

招商证券人工智能工程营课程报告

**项目名称：**

基于yolox和ByteTrack的人体识别与跟踪

|  |  |
| --- | --- |
| 院系： | 电气与电子工程学院 |
| 小组成员： | 刘向前 U202112570 |
|  | 陆奕丞 U202111463 |
|  | 陈庭柱 U202111740 |
| 指导教师： | 郑 玮 |
| 2023年12月7日 | |

1 研究背景和项目目标

1.1选题依据

人体检测可以用来判断输入图片(或视频，摄像头)内是否包含行人。如果检测到行人，则给出其具体的位置信息。该位置信息是智能视频监控、人体行为分析、智能驾驶、智能机器人等应用的关键基础。

1.2业界现状介绍

目前，常用的行人检测方法基本分为两大类型。首先是早期的传统检测方法，主要包括光流法、帧差法和背景建模法等，该类算法理论成熟，在实际视频监控项目中应用广泛。

另一方面，基于特征检测目标的方法在近几年取得了不小的发展，最近几年由深度学习提取特征并判断行人目标的技术越来越多，包括 R-CNN（Region Convolutional Neural Networks）、YOLO（You Only Look Once）和 SSD（Single Shot MultiBox Detector）等几类主流的框架。根据不同的检测思路可将这些框架分为两种类型。首先是基于候选框的类型，这类目标检测算法第一步会在原始图像上生成极多的候选框，接着通过类别判断来进行筛选，这个过程对于硬件要求较高，不然运算会非常缓慢，在硬件条件不够的情况下是无法将算法部署用来进行实时检测的。

