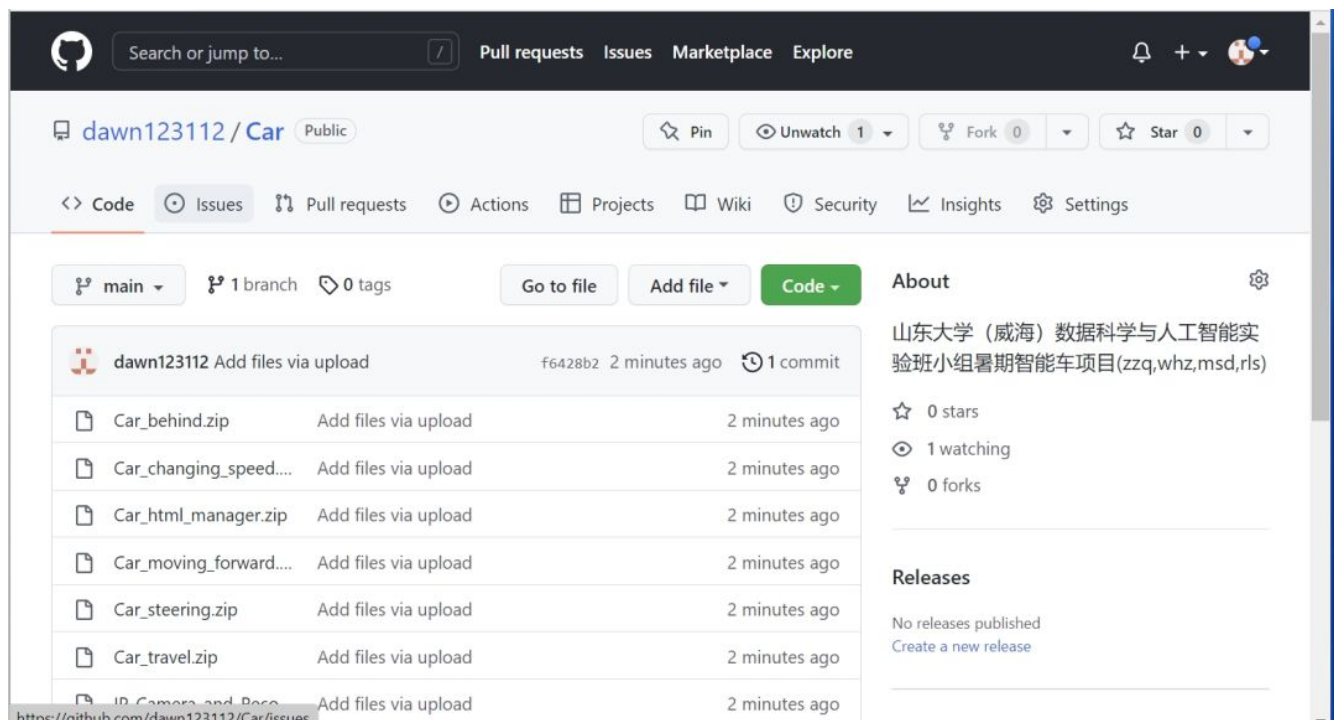


# 2022暑期大作业说明文档

## (特别鸣谢：人工智能领航计划)

代码已上传至Github(<https://github.com/dawn123112/Car>)



### 1.项目实现过程简介：

首先由于大一下学期的疫情影响，我组两名成员未能及时参与PCB线下课程，所以在项目进行之初，我们对PCB设计进行了组内讨论，帮助他们理清思路，并且尝试重新复刻了领航计划所提供的ESP32电路板。

在完成了前期的准备工作后，我组跟随领航计划中老师的思路和引导，先后实现了小车的前进后退以及变速、转向等功能，并应用领航计划所提供的资源建立了网站，实现了通过连接开发板信号以达到浏览器控制小车运动的目的。接着我们成功连接好了摄像头并尝试通过调整摄像头拍照频率采集地图数据并将其存储在SD卡中。至此，大作业项目的第一和第四部分基本实现。

关于大作业的第二部分通过python代码采用键盘修改参数控制小车移动，我们结合之前arduino控制的经验，通过Anaconda建立python环境（使用Anaconda的优点在于



其可以创建不同版本的python环境，更加便于调试），同时我们应用VScode将代码文件进行整合，在烧录完成开发板及摄像头代码后，通过改变电脑端的网络连接，使其于开发板处于同一信号下，通过修改python代码中的参数实现了小车的自由移动及摄像头采集地图数据。

在第三部分的实现上，首先通过Anaconda构建一个python环境，然后通过百度EasyData对地图数据进行标注并导出，之后对标注结果进行mask处理，采用UNet道路分割模型进行训练并预测，这里需要注意采用中科曙光的GPU环境来保证训练成功。之后调节好测速仪和测角仪使其正常工作，利用路径跟踪以及路径规划等原理结合摄像头画面实现小车的独立行走。

在收尾环节，我们再次试验了第四部分中浏览器的可操作性，并总结了项目实操过程中遇到的问题以备在下学期的学习中加以改进，同时我组也充分发挥了合作精神，在其他小组遇到问题时给予了一定支持，共同讨论遇到的一些问题，在项目推进过程中，我组和大二范学长小组在模型训练等方面进行了交流，学习到了很多宝贵的经验并加以运用，最后，我们小组结合项目本身和校区暑期社会实践活动进行了理论层面的前瞻应用分析，比如在疫情期间通过小车无人驾驶实现送餐等问题，使项目的学习意义得到了充分发挥。

## 2.重点部分理论知识说明（个人理解）：

### （1）电机：

在本项目中，电机作为小车运动的核心器件之一，通过齿轮结构控制小车的前进、后退以及变速等功能，通过PWM信号进行控制，在这里占空比为高电平/周期，产生不同的模拟信号，该信号是由5V和0V所占的比重而生成的。产生PWM信号的方法：通过LEDC，先要有一个频率，然后需要设置信号通道，我们有16个信号通道，执行计数操作，接着需要一个分辨率来控制精度问题。在实际设置过程中，需要把分辨率和频率全部设置给信号通道，在理论上，占空比越大，高电平输出越多。

在电机的实际控制中，通过电机内部两个引脚的高低电平控制，来实现小车的前进与后退（停止时需要同时输出HIGH，若都为LOW则不确定状态），在电机中，PWM信号对应电机转速（半速占空比取128，全速取255），电机通过电池与开发板进行连接。

电机运动代码：

```
const int a = 26;
```

```
const int b = 27;
```



```

const int freq = 2000;
const int resolution = 8;
const int channelA = 0;
const int channelB = 1;
const int dutyCycle = 255;
void setup() {
  ledcSetup(channelA,freq,resolution);
  ledcAttachPin(a,channelA);
  ledcSetup(channelB,freq,resolution);
  ledcAttachPin(b,channelB);
  pinMode(a,OUTPUT);
  digitalWrite(a,LOW);
}
void loop() {
  ledcWrite(channelA,dutyCycle);
  ledcWrite(channelB,0);
}

```

## (2) 舵机：

舵机在小车中位于小车前端，又被称作伺服马达，通过齿轮控制小车转向。PWM信号在这里用来控制舵机的角度，频率在这里设定为固定值50HZ，通过控制占空比来调整精确角度，舵机通过开发板引脚连接。

舵机调试代码：

```

const int a = 13;
const int freq = 50;
const int resolution = 8;
const int c = 0;

```



```

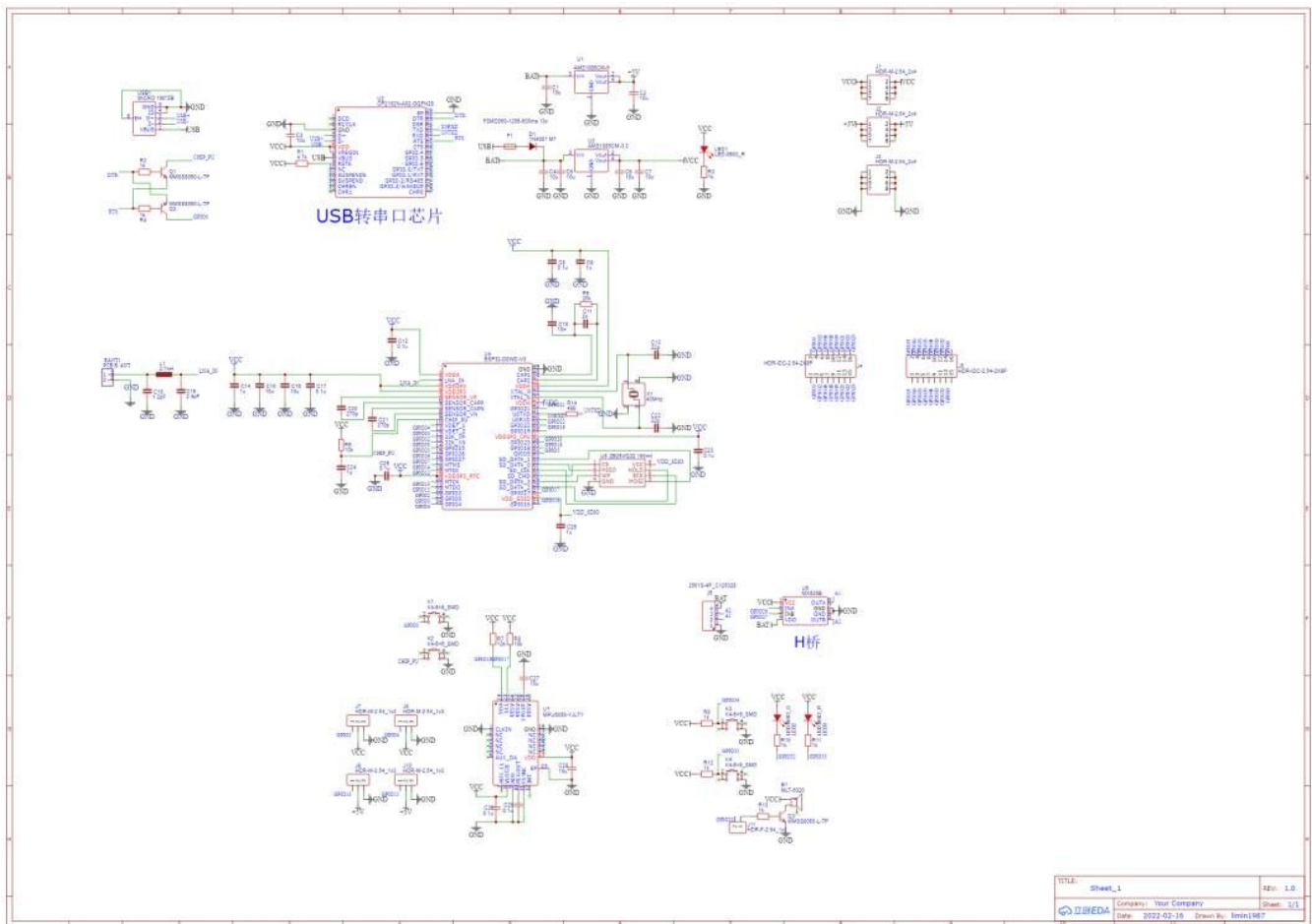
const int d = 32;

void setup() {
  ledcSetup(c,freq,resolution);
  ledcAttachPin(a,c);
}

void loop() {
  for(int i = 7;i <= 32;i
  ledcWrite(c,i);
  delay(500);
}
}

