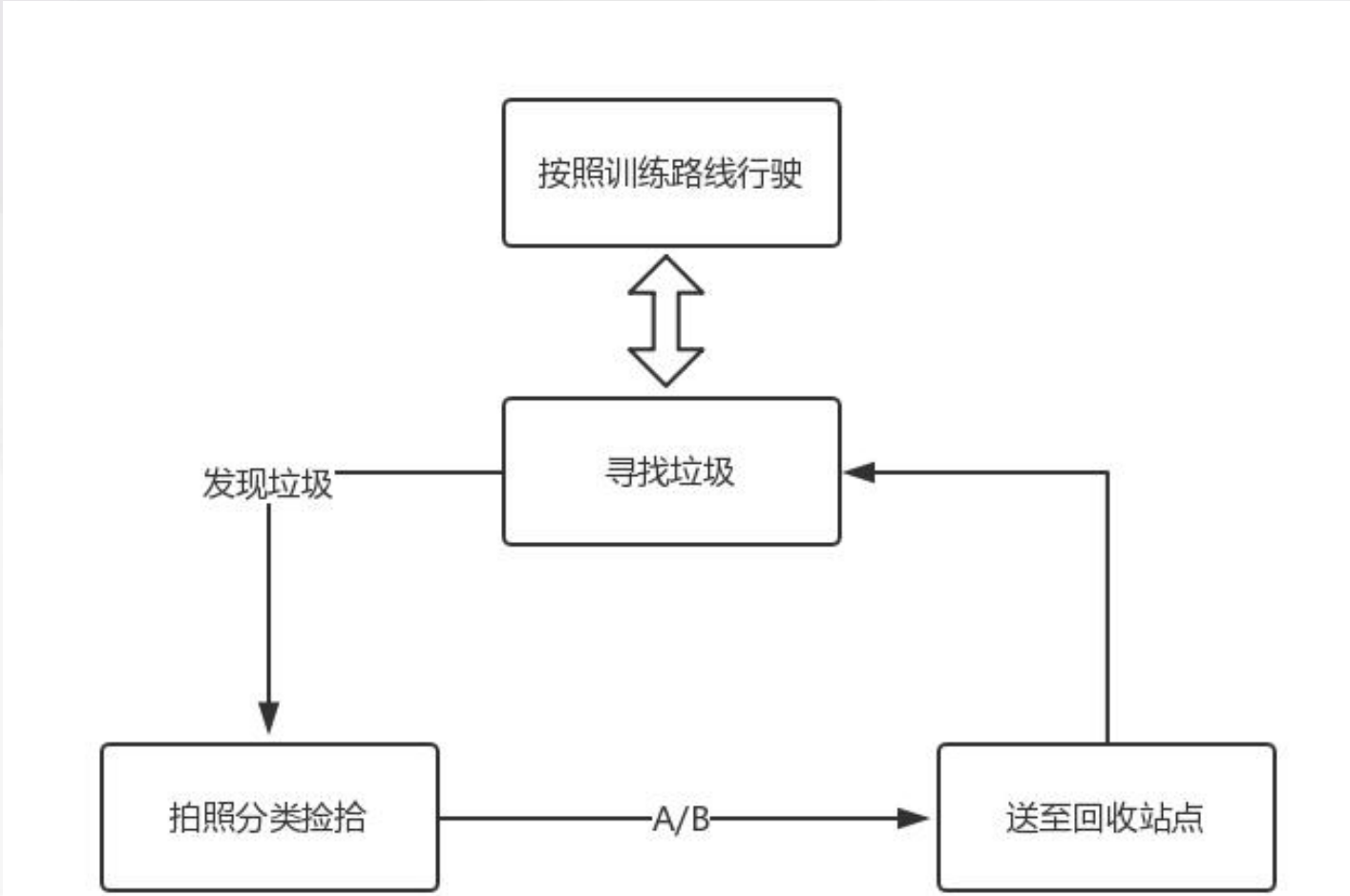
捡拾分类

拍照上传服务器，示例分类A/B，机械臂同时捡拾垃圾

垃圾回收

依据A/B，启动不同回收模型，将垃圾运送至回收站点.