而这些目标检测方法大多只是提供了广泛的图像识别，而没有提供选中既定目标进行特殊标识与特别呈现。

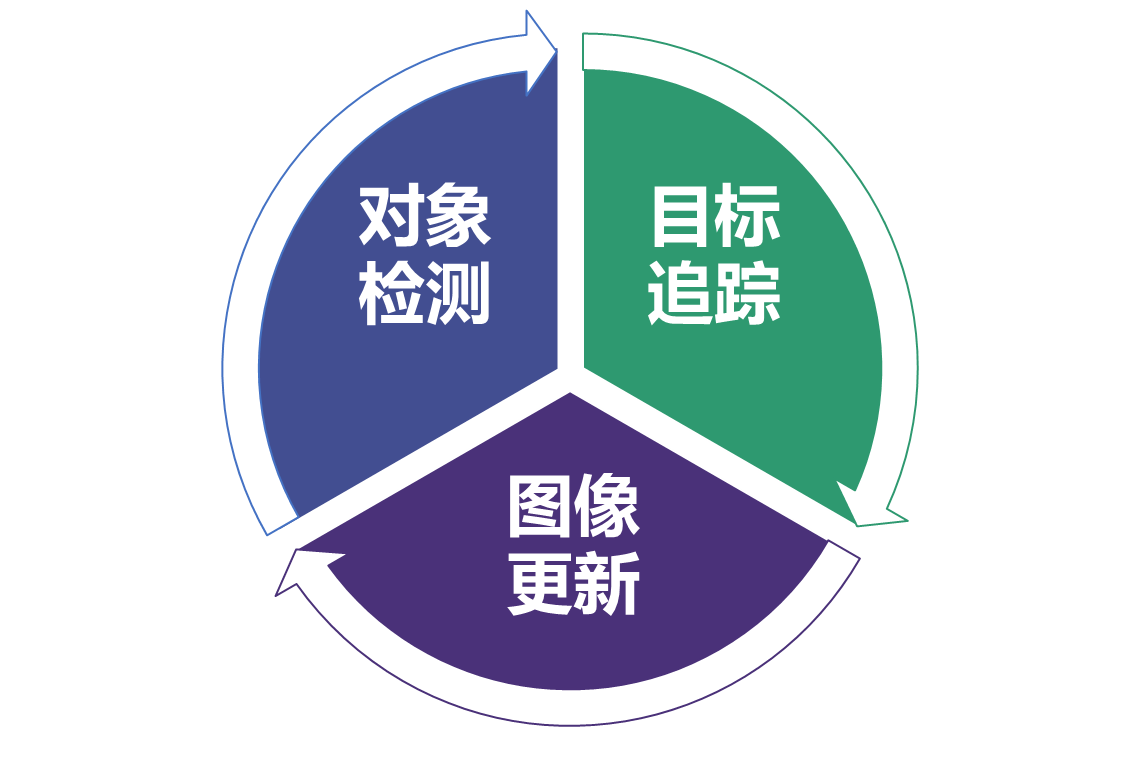
1.3 本项目的目标

本项目，旨在利用BYTETRACK框架与yolo模型，找出摄像头读取到的视频帧中所有的行人，包括位置和大小，并用矩形框与唯一id标识出来。在此基础上，可以人为选中跟踪目标做出特别标识，并使摄像头通过转动等方式跟踪选定目标。目前设计的选定方式为鼠标左键点击相应的检测框，取消选定的方式为鼠标右键点击相应的检测框或特定按钮。除此之外，界面还提供了控制摄像头移动，放大倍数，自动变焦等按钮。具体设定如下：



2 项目总体设计

2.1项目主要运行逻辑



2.2项目分解

根据既定目标，我们将本项目的主题程序分解为四个基本问题，即：

1. 图像识别模块运行逻辑的设计。也就是将在该模块中完成构建GUI界面，完成图像识别等基础功能，同时针对各个基本功能，来完成对函数参数，全局参数，函数的功能，以及进程间通信数据格式的设计。
2. 拉取网络云台视频流的工具类设计。在这个特定的类中，将会完成获取最新数据帧，使用其替换掉未被图像识别模块读取但已明显滞后的数据帧，并将其以进程间通信的方式传递给图像识别模块。
3. 控制摄像头运动的工具类设计。在这一部分的任务中，将会通过控制摄像头转动从而实现对特定跟踪对象的跟踪，同时响应识别模块创建的GUI界面传来的例如自动变焦，摄像头的移动等命令。
4. 上述三个部分的运行方式。我们采用的是进程池统一管理。具体实现为：基于进程池的异步进程，图像识别模块由于其GUI 的特殊性需要不断循环，故单开一个异步进程；而将拉取网络云台视频流的工具类和控制摄像头运动的工具类循环创建异步进程，以实现系统资源的最大利用。

2.3项目组织

由于程序运行标志设置为全局变量将极大的减少各个函数所需要的参数，而且能够及时更新数据，以及各个函数之间存在一定的调用关系，所以我们选择将各个功能各自涉及到的文件分别放在一个文件内，故共有3个工具文件，1个程序入口文件。并且图像识别模块主体循环主要采用逻辑更为清晰的面向过程编程的方法，而程序具体功能的实现利用ByteTrack框架提供的一些工具类采用面向对象的编程方法。

2.4项目内部各个部分之间的关系

对于在2.1中分解出的四个基本部分，其中包含着调用的逻辑关系，主要为：主函数作为程序入口，负责调配三个基本功能的进程；而图像识别模块为程序基础，从拉取网络云台视频流的工具类中获取视频帧，负责主要人体识别功能的实现以及人机交互接口的设计，并将相关参数传递给控制摄像头运动的工具类完成追踪这一功能；拉取网络云台视频流部分负责获取最新数据帧，使用其替换掉已明显滞后的数据帧，并将其及时传递图像识别模块进行处理，实现数据的实时获取与处理；控制摄像头运动的部分负责使用图像识别模块传递过来的参数，利用PD控制算法和PELCO-D中规定的语法实现对摄像头相应的控制。

3 项目关键技术

3.1运行平台

考虑到构建ByteTrack的便捷性，我们采用了win11系统。

3.2语言与工具

考虑到目前python第三方库的广泛适配，以及可以较好的跨平台运行，同时其在计算机视觉、人工智能这一领域的快速发展与较为成熟的技术，我们选择python为程序编写语言。又因为考虑到对各种第三方库版本的适配问题，我们选择了支持较为广泛的3.9版本的python，以及强大的python集成开发环境Pycharm，和python包管理器anaconda。