```

### (3) ESP32开发板原理图：



#### (4) 小车前进及转向代码合并:

```
const int servo = 15;
const int motorA = 26;
const int motorB = 27;
const int freq = 50;
const int resolution = 8;
const int servo channel = 0;
const int motorA channel = 1;
const int motorB channel = 2;

void setup() {
  ledcSetup(servo channel, freq, resolution);
  ledcAttachPin(servo, servo channel);
  ledcSetup(motorA channel, freq, resolution);
  ledcAttachPin(motorA, motorA channel);
  ledcSetup(motorB channel, freq, resolution);
  ledcAttachPin(motorB, motorB channel);
  ledcWrite(servo channel, 20);
  ledcWrite(motorA channel, 0);
  ledcWrite(motorB channel, 0);
}

void loop() {
  while(1){
    ledcWrite(motorA channel, 128);
    delay(10500);
    ledcWrite(servo channel, 7);
    delay(4000);
```

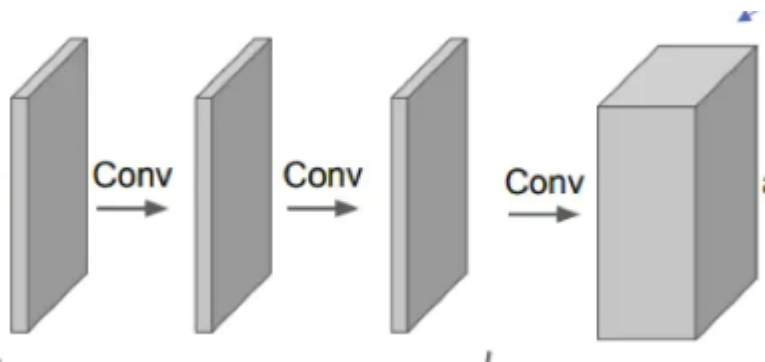


```
ledcWrite(servo channel,20);  
ledcWrite(motorA channel,128);  
delay(4500);  
ledcWrite(servo channel,7);  
delay(4000);  
ledcWrite(servo channel,20);  
ledcWrite(motorA channel,128);  
delay(6500);  
ledcWrite(servo channel,7);  
delay(4000);  
ledcWrite(servo channel,20);  
ledcWrite(motorA channel,128);  
delay(4500);  
ledcWrite(servo channel,7);  
delay(4000);  
ledcWrite(servo channel,20);  
delay(500);  
}  
}
```

### (5) UNet道路分割模型:

UNet网络是一种图像语义分割网络，图像语义分割网络让计算机根据图像的语义来进行分割。简单的来说，整个网络分为两个部分，左边部分负责特征提取，随着网络层加深，网络的channel逐渐变大，"图片"逐渐变小。右边的网络负责特征的还原，整个网络实际上就是一个编码-解码器。



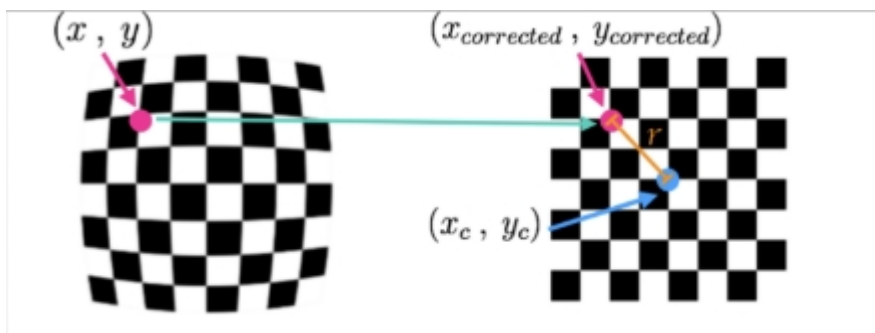


### (6) 鸟瞰图生成:

自动驾驶需要准确表示车辆周围的环境，环境包括道路布局和车道结构等静态元素，以及其他汽车、行人和其他类型的道路使用者等动态元素，静态元素可以通过包含车道级别信息的高清地图捕获，通过在BEV上做语义分割和检测，在线生成地图，轨迹预测和规划通常在这种自上而下的视图（即鸟瞰图）中完成，因为高度信息不太重要，而自动驾驶汽车所需的大部分信息可以通过透视变换方便地用 BEV 表示。

### (7) 去畸变操作:

镜头的畸变是透镜失真造成的，主要分为三类：枕形畸变、桶形畸变、线性畸变。想要解决畸变的问题需要用到相机的内参和外参，相机内参和相机外参一共有至少8个参数，而要想消除相机的畸变，就要靠相机标定来求解这8个未知参数。



### (8) 路径规划算法:

我们可以采用一种贪心模式来解决有向图中单个节点到另一节点的最短路径问题，其主要特点是每次迭代时选择的下一个节点是当前节点最近的子节点，也就是说每一次迭代行进的路程是最短的。而为了保证最终搜寻到的路径最短，在每一次迭代过程中，都要对起始节点到所有遍历到的点之间的最短路径进行更新，这一点与我们在上学期离散数学中所学到的知识不谋而合。

## 3.实操过程说明及展示:



安装Arduino并选好开发板型号以及端口；



烧录程序，点击上传；



下载Anaconda，在终端输入"conda"确认安装完成（需要在环境变量中配置路径）；



```

C:\Users\赵泽奇>conda
usage: conda-script.py [-h] [-V] command ...

conda is a tool for managing and deploying applications, environments and packages.

Options:
positional arguments:
  command
  clean                Remove unused packages and caches.
  compare              Compare packages between conda environments.
  config               Modify configuration values in .condarc. This is modeled after the git config command. Writes to the
                      user .condarc file (C:\Users\赵泽奇\.condarc) by default.
  create               Create a new conda environment from a list of specified packages.
  help                Displays a list of available conda commands and their help strings.
  info                 Display information about current conda install.
  init                 Initialize conda for shell interaction. [Experimental]
  install              Installs a list of packages into a specified conda environment.
  list                 List linked packages in a conda environment.
  package              Low-level conda package utility. (EXPERIMENTAL)
  remove               Remove a list of packages from a specified conda environment.
  uninstall            Alias for conda remove.
  run                  Run an executable in a conda environment.
  search               Search for packages and display associated information. The input is a MatchSpec, a query language
                      for conda packages. See examples below.
  update               Updates conda packages to the latest compatible version.
  upgrade              Alias for conda update.

optional arguments:
  -h, --help            Show this help message and exit.
  -V, --version          Show the conda version number and exit.

conda commands available from other packages:
  build
  content-trust
  convert
  debug


```

创建torch虚拟环境；

```

C:\Users\赵泽奇>conda create -n torch_1_12 python=3.7

```



```

done
#
# To activate this environment, use
#
#   $ conda activate torch_1_12
#
# To deactivate an active environment, use
#
#   $ conda deactivate
#
C:\Users\赵泽奇>