设计思路—主要步骤

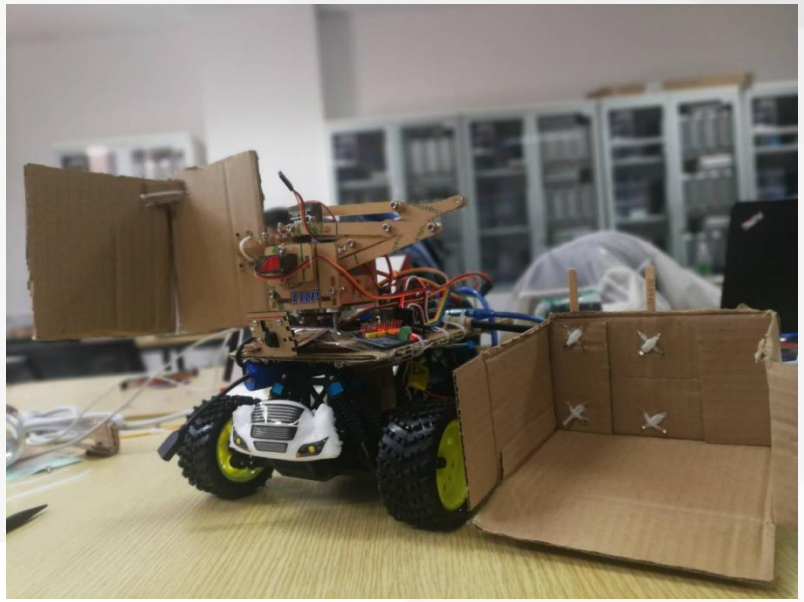


PART 4

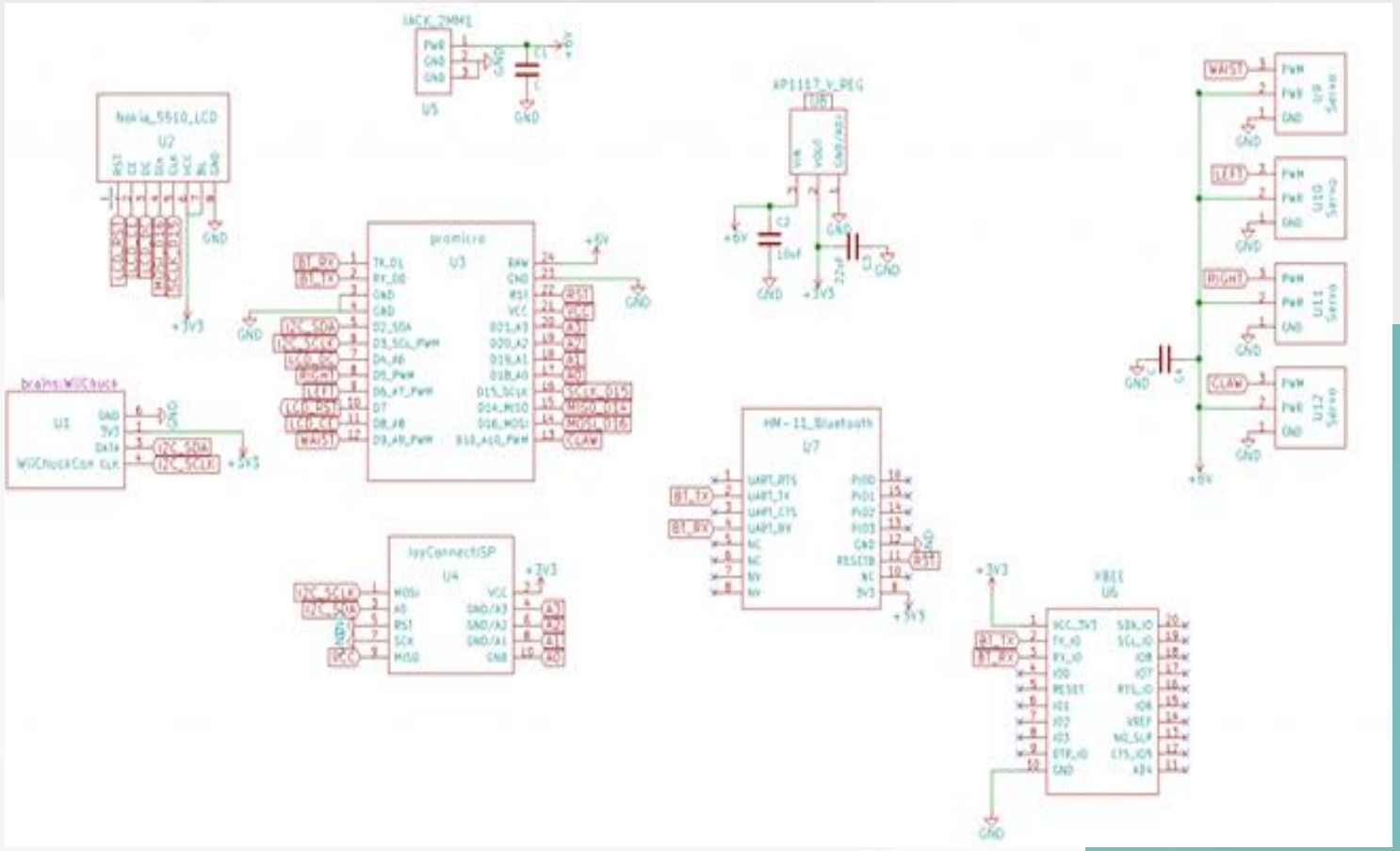
模块设计与调试实现（硬件—软件）

硬件部分

小车　机械臂　树莓派



硬件部分-机械臂



关键点

确定舵机控制位点
制作“爪子”
固定垃圾框
调试指令
arduino与树莓派的串口通信
调试机械臂与垃圾桶的相对位置

硬件部分-树莓派

关键点

路径规划
控制信号发出端
控制相机对垃圾进行拍照
对机械臂的不同动作的触发控制
与服务器的串口通信
……

硬件部分-小车训练

关键点

马力值调试
摄像头调试
模型融合与转换（难点）
.....

