

招商证券人工智能工程营课程报告

**项目名称：**

基于yolov7和BOTSORT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 U202123456 |
|  | 朱龙天 U202112587 |
|  | 陈雨奇U202112563 |
|  | 刘源浩U202112545 |
|  |  |
| 指导教师： | 郑 玮 |
| 2023年9月16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。

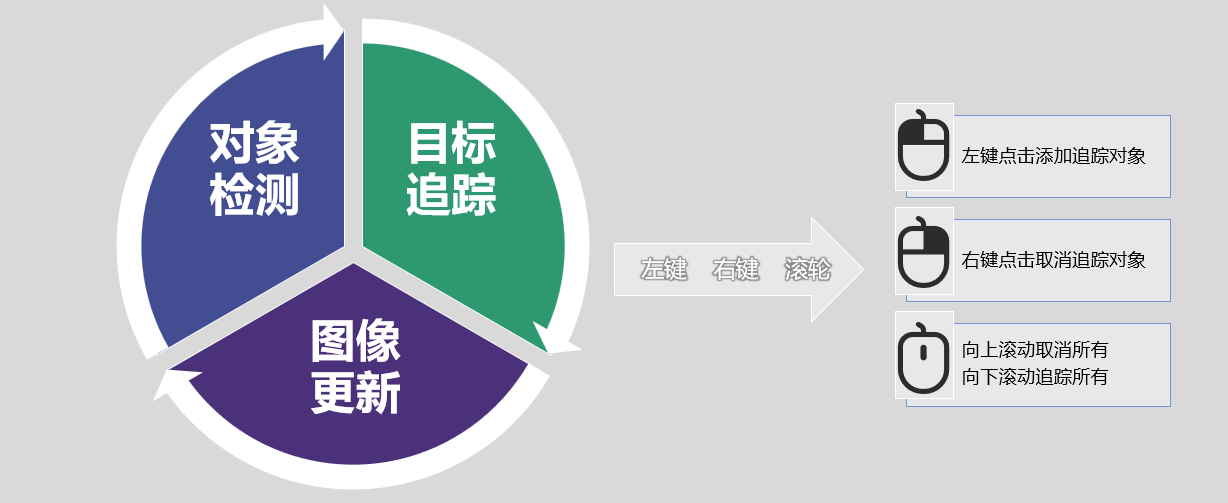
而这些目标检测方法大多只是提供了广泛的图像识别，而没有提供选中既定目标进行特殊标识与特别呈现。

1.3 本项目的目标

本项目，旨在利用BOTSORT框架与yolo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。在此基础上，可以人为选中跟踪目标做出特别标识，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。同时也可以自行更换跟踪模式，如只标识出人类，或标识出其他任何可标识的物体两种模式。

2 项目总体设计

2.1项目主要运行逻辑



考虑到运行平台的局限性，我们使用按钮代替滚轮的作用

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. 主程序detect()运行逻辑的设计，代码所需要的各个函数的设计。也就是将在detect()主程序中完成对GUI界面，设备摄像头或数据流的初始化，同时针对各个基本功能，来完成对函数参数，全局参数，以及函数的功能的初步设计。
2. 主循环图像更新函数update()的实现。在这个特定的函数的中，将会完成对数据帧的对象检测，再根据运行状态标志来进行相应的数据过滤，同时将检测、筛选结果显示到GUI界面上，并且完成对特定追踪目标状态（是否于视频帧中消失）的更新。
3. 图像放大显示函数magnify()以及鼠标坐标转换函数的实现。在这一部分的任务中，将会完成对特定跟踪对象的放大显示（局限于设备原因，只能实现图像的局部放大，而不能实现摄像头的移动）。同时由于Tkinter画布的特殊性，即在完成图像局部放大后鼠标点击位置与实际画布位置不匹配，所以需要根据程序目前所处的运行状态给鼠标点击事件的坐标一定的偏置。
4. 选定追踪目标的事件处理函数，与程序运行状态相关的函数的实现。这一部分的任务包含了一系列函数，如响应左键点击事件的函数将完成对既定目标列表的添加，响应右键点击事件的函数将完成对既定目标列表的移除，以及响应各种按钮点击事件对程序运行状态标志的及时更新。

2.3项目组织

由于程序运行状态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，所以我们选择将所有函数置于一个文件内，并且程序主体循环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BOTSORT框架提供的一些工具类采用面向对象的编程方法。