```

启动环境（conda activate）；



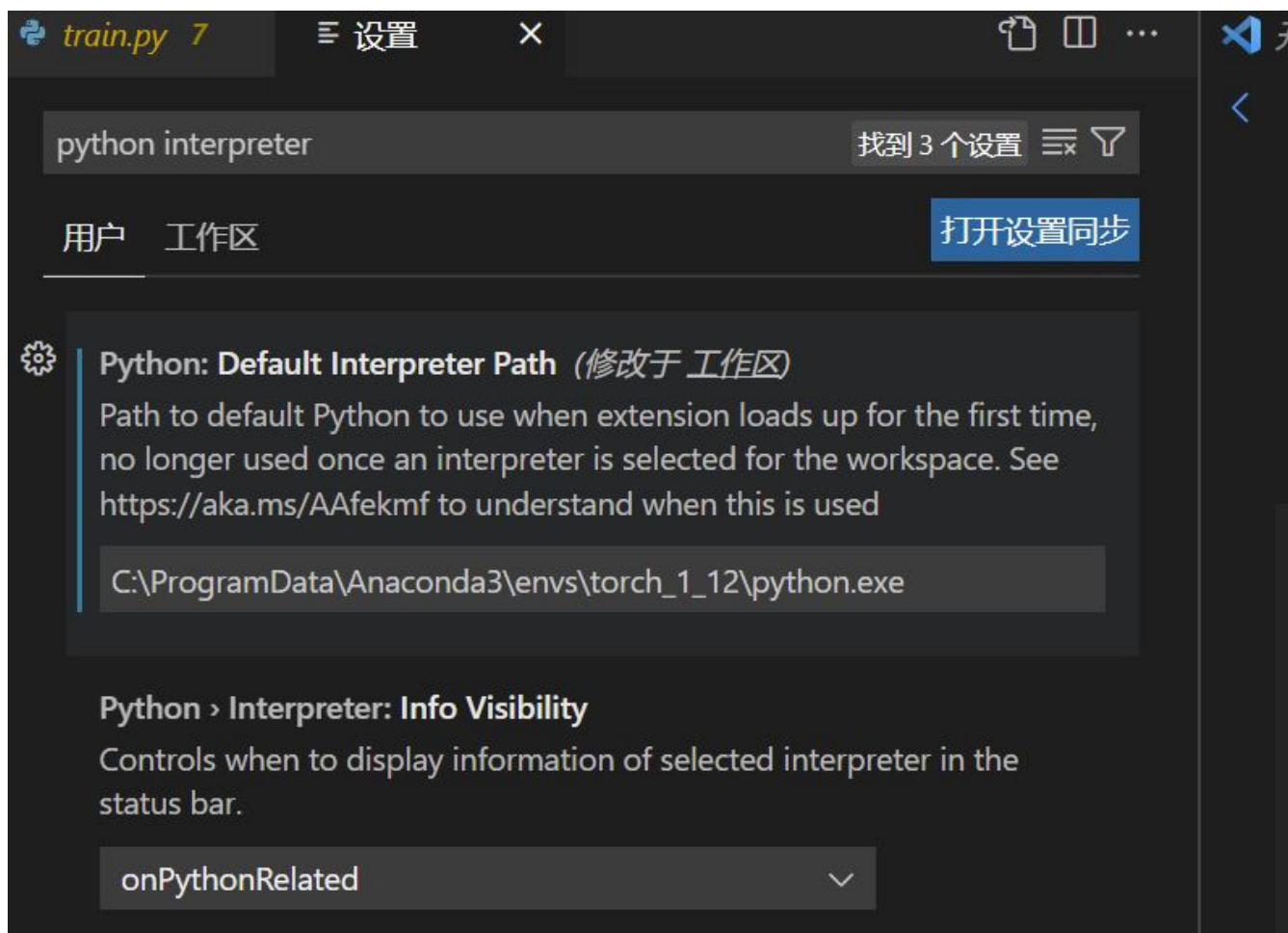
```
C:\Users\赵泽奇>conda activate torch_1_12  
(torch_1_12) C:\Users\赵泽奇>
```

安装所需python包 (pip install) ;

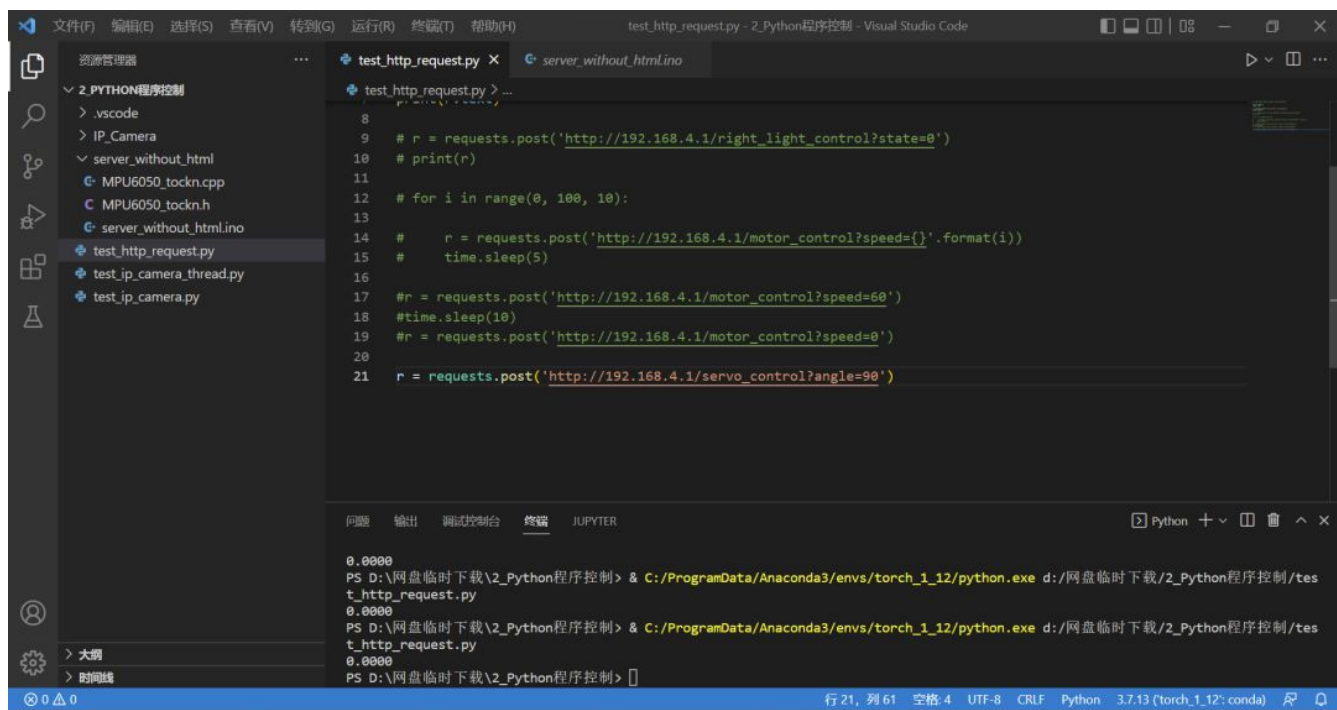
```
Dev>>> quit()  
(torch_1_12) C:\Users\赵泽奇>pip install torch-vision  
Collecting torch-vision  
  Downloading torch_vision-0.1.6.dev0-py2.py3-none-any.whl (23 kB)  
Installing collected packages: torch-vision  
Successfully installed torch-vision-0.1.6.dev0  
(torch_1_12) C:\Users\赵泽奇>where python  
C:\ProgramData\Anaconda3\envs\torch_1_12\python.exe  
(torch_1_12) C:\Users\赵泽奇>pip install tqdm torchsummary  
Collecting tqdm  
  Downloading tqdm-4.64.0-py2.py3-none-any.whl (78 kB)  
78.4/78.4 kB 545.7 kB/s eta 0:00:00  
Collecting torchsummary  
  Downloading torchsummary-1.5.1-py3-none-any.whl (2.8 kB)  
Collecting colorama  
  Downloading colorama-0.4.5-py2.py3-none-any.whl (16 kB)  
Installing collected packages: torchsummary, colorama, tqdm  
Successfully installed colorama-0.4.5 torchsummary-1.5.1 tqdm-4.64.0  
(torch_1_12) C:\Users\赵泽奇>pip install numpy  
Collecting numpy  
  Downloading numpy-1.21.6-cp37m-win_amd64.whl (14.0 MB)  
14.0/14.0 MB 61.1 kB/s eta 0:00:00  
Installing collected packages: numpy  
Successfully installed numpy-1.21.6  
(torch_1_12) C:\Users\赵泽奇>
```

部署VScode中的python解释器;





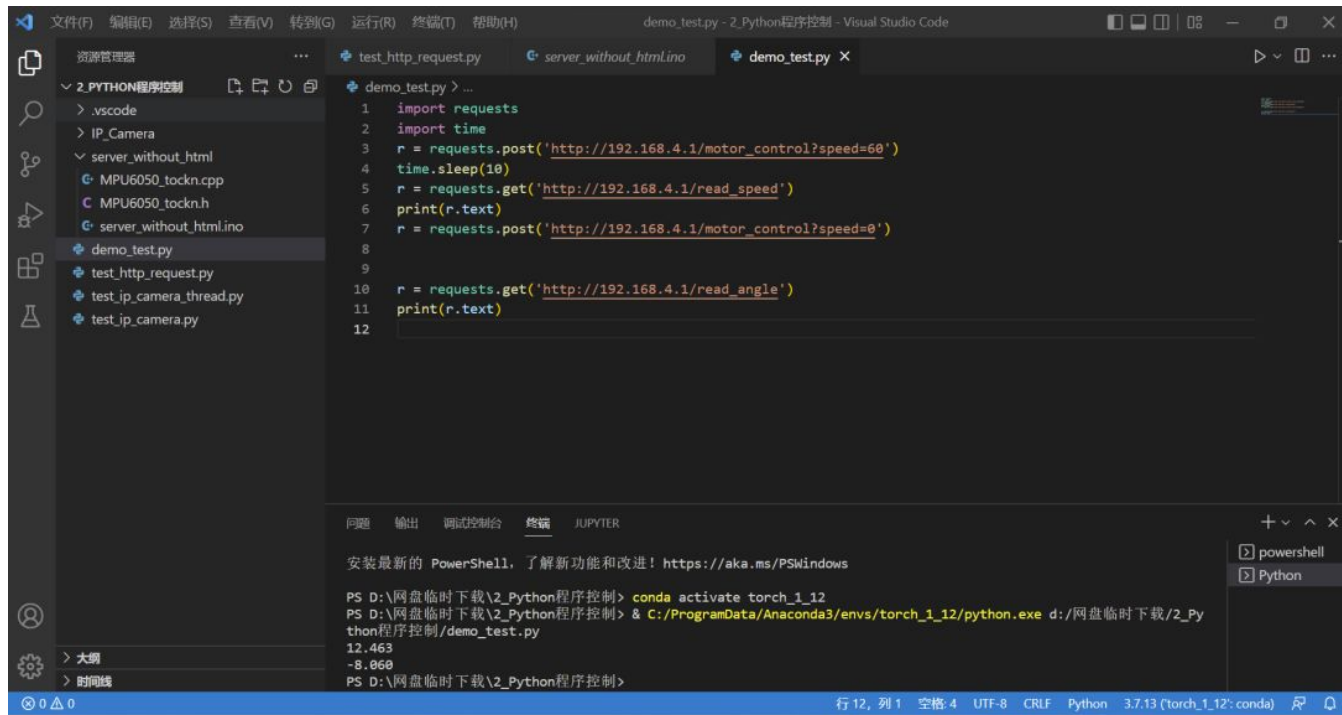
调试python程序控制小车移动;