软件部分—donkercar路线规划

1.DonkeyCar的模型简述

模型一：特定轨迹巡逻，寻找垃圾且在垃圾面前停止 。

模型二：若捡到A类垃圾，则垃圾收入后送至A类垃圾站点。

模型三：若捡到B类垃圾，则垃圾收入后送至B类垃圾站点。

不同模型可以自动切换

软件部分—donkercar路线规划

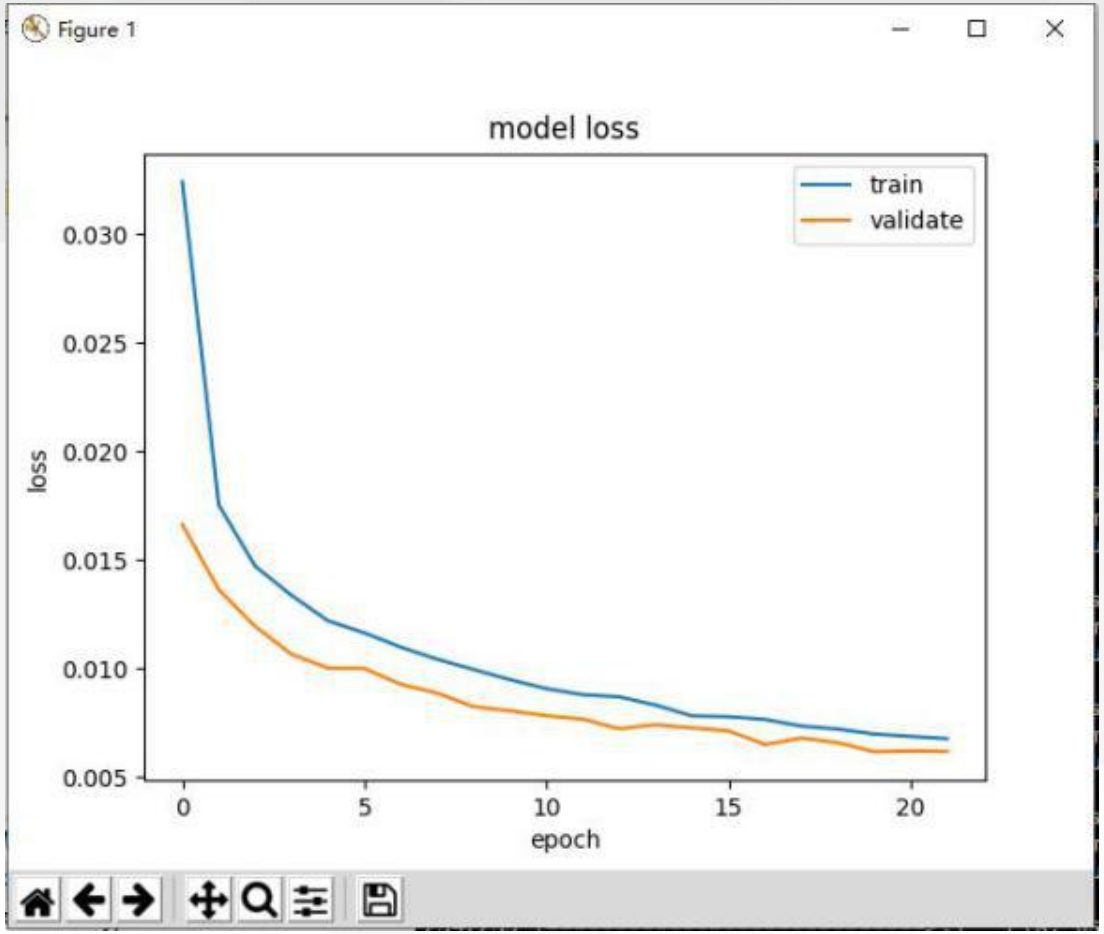
2、donkeycar模型切换（创新）

开机完全自启：manage.py增设全局变量，修改其他对应文件
模型开关控制：self.on+连续马力值判断
调用回收模型：以1s的间隔不停读取sort.txt，得到信号。

难点：donkeycar封装性较好，寻找马力值、角度、网页等的数据接口，突破不同语言的串口通信界限。

软件部分—donkercar路线规划

3.DonkeyCar的数据采集与模型训练（示例）



大车及其不稳定，训练一周半后换了小车从头再来

loss曲线经历了大约23次epoch收敛并停止了训练，不能过度训练以防止模型过拟合。

软件部分—yolov3垃圾分类图像识别

1.使用YOLOv3架构区分垃圾的种类

yolo v3

使用Darknet-53的网络结构（含有53个卷积层），它借鉴了残差网络Residual network的做法，在一些层之间设置了快捷链路，采用了3个不同尺度的特征图来进行对象检测，采用K-means聚类得到先验框的尺寸，为每种下采样尺度设定3种先验框，总共聚类出9种尺寸的先验框，在精确度相当的情况下，YOLOv3的速度是其它模型的3、4倍。考虑到训练数据较大，采用了GPU服务器并自己配置环境进行训练。

软件部分—yolov3垃圾分类图像识别

2.图像分类数据集处理

```
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 82 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 94 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000257, .5R: -nan, .75R: -nan,  count: 0
Region 106 Avg IOU: 0.759684, Class: 0.999502, Obj: 0.999943, No Obj: 0.000111, .5R: 1.000000, .75R: 1.000000,  count: 1
Region 82 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 94 Avg IOU: 0.947585, Class: 0.999666, Obj: 0.999922, No Obj: 0.000600, .5R: 1.000000, .75R: 1.000000,  count: 1
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 82 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 94 Avg IOU: 0.872336, Class: 0.999936, Obj: 0.999999, No Obj: 0.000306, .5R: 1.000000, .75R: 1.000000,  count: 1
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 82 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 94 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000253, .5R: -nan, .75R: -nan,  count: 0
Region 106 Avg IOU: 0.752868, Class: 0.999733, Obj: 0.999975, No Obj: 0.000148, .5R: 1.000000, .75R: 1.000000,  count: 1
30962: 0.023960, 0.014048 avg, 0.002000 rate, 1.431334 seconds, 990784 images
Loaded: 0.000059 seconds
Region 82 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
```

labellmg 用于人工处理收集到的数据集，首先生成对应图片txt格式（yolo要求标准格式，xml也可），而后通过编写代码自动划分数据集以及配置数据集接口进行训练。