3.3 库与模型

考虑到BOTSORT的基础框架所需要的依赖，以及图像识别所需要的高效计算，我们使用的python环境包含以下的库：

absl-py 1.4.0

aiohttp 3.8.5

aiosignal 1.2.0

archspec 0.2.1

async-timeout 4.0.2

attrs 23.1.0

blinker 1.6.2

boltons 23.0.0

Brotli 1.0.9

cachetools 4.2.2

certifi 2023.11.17

cffi 1.16.0

charset-normalizer 2.0.4

click 8.1.7

cloudpickle 2.2.1

colorama 0.4.6

coloredlogs 15.0.1

conda 23.10.0

conda-libmamba-solver 23.11.1

conda-package-handling 2.2.0

conda\_package\_streaming 0.9.0

contourpy 1.2.0

cPython 0.0.6

cryptography 41.0.3

cycler 0.12.1

Cython 3.0.5

cython-bbox 0.1.5

cytoolz 0.12.0

dask 2023.6.0

dill 0.3.7

dnspython 2.4.2

ffmpeg-python 0.2.0

filelock 3.13.1

filterpy 1.4.5

flatbuffers 23.5.26

fonttools 4.45.0

frozenlist 1.4.0

fsspec 2023.9.2

future 0.18.3

google-auth 2.22.0

google-auth-oauthlib 0.4.4

grpcio 1.48.2

h5py 3.10.0

humanfriendly 10.0

idna 3.4

imagecodecs 2023.1.23

imageio 2.31.4

importlib-metadata 6.0.0

importlib-resources 6.1.1

Jinja2 3.1.2

jsonpatch 1.32

jsonpointer 2.1

kiwisolver 1.4.5

lap 0.4.0

libmambapy 1.5.3

locket 1.0.0

loguru 0.5.3

Markdown 3.4.1

markdown-it-py 3.0.0

MarkupSafe 2.1.1

matplotlib 3.8.2

mdurl 0.1.2

menuinst 1.4.19

mkl-fft 1.3.8

mkl-random 1.2.4

mkl-service 2.4.0

motmetrics 1.4.0

mpmath 1.3.0

multidict 6.0.2

multiprocess 0.70.15

networkx 3.1

numpy 1.23.5

oauthlib 3.2.2

onnx 1.15.0

onnx-simplifier 0.4.35

onnxruntime 1.16.3

opencv-python 4.8.1.78

packaging 23.1

pandas 2.1.3

partd 1.4.1

Pillow 10.0.1

pip 23.3

pluggy 1.0.0

protobuf 3.20.3

pyasn1 0.4.8

pyasn1-modules 0.2.8

pycocotools 2.0.7

pycosat 0.6.6

pycparser 2.21

Pygments 2.17.2

PyJWT 2.4.0

pymongo 4.6.0

pyOpenSSL 23.2.0

pyparsing 3.1.1

pyreadline3 3.4.1

pyserial 3.5

PySocks 1.7.1

python-dateutil 2.8.2

pytz 2023.3.post1

PyWavelets 1.4.1

PyYAML 6.0.1

requests 2.31.0

requests-oauthlib 1.3.0

rich 13.7.0

rsa 4.7.2

ruamel.yaml 0.17.21

ruamel.yaml.clib 0.2.6

scikit-image 0.19.3

scipy 1.11.3

sympy 1.11.1

tensorboard 2.10.0

tensorboard-plugin-wit 1.8.1

thop 0.1.1.post2209072238

tifffile 2023.4.12

toolz 0.12.0

torch 2.1.0

torchaudio 2.1.0

torchvision 0.16.0

tqdm 4.65.0

typing\_extensions 4.7.1

tzdata 2023.3

urllib3 1.26.18

Werkzeug 2.2.3

wheel 0.41.2

win-inet-pton 1.1.0

win32-setctime 1.1.0

xmltodict 0.13.0

yarl 1.8.1

zipp 3.11.0

zstandard 0.19.0

考虑到对象检测模型的不断迭代带来的性能提高，以及模型大小对检测正确率和程序运行速度的影响，我们选择了较新的，且较小的yolo模型——yolox-s，与行人识别模型——ByteTrack\_s\_mot17.pth.tar

4 项目实现

为了实现项目的基础的人体检测功能，我们使用了ByteTrack中提供的一系列函数与类，同时，为了完成我们自定义的特定目标追踪功能，我们使用了一系列的自定义函数，如下：

4.1 main.py

（1）非自定义函数

BaseManager.register('VideoCapture', VideoCapture): 注册VideoCapture类到BaseManager，用于进程间通信。