创作于 Effie (试用版)

```
demo_test.py > ...
1  import requests
2  import time
3  r = requests.post('http://192.168.4.1/motor_control?speed=60')
4  time.sleep(10)
5  r = requests.get('http://192.168.4.1/read_speed')
6  print(r.text)
7
8
9  r = requests.post('http://192.168.4.1/motor_control?speed=0')
10 r = requests.get('http://192.168.4.1/read_angle')
11 print(r.text)
12
```

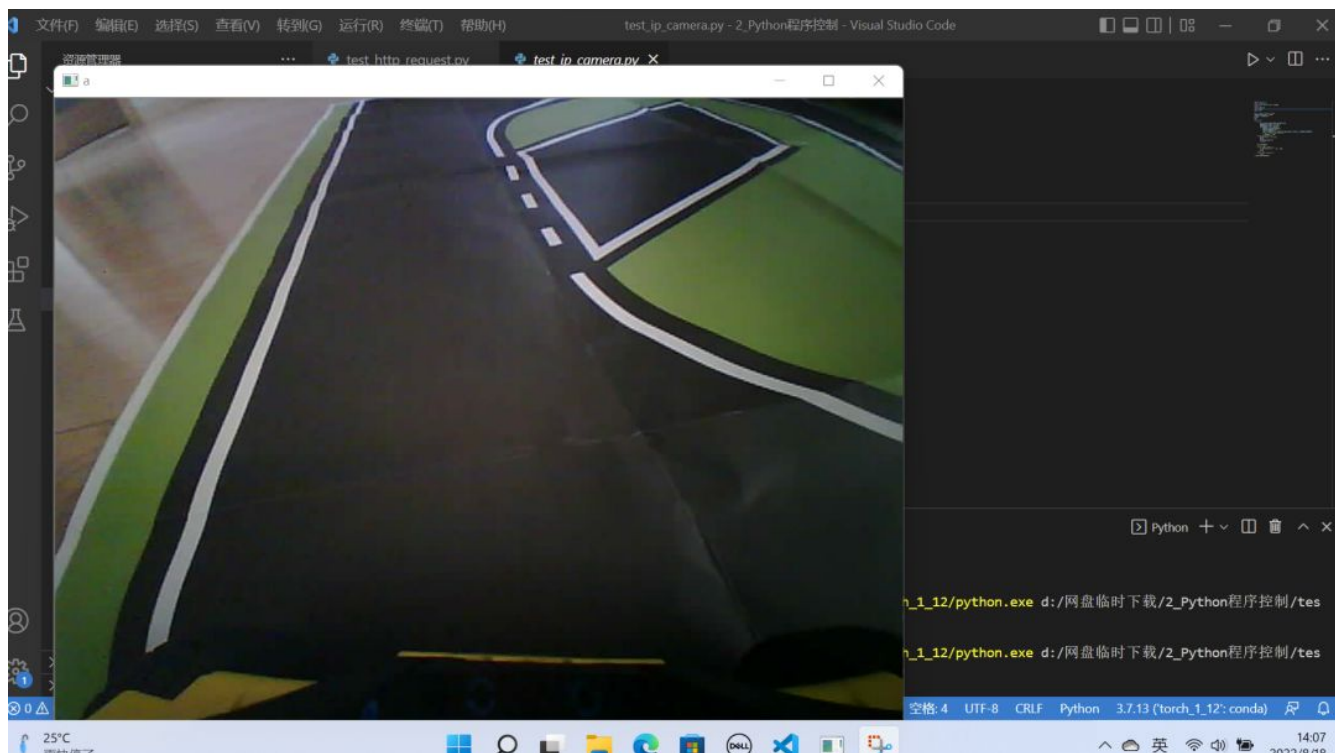
测出速度及角度；



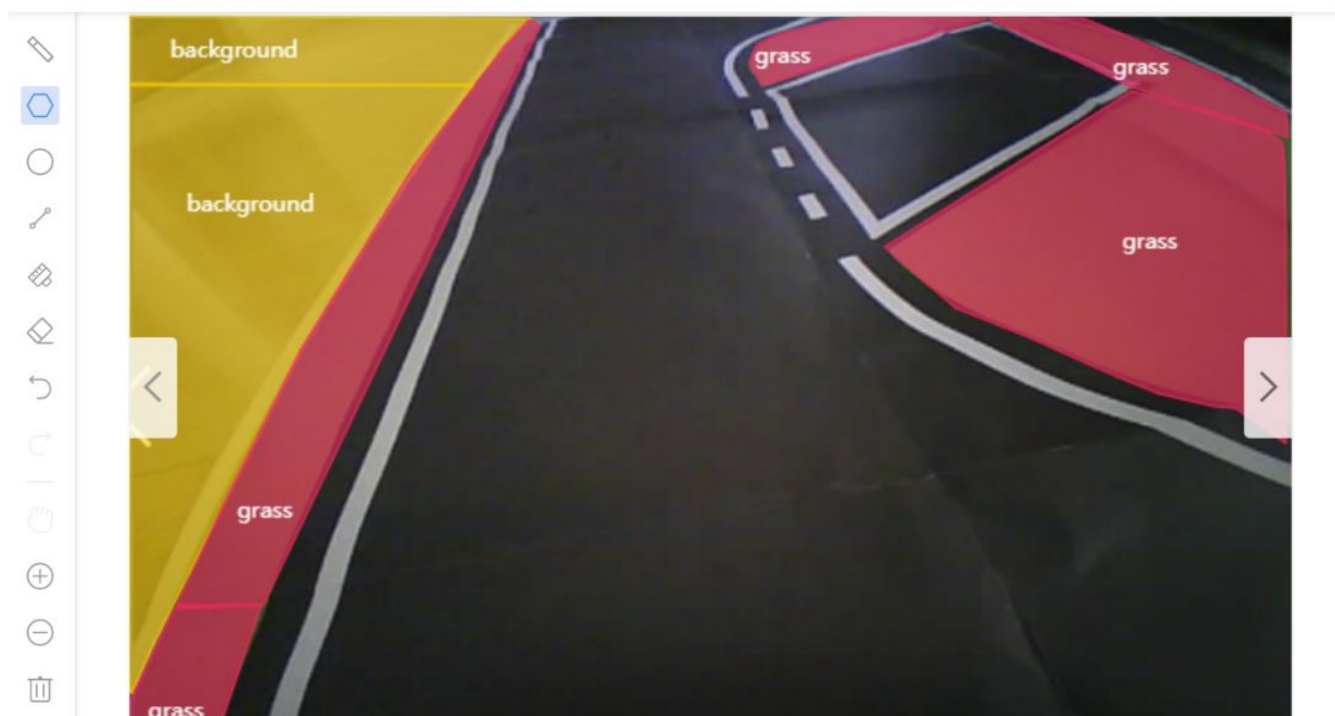
对采集的地图照片进行标注；







我的数据总览 > 【图片】智能小车作业/V1/查看与标注 > 标注



对导出的图片进行mask处理;



创作于 Effie (试用版)

```
mask_rgb_to_gray.py > ...
1 import cv2
2 import os
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 dataset_dir = './0719_labeled/'
7 image_dir = 'JPEGImages/'
8 rgb_mask_dir = 'SegmentationClass/'
9 gray_mask_dir = 'Masks/'
10
11 mask_list = os.listdir(dataset_dir + rgb_mask_dir)
12 for i in range(len(mask_list)):
13
14     rgb_mask = cv2.imread(dataset_dir + rgb_mask_dir + mask_list[i])
15     grass_layer = rgb_mask[:, :, 1] / 128
16     background_layer = rgb_mask[:, :, 2] / 128
17     gray_mask = grass_layer + background_layer * 2
18     # plt.imshow(gray_mask, 'gray')
19     # plt.show()
20
21     cv2.imwrite(dataset_dir + gray_mask_dir + mask_list[i], gray_mask)
```

应用sugon环境 (ac.sugon.com) ;



在服务器内申请GPU资源;



```
区域：华东一区【昆山】 计算用户：zhaozeqi
terminal-1 +
# 1.Use 'module avail' to search system available software.
# 2.Job templates are located in $HOME/slurm_template.
# 3.Login node is NOT allowed to run programs!!
#####
[zhaozeqi@login07 ~]$ salloc -p kshdtest
salloc: Pending job allocation 23054911
salloc: job 23054911 queued and waiting for resources
salloc: job 23054911 has been allocated resources
salloc: Granted job allocation 23054911
salloc: Waiting for resource configuration
salloc: Nodes e10r4n14 are ready for job
[zhaozeqi@login07 ~]$ salloc -p kshdtest -N 1 --gres=dcu:4 --cpus-per-task=6
salloc: Pending job allocation 23055145
salloc: job 23055145 queued and waiting for resources
salloc: job 23055145 has been allocated resources
salloc: Granted job allocation 23055145
salloc: Waiting for resource configuration
salloc: Nodes j17r2n00 are ready for job
[zhaozeqi@login07 ~]$ squeue
      JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
      23055145  kshdtest    bash  zhaozeqi  R        0:43      1 j17r2n00
      23054911  kshdtest    bash  zhaozeqi  R        6:33      1 e10r4n14
[zhaozeqi@login07 ~]$
```

进入环境 (ssh) ；

```
salloc: Nodes j17r2n00 are ready for job
[zhaozeqi@login07 ~]$ squeue
      JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
      23055145  kshdtest    bash  zhaozeqi  R        0:43      1 j17r2n00
      23054911  kshdtest    bash  zhaozeqi  R        6:33      1 e10r4n14
[zhaozeqi@login07 ~]$ ssh j17r2n00
Warning: Permanently added 'j17r2n00,10.10.17.21' (ECDSA) to the list of known hosts.
[zhaozeqi@j17r2n00 ~]$
```

选中文本复制, 点击 [这里](#) 或ctrl+shift+v粘贴 查看调度器常用命令, 点击 [这里](#) [下载秘钥](#) Putty [打开客户端](#) 发送组合