后期评估网络mAP时，需要将数据集转换为voc格式。

软件部分—yolov3垃圾分类图像识别

3.示例训练识别结果

```
   94 yolo
   95 route  91
   96 conv    128  1 x 1 / 1    26 x  26 x 256   ->    26 x  26 x 128  0.044 BFLOPs
   97 upsample           2x    26 x  26 x 128   ->    52 x  52 x 128
   98 route  97 36
   99 conv    128  1 x 1 / 1    52 x  52 x 384   ->    52 x  52 x 128  0.266 BFLOPs
  100 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1.595 BFLOPs
  101 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128  0.177 BFLOPs
  102 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1.595 BFLOPs
  103 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128  0.177 BFLOPs
  104 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1.595 BFLOPs
  105 conv     24  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x  24  0.033 BFLOPs
  106 yolo
Loading weights from /home/prj04/xiaoxueqi/darknet/lyx/backup/yolov3-voc.backup...Done!
/home/prj04/xiaoxueqi/darknet/lyx/174_cam-image_array_.jpg: Predicted in 0.020086 seconds.
3
matal: 100%
```

```
   96 conv    128  1 x 1 / 1    26 x  26 x 256   ->    26 x  26 x 128  0.044 BFLOPs
   97 upsample           2x    26 x  26 x 128   ->    52 x  52 x 128
   98 route  97 36
   99 conv    128  1 x 1 / 1    52 x  52 x 384   ->    52 x  52 x 128  0.266 BFLOPs
  100 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1.595 BFLOPs
  101 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128  0.177 BFLOPs
  102 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1.595 BFLOPs
  103 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128  0.177 BFLOPs
  104 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1.595 BFLOPs
  105 conv     24  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x  24  0.033 BFLOPs
  106 yolo
Loading weights from /home/prj04/xiaoxueqi/darknet/lyx/backup/yolov3-voc.backup...Done!
/home/prj04/xiaoxueqi/darknet/lyx/105_cam-image_array_.jpg: Predicted in 0.020069 seconds.
3
plastic: 100%
```

该图片经过识别被标记为metal

由上图可知，174和105图像分别识别为metal和plastic，时延为0.020086s和0.20096s。

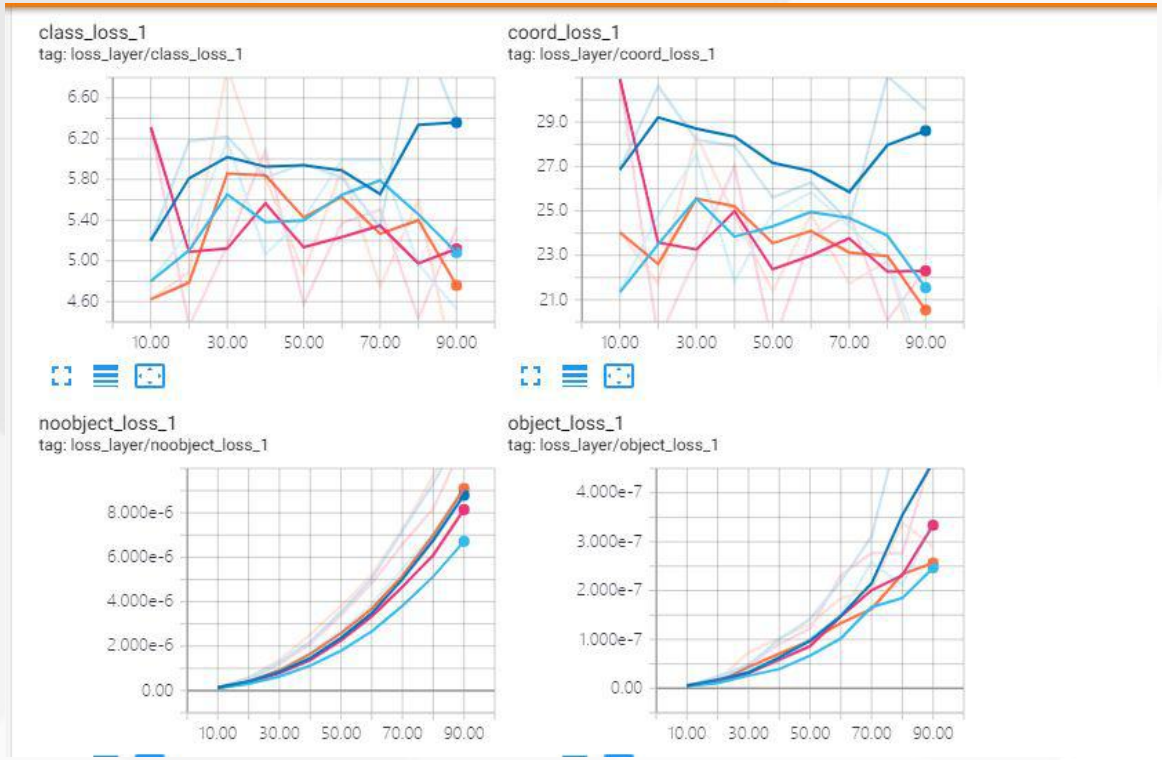
软件部分—yolov3垃圾分类图像识别

4.简述使用过的网络及其优缺点

尝试过四个最基础的神经网络:
inception_net，alexnet，
restnet，mobilenet，vgg

yolo1-yolo2...
yolov3比较合适

尝试最新的centernet以及cornernet的2019年四五月份新出的onestage网络



tensorboard展示训练过程 mAP约为89
（yolo训练了2个小时）

软件部分—串口通信

1.树莓派与auduino的通信

树莓派：
串口库
ls /dev/tty*与/dev/ttyUSB0设置端口
波特率9600

arduino：
预置指令

（不要多次插拔，否则端口可能变化）

软件部分—串口通信

2.树莓派向服务器发送数据

```
s=socket()
s.connect(('10.110.210.24', 2578))
f=open(r'/home/pi/mycar/test0.jpg', 'rb')
while True:
    data=f.read(1024)
    if not data:
        break
    s.send(data)
f.close()
s.close()
```