BaseManager.register('Magnify', Magnify): 注册Magnify类到BaseManager，用于进程间通信。

Pool(processes=3): 创建一个进程池，最多同时执行3个进程。

pool.apply\_async(func=main,args=(exp,args,q\_main\_mag,size, q\_main\_rtsp)): 异步执行main函数。

pool.apply\_async(func=run\_mag, args=(mag,)): 异步执行run\_mag函数。

pool.apply\_async(func=run\_rtsp, args=(rtsp,)): 异步执行run\_rtsp函数。

（2）自定义函数

run\_mag(magnify): 用于运行视频放大处理的函数。

run\_rtsp(rtsp): 用于运行视频流处理的函数。

main(exp, args, q\_main\_mag, size, q\_main\_rtsp): 初始化函数

4.2 camera.py

（1）非自定义函数

serial.Serial('com4', baudrate=19200, timeout=0.5): 用于获取串口对象

ser.write(): 用于向串口写入数据。

ser.read(): 用于从串口读取数据。

（2）自定义函数

setspeed(speed, ser, address, mode='x'): 控制摄像头速度的函数，根据给定的速度和模式发送控制指令到摄像头。

setspeedxy(speedx, speedy, ser, address): 控制摄像头在x和y方向的速度的函数，根据给定的速度发送控制指令到摄像头。

setstop(ser, address): 控制摄像头停止运动的函数，发送停止指令到摄像头。

moveto(position, ser, address, mode='x'): 控制摄像头移动到指定位置的函数，根据给定的位置和模式发送控制指令到摄像头。

queryPosition(ser, address, mode='x'): 查询摄像头位置的函数，根据给定的模式发送查询指令到摄像头，并返回摄像头位置。

setZoom(ser, address, zoom): 控制摄像头变焦的函数，根据给定的变焦值发送控制指令到摄像头。

queryZoom(ser, address): 查询摄像头变焦值的函数，发送查询指令到摄像头，并返回变焦值。

4.3 ffmpeg\_my.py

（1）非自定义函数

ffmpeg.probe(): 用于获取音视频流的信息。

ffmpeg.input(): 用于指定输入文件或流。

ffmpeg.output(): 用于指定输出文件或流的格式和参数。

ffmpeg.run\_async(): 用于异步执行ffmpeg命令。

（2）自定义函数

ffmpeg\_init(source): 用于初始化ffmpeg处理音视频流的参数，并返回处理进程和视频流信息。

4.4 demo\_track.py

make\_parser(): 创建一个命令行参数解析器，用于解析运行脚本时传入的参数。

get\_image\_list(path): 获取给定路径下的所有图像文件的列表。

write\_results(filename, results): 将跟踪结果写入到指定的文件中。

class Predictor(object): 定义了一个预测器类，用于对输入的图像进行预测。

Predictor.\_\_init\_\_(self, model, exp, trt\_file=None, decoder=None, device=torch.device("cpu"), fp16=False): 预测器类的初始化函数，初始化模型、实验参数、设备等。

Predictor.inference(self, img, timer): 预测器类的推理函数，对输入的图像进行预测，并返回预测结果和图像信息。

image\_demo(predictor, vis\_folder, current\_time, args): 对单个图像进行预测，并将预测结果保存和可视化。

imageflow\_demo(predictor, vis\_folder, current\_time, args,exp,queue\_mag,size\_screen,queue\_rtsp): 对视频流进行预测，并将预测结果保存和可视化。

update(vid\_writer, current\_time, args, results, frame\_id, canvas, photo\_image,exp,queue\_mag,queue\_rtsp): 在GUI界面上更新预测结果。

on\_click\_left(event): 处理鼠标左键点击事件，将点击的目标添加到跟踪列表中。

on\_click\_right(event): 处理鼠标右键点击事件，将点击的目标从跟踪列表中移除。

startid(): 开始目标检测。

stopid(): 停止目标检测。

autozoom(): 开始自动缩放。

starttrack(): 开始目标跟踪。

stoptrack(): 停止目标跟踪。

cancelselect(): 取消所有已选择的目标。

up\_press(event): 处理向上键按下事件，向上移动视角。

down\_press(event): 处理向下键按下事件，向下移动视角。

left\_press(event): 处理向左键按下事件，向左移动视角。

right\_press(event): 处理向右键按下事件，向右移动视角。

release(event): 处理键盘释放事件，停止移动视角。

cancel\_tracking(): 取消所有目标的跟踪。

addzoom(): 增加缩放倍数。

subtractzoom(): 减少缩放倍数。

4.5 重要类

（1） Magnify() # 实现对摄像头的控制

1） 基本属性

# 进程间通信队列

self.queue = queue

# 串口配置

self.size = size

self.ser = serial.Serial('com4', baudrate=19200, timeout=0.5)

self.address = 1 # 设备地址

# 摄像头初始化

setZoom(self.ser,self.address,0)

setstop(self.ser, self.address)