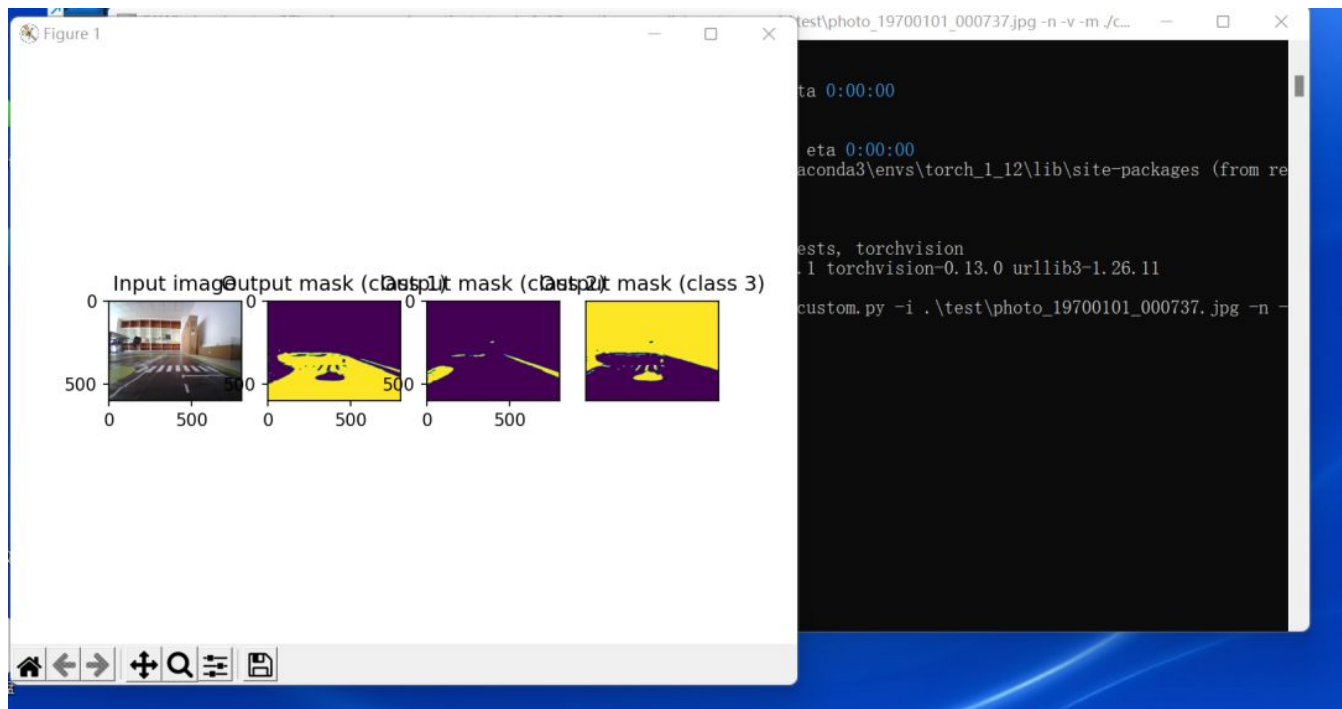
训练模型；

```
(pytorch-1.9) [zhaozeqi@j17r2n00 unet_torch]$ ./submit.sh
INFO: Using device cuda
INFO: Network:
      3 input channels
      3 output channels (classes)
      Transposed conv upscaling
../datasets/0719_labeled/JPEGImages ../datasets/0719_labeled/Masks
INFO: Creating dataset with 233 examples
Epoch 1/30: 0%| 0/210 [00:00<?, ?img/s]
/public/home/zhaozeqi/anaconda3/envs/pytorch-1.9/lib/python3.6/site-packages/torch/nn/functional.py:718: UserWarning: Named tensors a
nd all their associated APIs are an experimental feature and subject to change. Please do not use them for anything important until t
hey are released as stable. (Triggered internally at /pytorch/c10/core/TensorImpl.h:1156.)
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ceil_mode)
Epoch 1/30: 100%| 210/210 [01:08<00:00, 3.07img/s, loss (batch)=1.5]
INFO: Checkpoint 1 saved!
Epoch 2/30: 100%| 210/210 [00:39<00:00, 5.32img/s, loss (batch)=1.45]
INFO: Checkpoint 2 saved!
Epoch 3/30: 100%| 210/210 [00:39<00:00, 5.33img/s, loss (batch)=1.54]
INFO: Checkpoint 3 saved!
Epoch 4/30: 100%| 210/210 [00:39<00:00, 5.32img/s, loss (batch)=1.49]
INFO: Checkpoint 4 saved!
Epoch 5/30: 100%| 210/210 [00:39<00:00, 5.32img/s, loss (batch)=1.52]
INFO: Checkpoint 5 saved!
```



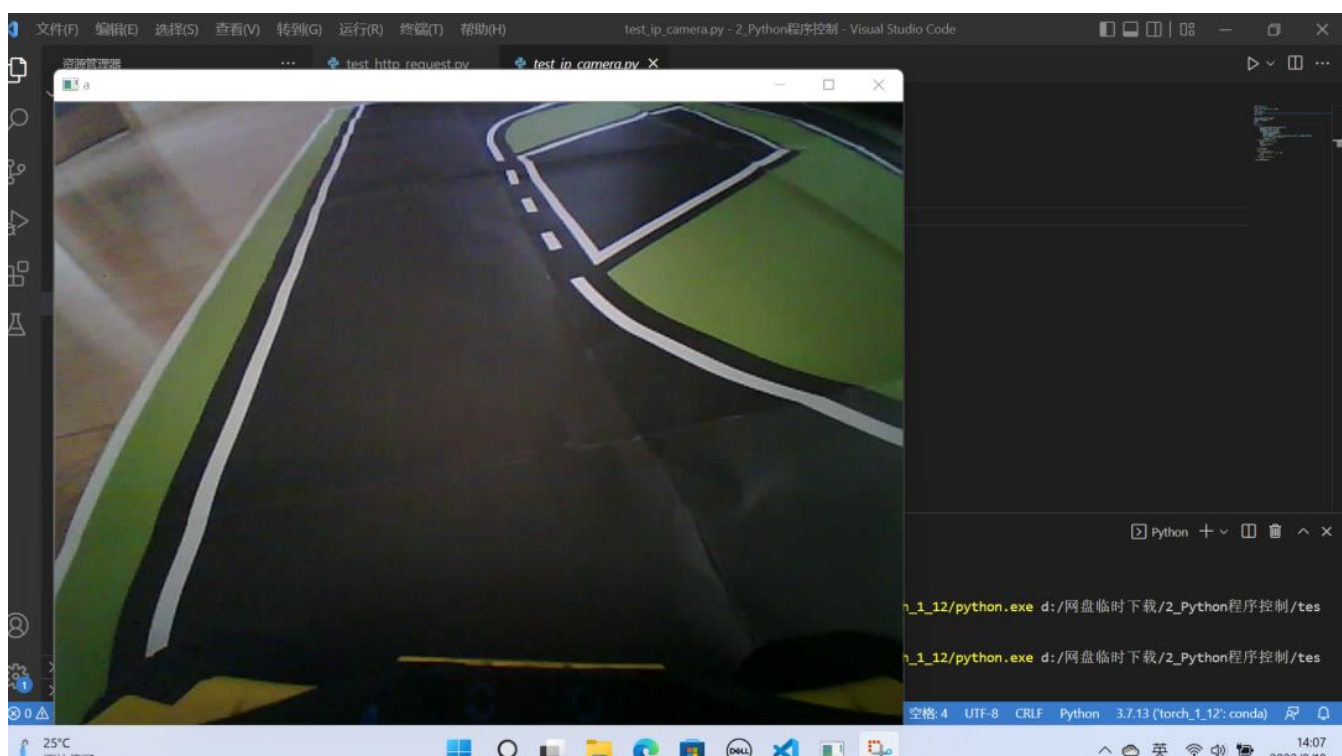
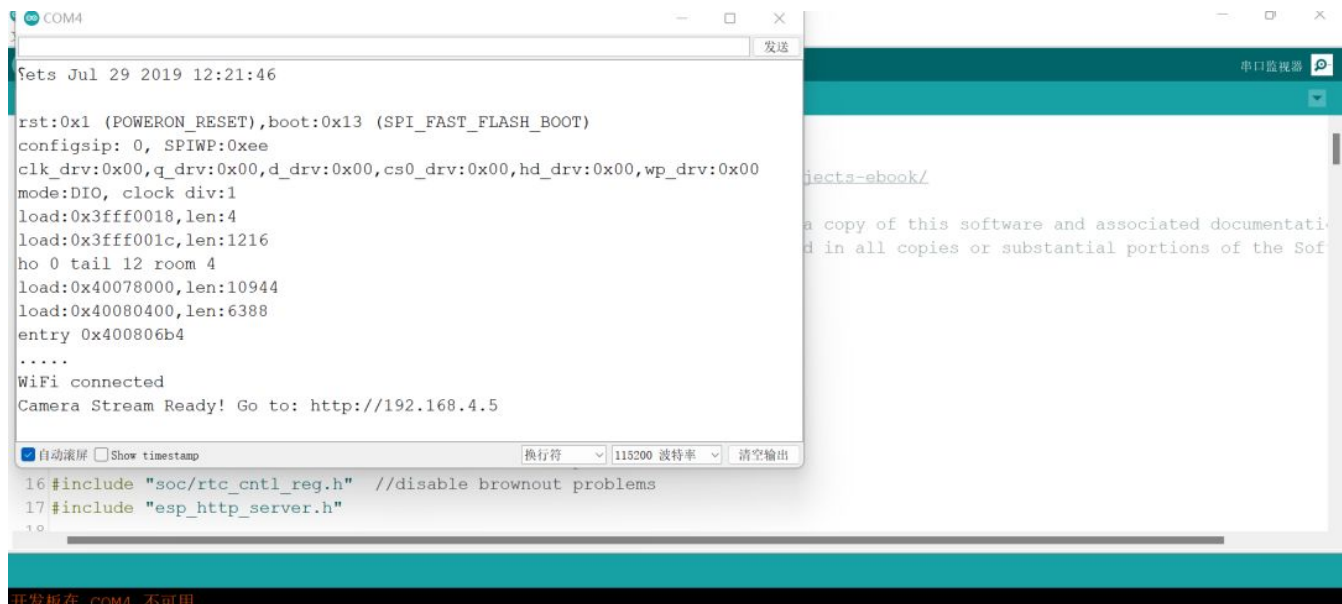
```
区域：华东一区【昆山】 计算用户：zhaozeqi
terminal-1
Epoch 21/30: 100% | 210/210 [00:39<00:00, 5.32img/s, loss (batch)=1.5]
INFO: Checkpoint 21 saved!
Epoch 22/30: 100% | 210/210 [00:39<00:00, 5.31img/s, loss (batch)=1.45]
INFO: Checkpoint 22 saved!
Epoch 23/30: 100% | 210/210 [00:39<00:00, 5.32img/s, loss (batch)=1.57]
INFO: Checkpoint 23 saved!
Epoch 24/30: 100% | 210/210 [00:39<00:00, 5.29img/s, loss (batch)=1.47]
INFO: Checkpoint 24 saved!
Epoch 25/30: 100% | 210/210 [00:39<00:00, 5.28img/s, loss (batch)=1.51]
INFO: Checkpoint 25 saved!
Epoch 26/30: 100% | 210/210 [00:39<00:00, 5.32img/s, loss (batch)=1.51]
INFO: Checkpoint 26 saved!
Epoch 27/30: 100% | 210/210 [00:39<00:00, 5.28img/s, loss (batch)=1.48]
INFO: Checkpoint 27 saved!
Epoch 28/30: 100% | 210/210 [00:39<00:00, 5.31img/s, loss (batch)=1.44]
INFO: Checkpoint 28 saved!
Epoch 29/30: 15% | 32/210 [00:05<00:23, 7.47img/s, loss (batch)=1.49C
| 0/1 [00:00<?, ?batch/s]
Connection to j17r2n00 closed by remote host.
Connection to j17r2n00 closed.
(pytorch-1.9) [zhaozeqi@login07 ~]$
(pytorch-1.9) [zhaozeqi@login07 ~]$ ssh j17r2n00
Access denied by pam_slurm_adapt: you have no active jobs on this node
Authentication failed.
(pytorch-1.9) [zhaozeqi@login07 ~]$
```