以二进制文件发送图片数据

```
s=socket()
s.bind(('10.110.210.24',2578))
s.listen(5)
c,addr=s.accept()
print('connnect from ',addr)
f=open(r'/home/prj04/xiaoxueqi/darknet/test0.jpg','wb')
while True:
    data=c.recv(1024)
    #print(data)
    if not data:
        break
    f.write(data)
f.close()
c.close()
s.close()
```

软件部分—串口通信

3.pc向树莓派发送数据

```python
def socket_service_data():
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        #s.bind(('127.0.0.1', 6666))
        s.bind(('10.128.205.61', 6665))
        s.listen(10)
    except socket.error as msg:
        print(msg)
        sys.exit(1)
    print("Wait for Connection.....................")
    ff=open(r'/home/pi/mycar/sort.txt','wb')
    while True:
        sock, addr = s.accept()
        buf = sock.recv(1024)
        if not buf:
            break
        buf = buf.decode()
        ff.write(buf)
        print("The data from " + str(addr[0]) + " is " + str(buf))
        print("Successfully")
        # return buf
        # sock.close()
    ff.close()
if __name__ == '__main__':
    socket_service_data()
```

```python
def read():
    with open(r'C:\Users\liyunxuan\Desktop\test.txt','r',encoding='u
            lines=[] # 创建了一个空列表，里面没有元素
            for line in f.readlines():
                if line!='\n':
                    lines.append(line.strip('\n'))
            print(lines)
    return lines
def commun(Line):
    s=socket()
    #ip地址写树莓派的
    s.connect(('10.128.205.61',6665))
    s.send((line).encode("utf-8"))
    print(line)
    s.close()

if __name__ == '__main__':
    while True:
        time.sleep(0.5)
        read()
        if read()[0]!=str(0):
            commun(read()[0])
            break
```