与此同时，因为程序主体放在同一个文件里，并且特定的运行环境，我们选择了个人完成对应任务后传递给下一位成员，最后统一调试修改。

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：detect()函数将update()函数作为主循环，检测到没有数据流时或用户关闭GUI界面时才退出循环。而update()函数将调用magnify()函数，完成对满足条件的特定选中对象进行放大。在程序主循环过程中，当检测到鼠标点击事件，或按钮点击事件，将跳转到选定追踪目标的事件处理函数，与程序运行状态相关的函数进行跟踪对象的更新以及程序运行状态的更新。

3 项目关键技术

3.1运行平台

由于BOTSORT的依赖库中包含的faiss-gpu库不支持windows平台，同时考虑到其对加快程序运算速度的必要性，以及如果选择更换所带来的效果不确定性，我们决定利用WSL，即Windows Subsystem for Linux，可以提供Linux（Ubuntu 22.04.2 LTS）环境，可以有效解决平台问题

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，我们使用了以下的第三方库：

numpy==1.22.3

opencv-python==4.8.0.76

loguru==0.7.0

scikit-image==0.21.0

scikit-learn==1.3.0

tqdm=4.66.1

torch==2.0.0+cu118

torchaudio==2.0.1+cu118

torchvision==0.15.1+cu118

Pillow==9.3.0

thop==0.1.1.post2209072238

ninja==1.11.1

tabulate==0.9.0

tensorboard==2.14.0

tensorboard-data-server==0.7.1

lap==0.4.0

motmetrics==1.4.0

filterpy==1.4.5

h5py==3.9.0

matplotlib==3.7.2

scipy==1.11.2

prettytable==3.8.0

easydict==1.10

pyyaml==6.0.1

yacs==0.1.8

termcolor==2.3.0

gdown==4.7.1

onnx==1.8.1

onnxruntime==1.8.0

onnx-simplifier==0.3.5

faiss-gpu==1.72

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——yolov7-x，与REID模型——mot17\_sbs\_S50

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BOTSORT中提供的一系列函数与类，如下：

1. set\_logging(): 设置日志记录，用于记录程序运行过程中的信息。

2. select\_device(opt.device): 选择运行设备，可以是CPU或者GPU。

3. attempt\_load(weights, map\_location=device): 尝试加载模型权重。

4. check\_img\_size(imgsz, s=stride): 检查图像尺寸，确保图像尺寸可以被模型的步长整除。

5. TracedModel(model, device, opt.img\_size): 如果启用了模型追踪，将模型转换为追踪模型。

6. BoTSORT(opt, frame\_rate=60.0): 创建一个BoTSORT跟踪器。

8. window.mainloop(): 启动Tkinter窗口的主循环。

9. cap.read(): 从摄像头读取一帧图像。

10. letterbox(img=new\_shape=imgsz,stride=stride): 对图像进行尺寸调整和填充，以适应模型的输入尺寸。

11. torch.from\_numpy(img).to(device): 将numpy数组转换为PyTorch张量，并将其移动到指定的设备上（CPU或GPU）。

12. model(img, augment=opt.augment): 使用模型对图像进行推理。

13.non\_max\_suppression(pred, opt.conf\_thres, opt.iou\_thres, classes=opt.classes, agnostic=opt.agnostic\_nms): 对推理结果进行非最大抑制处理，去除重叠的边界框。

14. tracker.update(detections, im0): 使用跟踪器对检测结果进行跟踪。

15. plot\_one\_box(tlbr, im0, label=label, color=(0,255,0), line\_thickness=2): 在图像上绘制一个边界框。

16. magnify(boxes\_ids,boxes\_tlwhs,tp\_ids,Label,im0): 根据目标对象的位置和状态，对图像进行放大显示。

17. cv2.cvtColor(im0, cv2.COLOR\_BGR2RGBA): 将图像从BGR颜色空间转换为RGBA颜色空间。

18. Image.fromarray(im0): 将numpy数组转换为PIL图像。

19. ImageTk.PhotoImage(image=img): 将PIL图像转换为Tkinter PhotoImage。

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：

1. magnify(boxes\_ids, boxes\_tlwhs, tp\_ids, Label, img): 这个函数用于放大图像中的目标对象。

2. Mouse\_Coordinate\_Conversion(x, y): 这个函数用于将鼠标点击的坐标转换为放大后图像中的坐标。

3. update(opt, model, cap, device, half, tracker, imgsz, stride, count, names, colors, window, Label): 这个函数用于更新目标检测和跟踪的结果。它首先读取摄像头的图像，然后使用目标检测模型进行推理，得到检测结果。接着，它使用目标跟踪算法对检测结果进行跟踪，得到跟踪结果。最后，它根据跟踪结果，绘制边界框和标签，并将结果显示在窗口中。