查看预测结果；



调试摄像头；



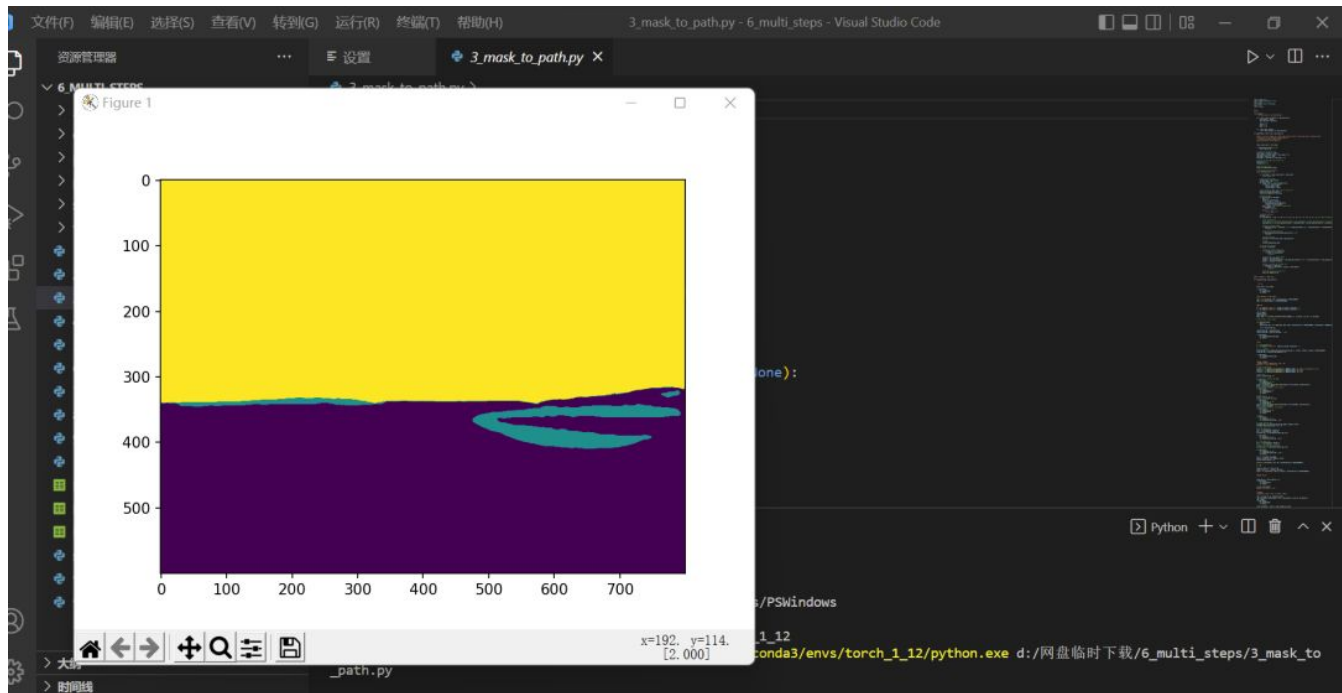


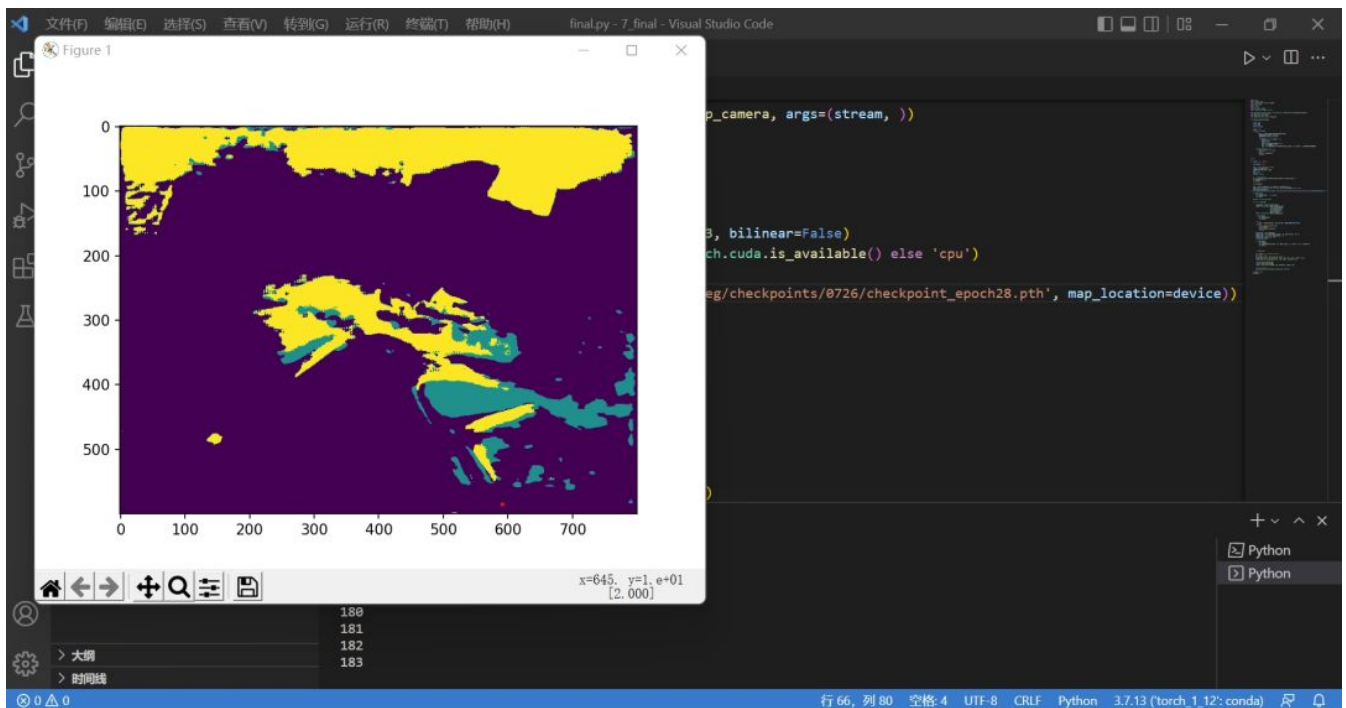
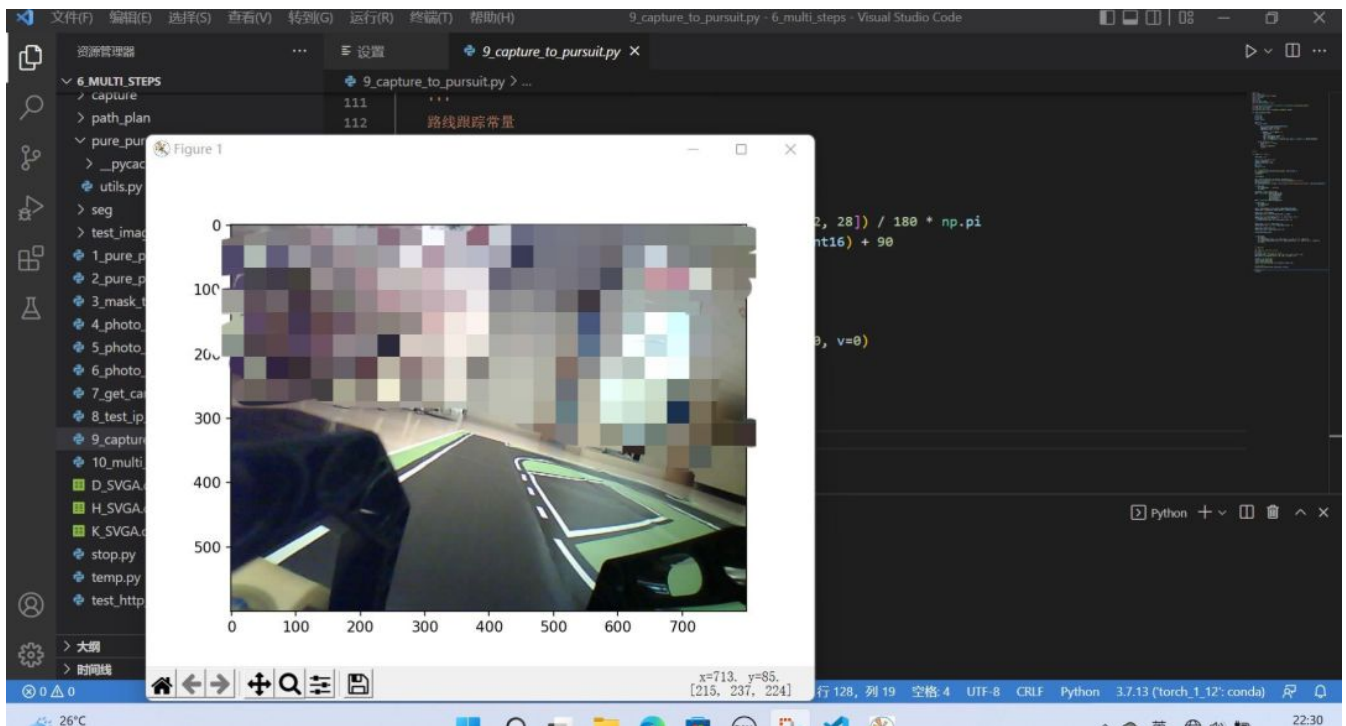
The screenshot shows a Visual Studio Code editor with a Python file named `test_ip_camera.py`. The script imports `cv2`, `numpy`, `urllib.request`, `os`, `datetime`, `time`, and `sys`. It defines a URL `url = 'http://192.168.4.5:80/'`, a buffer size `CAMERA_BUFFER_SIZE = 4096`, and a stream `stream = urlopen(url)`. A loop reads data from the stream, finds JPEG headers, and saves the image. The output window shows a `urllib.error.URLError` with the message: `<urlopen error [WinError 10051] 向一个无法连接的网络尝试了一个套接字操作。>`