服务器与PC同步数据后，树莓派收到的数据被写入一个txt文件中

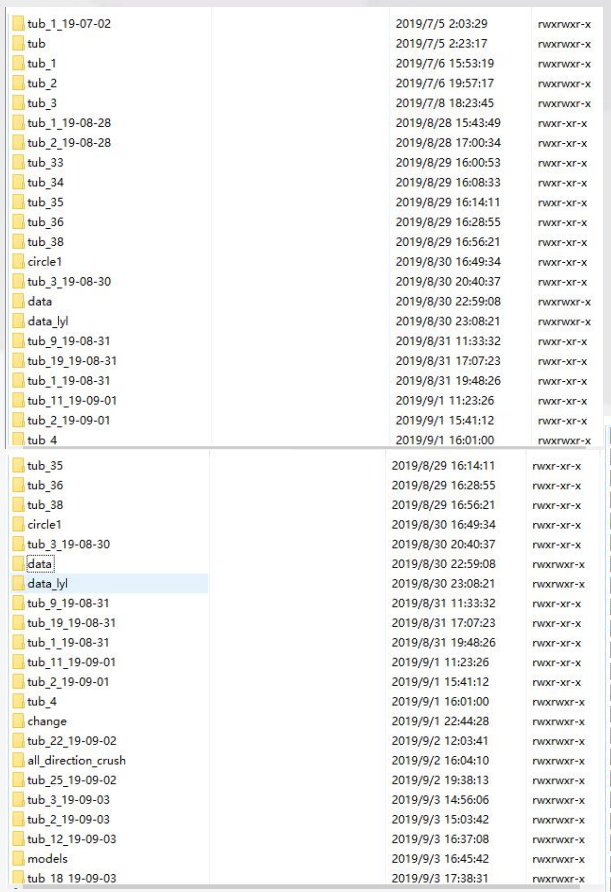
PART 5

课题成果及性能分析

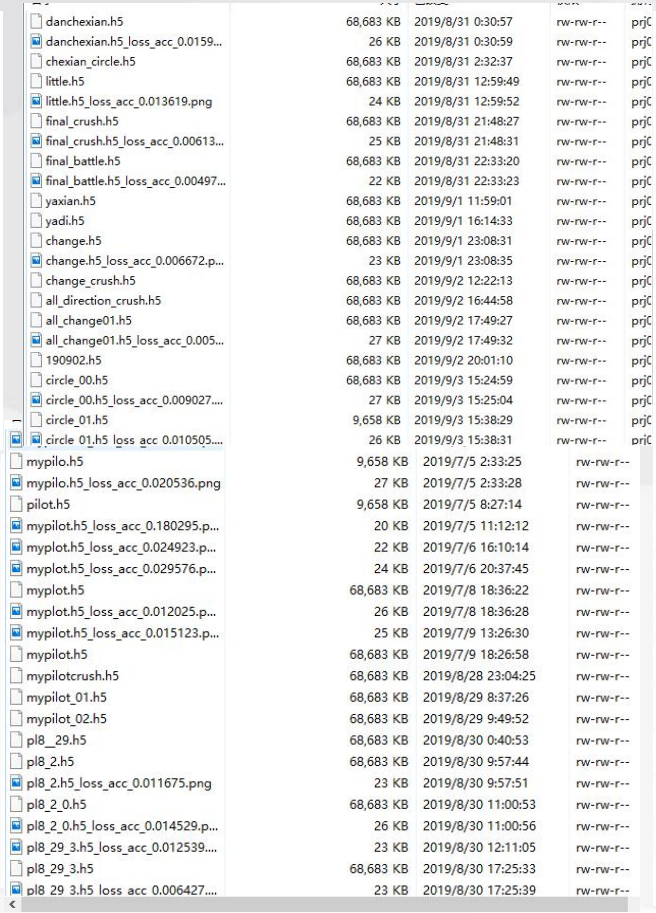
主要实验数据

主要包括两部分：训练小车、树莓派与服务器与PC与arduino的通信，d2包含了部分使用的数据

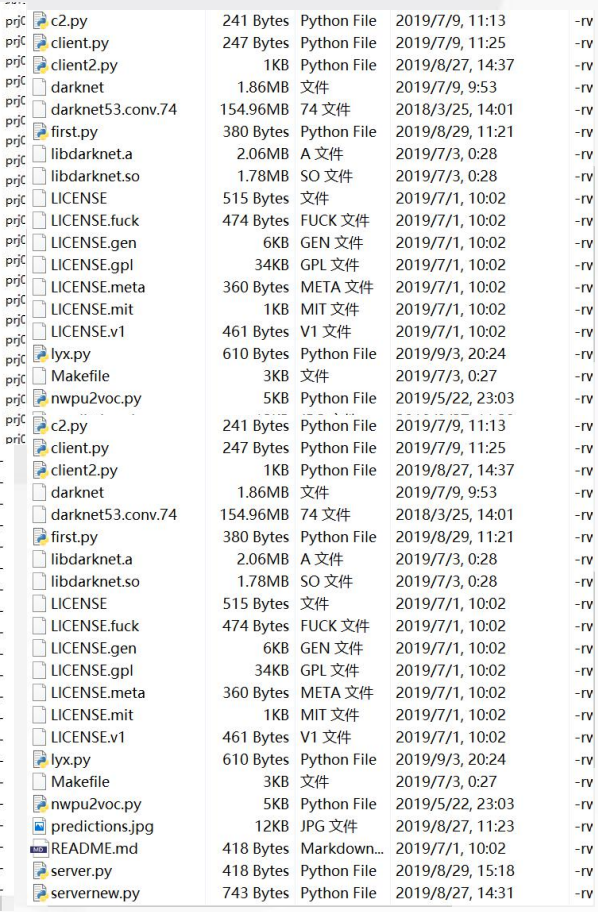
部分数据



tub_1_19-07-02	2019/7/5 2:03:29	rwxrwxr-x	
tub	2019/7/5 2:23:17	rwxrwxr-x	
tub_1	2019/7/6 15:53:19	rwxrwxr-x	
tub_2	2019/7/6 19:57:17	rwxrwxr-x	
tub_3	2019/7/8 18:23:45	rwxrwxr-x	
tub_1_19-08-28	2019/8/28 15:43:49	rwxr-xr-x	
tub_2_19-08-28	2019/8/28 17:00:34	rwxr-xr-x	
tub_33	2019/8/29 16:00:53	rwxr-xr-x	
tub_34	2019/8/29 16:08:33	rwxr-xr-x	
tub_35	2019/8/29 16:14:11	rwxr-xr-x	
tub_36	2019/8/29 16:28:55	rwxr-xr-x	
tub_38	2019/8/29 16:56:21	rwxr-xr-x	
circle1	2019/8/30 16:49:34	rwxr-xr-x	
tub_3_19-08-30	2019/8/30 20:40:37	rwxr-xr-x	
data	2019/8/30 22:59:08	rwxrwxr-x	
data_lyl	2019/8/30 23:08:21	rwxrwxr-x	
tub_9_19-08-31	2019/8/31 11:33:32	rwxr-xr-x	
tub_19_19-08-31	2019/8/31 17:07:23	rwxr-xr-x	
tub_1_19-08-31	2019/8/31 19:48:26	rwxr-xr-x	
tub_11_19-09-01	2019/9/1 11:23:26	rwxr-xr-x	
tub_2_19-09-01	2019/9/1 15:41:12	rwxr-xr-x	
tub_4	2019/9/1 16:01:00	rwxrwxr-x	
change	2019/9/2 22:44:28	rwxrwxr-x	
tub_22_19-09-02	2019/9/2 12:03:41	rwxrwxr-x	
all_direction_crush	2019/9/2 16:04:10	rwxrwxr-x	
tub_25_19-09-02	2019/9/2 19:38:13	rwxrwxr-x	
tub_3_19-09-03	2019/9/3 14:56:06	rwxrwxr-x	
tub_2_19-09-03	2019/9/3 15:03:42	rwxrwxr-x	
tub_12_19-09-03	2019/9/3 16:37:08	rwxrwxr-x	
models	2019/9/3 16:45:42	rwxrwxr-x	
tub_18_19-09-03	2019/9/3 17:38:31	rwxrwxr-x	