# 画布中心XY坐标

self.canvas\_height = int(canvas\_height)

self.canvas\_width = int(canvas\_width)

self.canvas\_center\_x = int(canvas\_width / 2)

self.canvas\_center\_y = int(canvas\_height / 2)

# 摄像头左右方向速度，画面x方向误差

self.speedx = 0

self.speedx\_last = 0

self.ex = 0

self.ex\_last1 = 0

self.ex\_last2 = 0

# 摄像头上下方向速度，画面y方向误差

self.speedy = 0

self.speedy\_last = 0

self.ey = 0

self.ey\_last1 = 0

self.ey\_last2 = 0

# 摄像头放大倍数，放大倍数误差

self.z = queryZoom(ser=self.ser, address=self.address)

self.z\_last = 0

self.ez = 0

self.ez\_last1 = 0

self.ez\_last2 = 0

# pid控制的比例、积分、微分系数

self.xkp = 10

self.xki = 0

self.xkd = 2

self.ykp = 10

self.yki = 0

self.ykd = 2

self.zoomkp = 400

self.zoomki = 2000

self.zoomkd = 0

# 自动变焦情况下放大倍数理想值

self.zoom = 0.8

# 进程运行标志

self.is\_running = True

# 追踪id

self.confirm\_id = -1

# 程序状态标志

self.zoom\_factor = 0

self.flag\_zoom = True

self.move = 'stop'

2）# 重要方法

run(self)：对摄像头的单次控制

reset(self)：清空之前的状态

（2） VideoCapture() # 实现实时视频流的拉取

1） 基本属性

self.process, self.cap\_info = ffmpeg\_init(url) # 视频流对象和信息

self.frame = None # 最新视频帧

self.ret = None # 当前帧是否有效

self.is\_running = True # 进程运行标志

self.size = self.getsize() # 视频帧的大小

self.bytes = None # 存储视频帧的字节数组

self.queue = queue # 进程间通信的队列

2）# 重要方法

run(self)：拉取最新视频帧并发送给图像识别模块

通过对这些函数和类的正确设计和使用，我们实现了利用ByteTrack框架与yolox模型，找出图像、视频或摄像头读取到的视频帧中所有的行人或物体，包括位置和大小，并用矩形框与唯一id标识出来，同时对人为选中的跟踪目标做出特别标识与实时跟踪，也能正常实现控制摄像头移动，放大倍数，自动变焦等功能。

5 项目测试

经过人为计时与程序本身状态检测，通过测试，我们观察到程序能够正常识别对象，并实现对特定目标的持续跟踪，做出特定标识。稳定帧率为35帧。

6 项目管理

6.1 团队人员组成

刘向前U202112570（组长）；

陆奕丞U202111463；

陈庭柱U202111740。

* 1. 任务分工

1.刘向前：主程序的设计，图像识别部分，GUI界面部分。

2.陆奕丞：摄像头控制部分。

3.陈庭柱：网络云台图像获取部分。

总体而言，每一位同学都及时完成了各自的任务，并以较高的效率进行错误更正，以及交流频繁，沟通较多，为项目最后的完成都有着较大的贡献。

7. 总结与反思

就这个项目本身而言，从技术层面，我们对于一些比较先进的技术有了初步了解，学习，与探索，运用。算是在一定程度上跳出了之前的舒适圈，毕竟我们整个小组之前都没有过计算机视觉方面的相关经历，甚至对python语言的使用都不是很了解，但最后还是较好的完成了既定的目标。从工具使用层面，从团队合作层面，通过此次的分工合作，提高了任务分配的能力，强化了责任意识，同时增进了同学之间的友谊；除此之外，还明显提高了分解复杂问题的能力，以及将零散部分综合为复杂目标的能力。

**评分页**

|  |  |  |  |
| --- | --- | --- | --- |
| 成员姓名 | 刘向前 | 陆奕丞 | 陈庭柱 |
| 贡献度 | 40% | 30% | 30% |

**指导教师评语**

|  |  |
| --- | --- |
| 学生1 |  |
| 学生2 |  |
| 学生3 |  |
| 学生4 |  |
| 学生5 |  |
| 教师 |  |

2024年4月20日