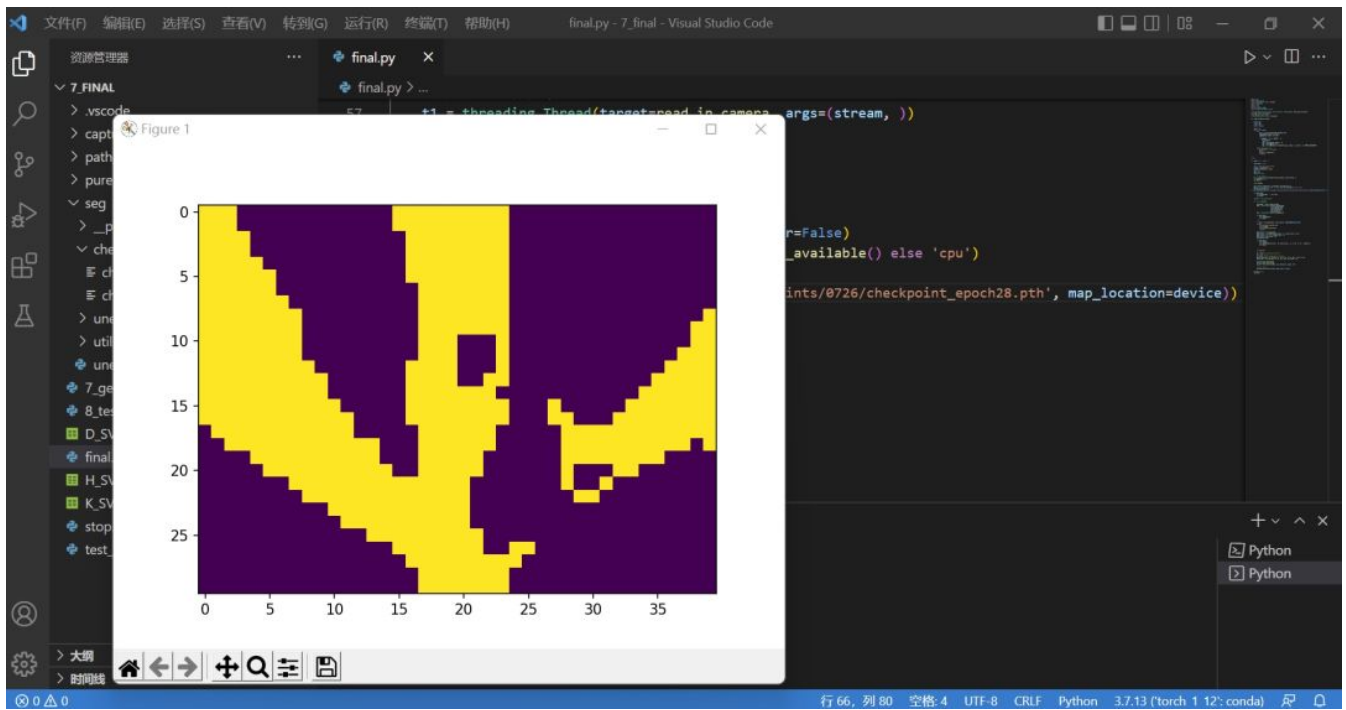
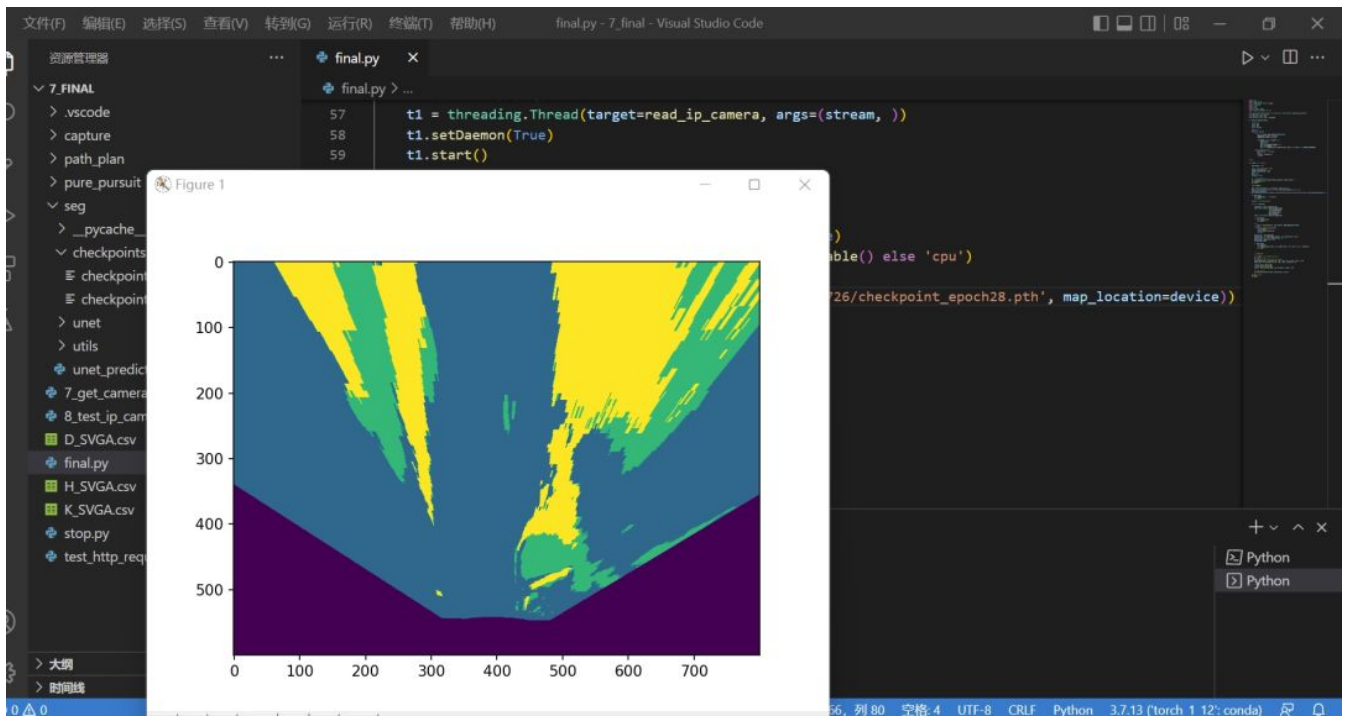
```
1 import cv2 as cv
2 import numpy as np
3 from urllib.request import urlopen
4 import os
5 import datetime
6 import time
7 import sys
8
9 url = 'http://192.168.4.5:80/'
10 CAMERA_BUFFER_SIZE = 4096
11 stream = urlopen(url)
12 bts=b''
13 i=0
14 while True:
15     try:
16         bts+=stream.read(CAMERA_BUFFER_SIZE)
17         jpghead=bts.find(b'\xff\xd8')
18         jpgend=bts.find(b'\xff\xd9')
19         if jpghead>-1 and jpgend>-1:
20             jpg=bts[jpghead:jpgend+2]
```