danchexian.h5	68,683 KB	2019/8/31 0:30:57	rw-rw-r--
danchexian.h5_loss_acc_0.0159...	26 KB	2019/8/31 0:30:59	rw-rw-r--
chexian_circle.h5	68,683 KB	2019/8/31 2:32:37	rw-rw-r--
little.h5	68,683 KB	2019/8/31 12:59:49	rw-rw-r--
little.h5_loss_acc_0.013619.png	24 KB	2019/8/31 12:59:52	rw-rw-r--
final_crush.h5	68,683 KB	2019/8/31 21:48:27	rw-rw-r--
final_crush.h5_loss_acc_0.00613...	25 KB	2019/8/31 21:48:31	rw-rw-r--
final_battle.h5	68,683 KB	2019/8/31 22:33:20	rw-rw-r--
final_battle.h5_loss_acc_0.00497...	22 KB	2019/8/31 22:33:23	rw-rw-r--
yaxian.h5	68,683 KB	2019/9/1 11:59:01	rw-rw-r--
yadi.h5	68,683 KB	2019/9/1 16:14:33	rw-rw-r--
change.h5	68,683 KB	2019/9/1 23:08:31	rw-rw-r--
change.h5_loss_acc_0.006672.p...	23 KB	2019/9/1 23:08:35	rw-rw-r--
change_crush.h5	68,683 KB	2019/9/2 12:22:13	rw-rw-r--
all_direction_crush.h5	68,683 KB	2019/9/2 16:44:58	rw-rw-r--
all_change01.h5	68,683 KB	2019/9/2 17:49:27	rw-rw-r--
all_change01.h5_loss_acc_0.005...	27 KB	2019/9/2 17:49:32	rw-rw-r--
190902.h5	68,683 KB	2019/9/2 20:01:10	rw-rw-r--
circle_00.h5	68,683 KB	2019/9/3 15:24:59	rw-rw-r--
circle_00.h5_loss_acc_0.009027....	27 KB	2019/9/3 15:25:04	rw-rw-r--
circle_01.h5	9,658 KB	2019/9/3 15:38:29	rw-rw-r--
circle_01.h5_loss_acc_0.010505....	26 KB	2019/9/3 15:38:31	rw-rw-r--
mypilo.h5	9,658 KB	2019/7/5 2:33:25	rw-rw-r--
mypilo.h5_loss_acc_0.020536.png	27 KB	2019/7/5 2:33:28	rw-rw-r--
pilot.h5	9,658 KB	2019/7/5 8:27:14	rw-rw-r--
mypilot.h5_loss_acc_0.180295.p...	20 KB	2019/7/5 11:12:12	rw-rw-r--
myplot.h5_loss_acc_0.024923.p...	22 KB	2019/7/6 16:10:14	rw-rw-r--
myplot.h5_loss_acc_0.029576.p...	24 KB	2019/7/6 20:37:45	rw-rw-r--
myplot.h5	68,683 KB	2019/7/8 18:36:22	rw-rw-r--
myplot.h5_loss_acc_0.012025.p...	26 KB	2019/7/8 18:36:28	rw-rw-r--
mypilot.h5_loss_acc_0.015123.p...	25 KB	2019/7/9 13:26:30	rw-rw-r--
mypilot.h5	68,683 KB	2019/7/9 18:26:58	rw-rw-r--
mypilotcrush.h5	68,683 KB	2019/8/28 23:04:25	rw-rw-r--
mypilot_01.h5	68,683 KB	2019/8/29 8:37:26	rw-rw-r--
mypilot_02.h5	68,683 KB	2019/8/29 9:49:52	rw-rw-r--
pl8__29.h5	68,683 KB	2019/8/30 0:40:53	rw-rw-r--
pl8_2.h5	68,683 KB	2019/8/30 9:57:44	rw-rw-r--
pl8_2.h5_loss_acc_0.011675.png	23 KB	2019/8/30 9:57:51	rw-rw-r--
pl8_2_0.h5	68,683 KB	2019/8/30 11:00:53	rw-rw-r--
pl8_2_0.h5_loss_acc_0.014529.p...	26 KB	2019/8/30 11:00:56	rw-rw-r--
pl8_29_3.h5_loss_acc_0.012539....	23 KB	2019/8/30 12:11:05	rw-rw-r--
pl8_29_3.h5	68,683 KB	2019/8/30 17:25:33	rw-rw-r--
pl8_29_3.h5_loss_acc_0.006427....	23 KB	2019/8/30 17:25:39	rw-rw-r--



c2.py	241 Bytes	Python File	2019/7/9, 11:13
client.py	247 Bytes	Python File	2019/7/9, 11:25
client2.py	1KB	Python File	2019/8/27, 14:37
darknet	1.86MB	文件	2019/7/9, 9:53
darknet53.conv.74	154.96MB	74 文件	2018/3/25, 14:01
first.py	380 Bytes	Python File	2019/8/29, 11:21
libdarknet.a	2.06MB	A 文件	2019/7/3, 0:28
libdarknet.so	1.78MB	SO 文件	2019/7/3, 0:28
LICENSE	515 Bytes	文件	2019/7/1, 10:02
LICENSE.fuck	474 Bytes	FUCK 文件	2019/7/1, 10:02
LICENSE.gen	6KB	GEN 文件	2019/7/1, 10:02
LICENSE.gpl	34KB	GPL 文件	2019/7/1, 10:02
LICENSE.meta	360 Bytes	META 文件	2019/7/1, 10:02
LICENSE.mit	1KB	MIT 文件	2019/7/1, 10:02
LICENSE.v1	461 Bytes	V1 文件	2019/7/1, 10:02
lyx.py	610 Bytes	Python File	2019/9/3, 20:24
Makefile	3KB	文件	2019/7/3, 0:27
nwpu2voc.py	5KB	Python File	2019/5/22, 23:03
c2.py	241 Bytes	Python File	2019/7/9, 11:13
client.py	247 Bytes	Python File	2019/7/9, 11:25
client2.py	1KB	Python File	2019/8/27, 14:37
darknet	1.86MB	文件	2019/7/9, 9:53
darknet53.conv.74	154.96MB	74 文件	2018/3/25, 14:01
first.py	380 Bytes	Python File	2019/8/29, 11:21
libdarknet.a	2.06MB	A 文件	2019/7/3, 0:28
libdarknet.so	1.78MB	SO 文件	2019/7/3, 0:28
LICENSE	515 Bytes	文件	2019/7/1, 10:02
LICENSE.fuck	474 Bytes	FUCK 文件	2019/7/1, 10:02
LICENSE.gen	6KB	GEN 文件	2019/7/1, 10:02
LICENSE.gpl	34KB	GPL 文件	2019/7/1, 10:02
LICENSE.meta	360 Bytes	META 文件	2019/7/1, 10:02
LICENSE.mit	1KB	MIT 文件	2019/7/1, 10:02
LICENSE.v1	461 Bytes	V1 文件	2019/7/1, 10:02
lyx.py	610 Bytes	Python File	2019/9/3, 20:24
Makefile	3KB	文件	2019/7/3, 0:27
nwpu2voc.py	5KB	Python File	2019/5/22, 23:03
predictions.jpg	12KB	JPG 文件	2019/8/27, 11:23
README.md	418 Bytes	Markdown...	2019/7/1, 10:02
server.py	418 Bytes	Python File	2019/8/29, 15:18
servernew.py	743 Bytes	Python File	2019/8/27, 14:31