4. on\_click\_left(event): 这个函数用于处理鼠标左键点击事件。当用户点击图像中的目标对象时，它将该目标对象添加到跟踪列表中。

5. on\_click\_right(event): 这个函数用于处理鼠标右键点击事件。当用户点击图像中的目标对象时，它将该目标对象从跟踪列表中移除。

6. track\_all(): 这个函数用于将所有目标对象添加到跟踪列表中。

7. cancel\_tracking(): 这个函数用于清空跟踪列表，取消所有目标对象的跟踪。

8. cancel\_zoom(): 这个函数用于取消图像的放大模式。

9. zoom(): 这个函数用于开启图像的放大模式。

10. track\_human(): 这个函数用于设置只跟踪人类目标对象。

11. track\_everything(): 这个函数用于设置跟踪所有类型的目标对象。

12. detect(save\_img=False): 这个函数是程序的主函数，用于初始化模型和设备，创建窗口和标签，启动摄像头，以及启动程序的主循环。

其中，最为重要的两个函数为detect(),update()

detect()的主要运行逻辑如下：

1. 导入所需的库和模块。

2. 设置日志记录，用于记录程序运行过程中的信息。

3. 选择运行设备，可以是CPU或者GPU。

4. 加载模型权重。

5. 检查图像尺寸，确保图像尺寸可以被模型的步长整除。

6. 如果启用了模型追踪，将模型转换为追踪模型。

7. 创建一个BoTSORT跟踪器。

8. 打开摄像头，设置摄像头的参数。

9. 创建一个Tkinter窗口和标签，用于显示图像。

10. 创建一些按钮，用于用户交互。

11. 绑定鼠标点击事件到画布。

12. 调用update函数，首次更新目标检测和跟踪的结果。

13. 启动Tkinter窗口的主循环，等待用户的操作和事件。

14. 用户退出或无图像来源时，释放摄像头资源，关闭所有窗口。

update()的主要运行逻辑如下：

1. 对一些全局变量进行声明

2. 对计数器count进行自增操作，并检查是否达到了每30帧打印一次的条件。

3. 读取视频帧，并检查是否成功读取。

4. 如果成功读取到视频帧，进行一系列的图像处理操作，包括尺寸调整、颜色转换和数据类型转换。

5. 使用模型对图像进行推理，得到预测结果。

6. 根据模式的不同，设置目标类别。

7. 对预测结果进行非最大抑制（NMS）处理，过滤掉重叠的边界框。

8. 处理检测结果，包括跟踪器的更新、目标的筛选和结果的保存。

9. 根据目标的状态和类别，绘制边界框和标签，并根据目标是否在tp\_ids列表中，设置不同的颜色。

10. 处理跟踪器中未确认的目标，根据一定的条件将其添加到tp\_ids列表中或更新其状态。

11. 更新全局变量boxes\_tlwhs和boxes\_ids，以便下一次循环使用。

12. 调用magnify函数对图像进行放大处理，并将结果显示在Tkinter窗口中。

13. 将图像转换为Tkinter可用的格式，并更新窗口中的图像。

通过对这些函数的正确使用，我们实现了利用BOTSORT框架与yolo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，也可以自行更换跟踪模式 等一系列功能。

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，通过特别设置后，即便跟踪对象被短时间的遮挡，再次出现在视频帧中时，能正确自动地将其选定为特定目标，做出特定标识。实时模式（即摄像头为数据来源时）稳定帧率为15帧。

6 项目管理

6.1 团队人员组成

刘向前U202112570（组长）；

朱龙天 U202112587；

陈雨奇U202112563；

刘源浩U202112545。

6.2 任务分工

1. 刘向前：主程序detect()运行逻辑的设计，代码所需要的各个函数的设计，程序调试，以及README，报告等文档的编写。

2. 刘源浩：主循环图像更新函数update()的实现。

3. 朱龙天：图像放大显示函数magnify()以及鼠标坐标转换函数的实现。

4. 陈雨奇： 选定追踪目标的事件处理函数，与程序运行状态相关的函数的实现。

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，初次尝试了利用github进行项目管理，确实发现了github对于团队合作中不同任务成果在不同负责人之间流传的便捷性，可靠性，以及可溯源性，可以很快找出错误，明确责任，及时反馈。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 |  |  |  |  |  |
| 预习报告  16%（百分制） |  |  |  |  |  |
| 平时成绩  20%（百分制） |  |  |  |  |  |
| 课程设计与答辩  64%（百分制） |  |  |  |  |  |
| 合计 |  |  |  |  |  |

**指导教师评语**

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

2023年9月16日