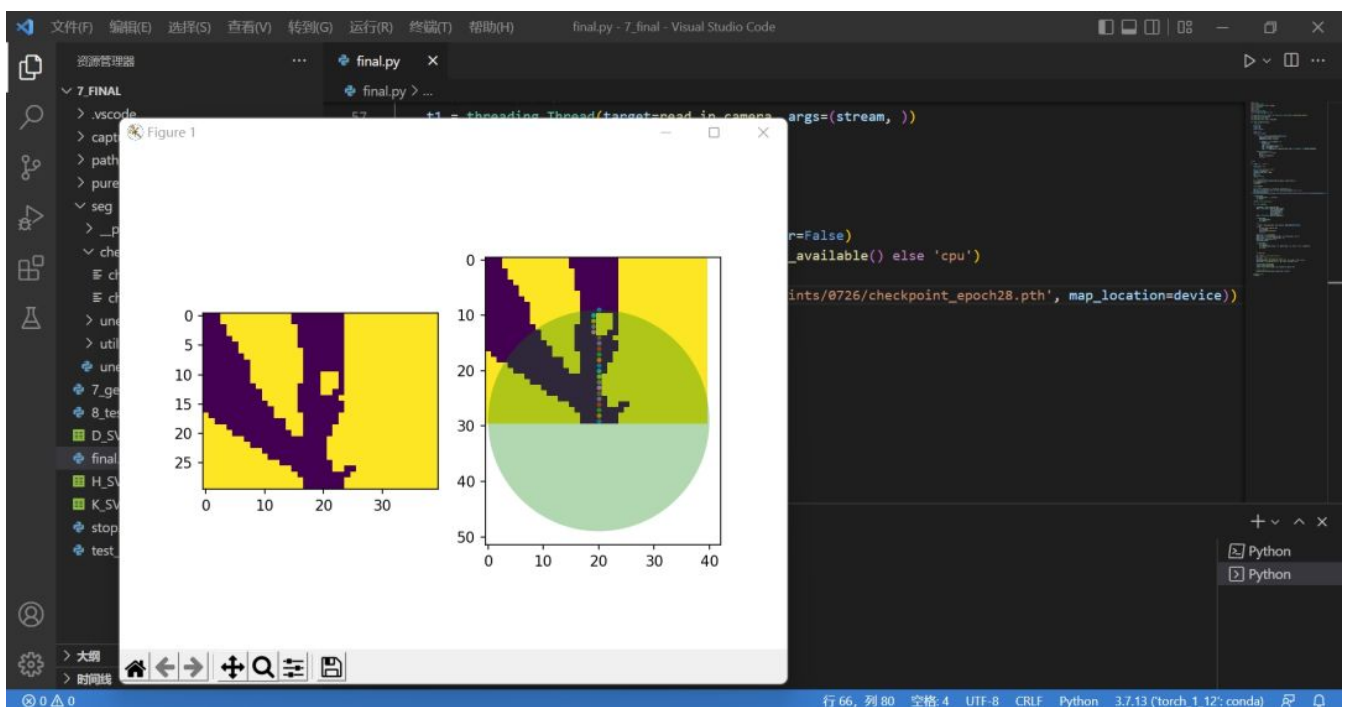
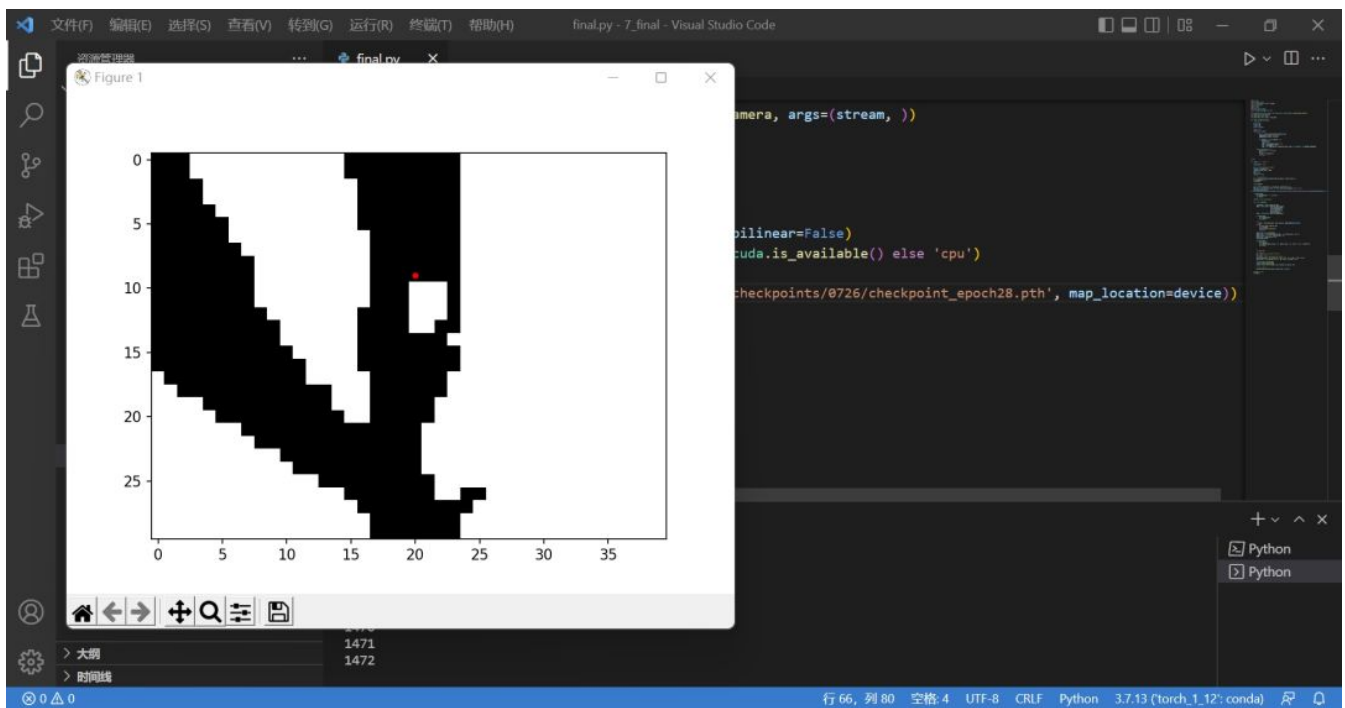
File "C:\ProgramData\Anaconda3\envs\torch\_1\_12\lib\urllib\request.py", line 503, in \_call\_chain  
result = func(\*args)  
File "C:\ProgramData\Anaconda3\envs\torch\_1\_12\lib\urllib\request.py", line 1378, in http\_open  
return self.do\_open(http.client.HTTPConnection, req)  
File "C:\ProgramData\Anaconda3\envs\torch\_1\_12\lib\urllib\request.py", line 1352, in do\_open  
raise URLError(err)  
urllib.error.URLError: <urlopen error [WinError 10051] 向一个无法连接的网络尝试了一个套接字操作。>  
PS D:\网盘临时下载\2\_Python程序控制>

通过摄像头实时数据进行路径规划；

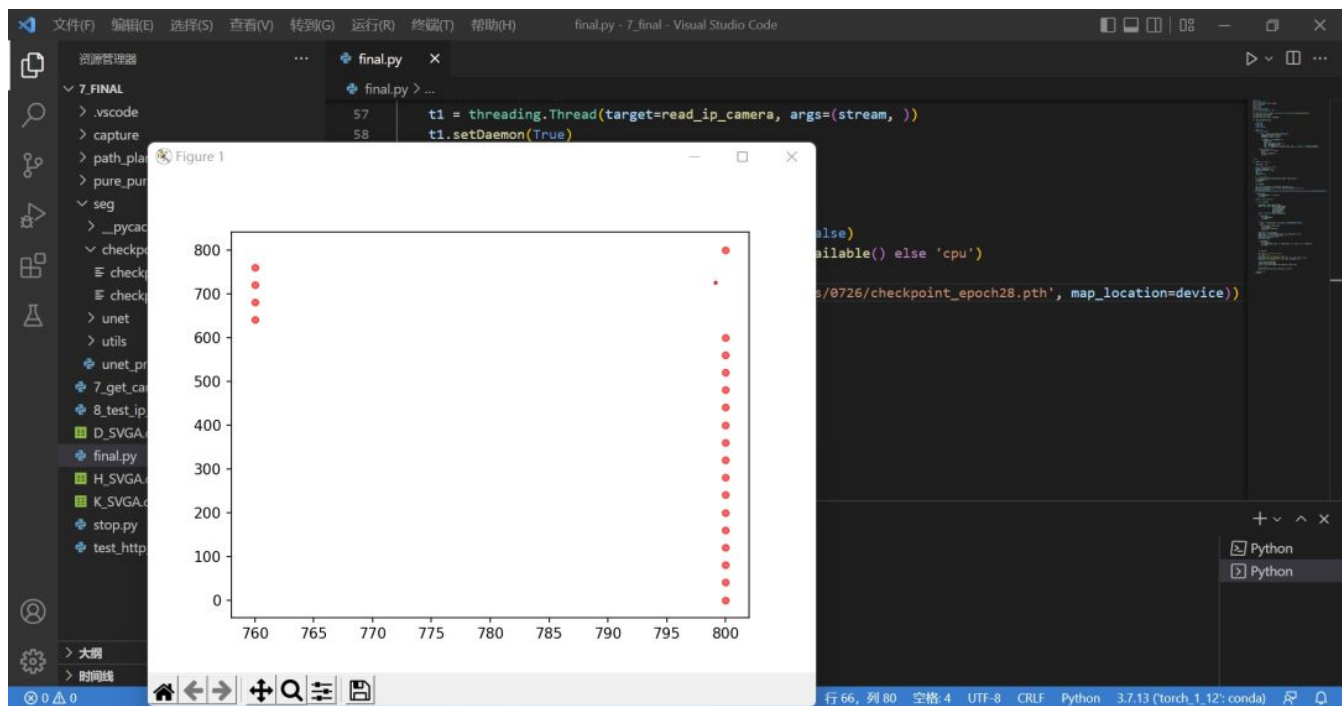














小车路径规划成功；  
利用浏览器控制小车移动；

Start Stream

Stop Stream

Set Record Time (in minute):  2

Set Record Interval (in second):  5

Start Record

Stop Record

## Light


Front Light

Brake Light

## Move

Real-Time Speed From Encoder: 0.00

Set Speed:  90

Set Turning Angle:  45

Forward

Stop

Backward

Left

Straight

Right

项目完成 (参考资料: 人工智能引航计划)