图片集　　模型集　　代码集

实验方法—性能逐渐增强

阅读代码

从网页端控制自动驾驶转换为命令行控制自动驾驶，思考python文件调用顺序，加入延时以获得更好的性能

控制变量

一周半的长时候开始系统训练。控制不同的变量：图片分辨率，专项与混合训练、树莓派处理速度...

专项训练

在初期训练小车循迹时，小车在拐弯后很难回到原轨道直走，因此我们将拐弯过后回到原轨道，再继续直走的部分进行了多次训练。

迁移学习

由于视野内若未出现垃圾或垃圾回收处时小车都将沿固定轨道寻迹，通过迁移学习，转圈的数据可以被有效地运用于训练别的模型。

通信部分

此由于往树莓派上安装库非常困难，而且以树莓派的运行速度处理图像识别会非常缓慢，因此我们想利用GPU来代替树莓派进行图像识别，然后将结果返回给树莓派。但是服务器与树莓派不在一个子网下，无法利用socket通信，于是我们又借助了pc，通过xftp让pc上的文件实时更新，再利用socket通信将图像识别的结果回传给树莓派。

PART 6

成员分工

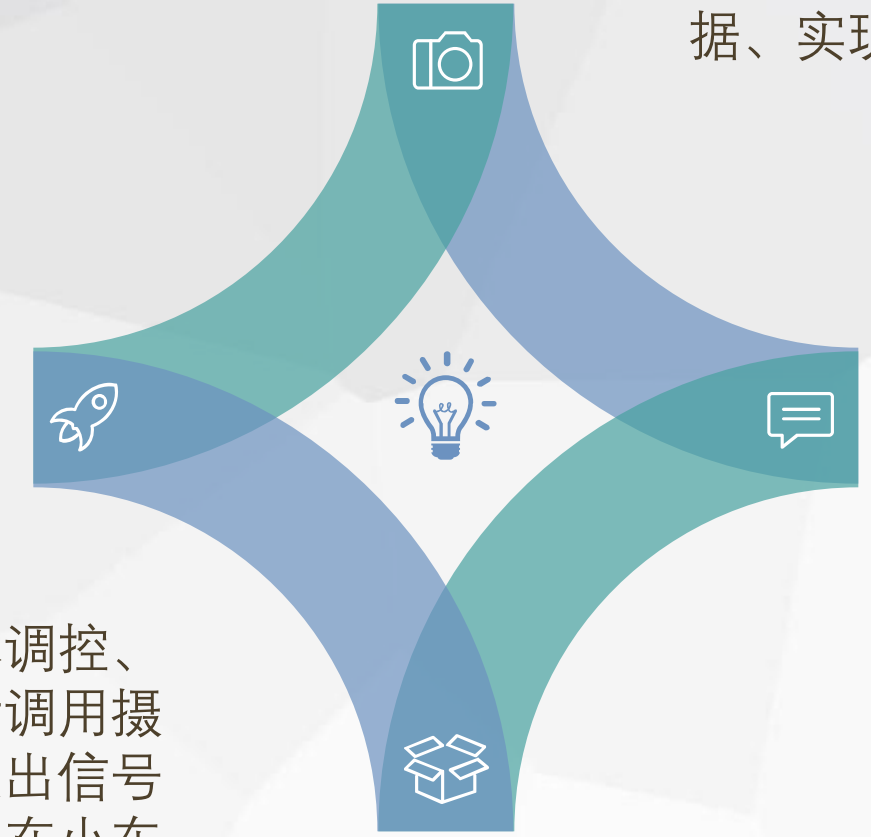
成员分工

杨奕冉、余康佳

主要负责donkeycar操作，环境配置、训练小车采集数据、实现基本自动驾驶功能。

赵行越

负责机械臂的控制。



罗如瑜

负责小车自动驾驶数据模型训练、垃圾分类数据模型训练。

负责全车各模块整体调控、包括小车靠近垃圾后调用摄像头拍摄并识别、发出信号让机械臂开始工作、在小车回到指定位置之后根据垃圾的类别放到不同的位置。

李韫瑄、许竞舟

THANKS!