CS159-Programing-ZH

Report for

“Magic Mirror of Video”

组名：　　达芬奇的双黄蛋队

成员：　　井普雪　王恬沅

指导老师：　　鲍　杨

15/06/2020

目录

第一章 项目背景

1.1 选题原因

2020 年春季学期，受疫情影响，体育课授课采用线上视频教学的方式进行。由于老师面对学生进行动作教学，其动作方向与学生实际操作方向呈镜面关系，导致学生无法获得直观的动作分解，从而大大增加了对肢体动作要求高的体育项目的学习难度。

本组成员在本学期均选择了太极拳课程，被动作学习所深深困扰，因此希望共同制作一个简单却实用的工具，实现将视频中的人物骨架进行镜像翻转的目的，以便应用于动作类视频学习、舆情监测等领域。

1.2 项目介绍

首先，电脑接受以电脑摄像头/图片/视频三种格式导入需要做人物镜像处理的对象。主要处理方式：导入的视频和摄像头材料切分多帧，通过模型提取每一帧包含关节坐标的列表，并在新建空白画面上呈现人体骨架，进而通过 OpenCv 的函数对骨架进行翻转，最后将切分成帧的视频通过循环拼接。导入图片同理，省却切分成帧和循环插入拼接过程。在第一种方案的同时，我们同步进行通过增加储存人体关节坐标的列表，在 matplotlib 中通过坐标轴上坐标的翻转实现镜像翻转。

1.3 所用库介绍

在该项目的代码实现中，除了 python 标准库外，我们主要用到了三个库

1.3.1 Numpy

在人体姿态估计中的关键点提取中，我们需要用到大量数学方法，因此选择了 Numpy 作为应用的库。NumPy(Numerical Python) 是 Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。

NumPy 通常与 SciPy（Scientific Python）和 Matplotlib（绘图库）一起使用， 这种组合广泛用于替代 MatLab，是一个强大的科学计算环境，有助于我们通过 Python 学习数据科学或者机器学习。

1.3.2 TensorFlow

TensorFlow 是一个采用数据流图（data flow graphs），用于数值计算的开源软件库。节点（Nodes）在图中表示数学操作，图中的线（edges）则表示在节点间相互联系的多维数据数组，即张量（tensor）。

利用 Tensorflow 可实现 tf_openpose 人体姿势估计算法,它还提供了几种变体，这些变体对网络结构进行了一些更改，以便在 CPU 或低功耗嵌入式设备上进行实时处理。

1.3.3 OpenCv

OpenCv(Open source Computer Vision Library)是一个基于 BSD 许可（开源）发行的跨平台计算机视觉库, 实现了图像处理和计算机视觉方面的很多通用算法。

一个典型的计算机视觉算法，应该包含以下一些步骤：(1)数据获取（对 OpenCv 来说，就是图片）;(2)预处理;(3)特征提取;(4)特征选择;(5)分类器设计与训练;(6)分类判别;而 OpenCv 对这六个部分，分别（记住这个词）提供了 API。在本项目中，输入视频的读取、处理与呈现都要借助于 OpenCv 来实现。

1.3.4 Matplotlib

Matplotlib 是一个 Python 的 2D 绘图库，它以各种硬拷贝格式和跨平台的交互式环境生成出版质量级别的图形。通过 Matplotlib，开发者仅需要几行代码，便可以生成绘图，直方图，功率谱，条形图，错误图，散点图等。

在本项目的关键点镜像翻转操作中，我们代码的 version2 需要借助 Matplotlib 来进行坐标变换，并在其中绘制经变换后的关键点坐标散点图。利用该工具，可以将位图矢量化，从而增强稳定性和可读性。

第二章 代码解释

2.1 导入数据库

Argparse：为 py 文件封装好可以选择的参数，解析命令行参数，使其更加灵活
Time：用来获取和转换时间，提供各种时间相关的模块
OpenCv：进行基于计算机视觉的图像处理
Numpy：由多维数组对象和用于处理数组的例程集合组成的库
tf_pose：（用于人体关键点读取，利用 Tensorflow 已经实现）相关代码放置在同一目录名下，见附件

```
import argparse
import time

import cv2
import numpy as np
from tf_pose.estimator import TfPoseEstimator
from tf_pose.networks import get_graph_path, model_wh
```

```
fps_time = 0
```

2.2 指定参数

通过 default 来指定参数（包括图像大小，模块等）

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='tf-pose-
estimation realtime webcam')
    parser.add_argument('--video', type=str, default=0)
    parser.add_argument('--resize', type=str, default='256x256',
                        help='if provided, resize images before they are
 processed. default=0x0, Recommends : 432x368 or 656x368 or 1312x736 ')
    parser.add_argument('--resize-out-ratio', type=float, default=4.0,
                        help='if provided, resize heatmaps before they a
re post-processed. default=1.0')
    parser.add_argument('--
model', type=str, default='mobilenet_v2_large', help='cmu / mobilenet_th
in / mobilenet_v2_large / mobilenet_v2_small')
    parser.add_argument('--show-process', type=bool, default=False,
```

```
                        help='for debug purpose, if enabled, speed for i
nference is dropped.')
    args = parser.parse_args()
```

2.3 获得模型输入图片的尺寸并加载模型进行后续计算

```
    w, h = model_wh(args.resize)
    if w > 0 and h > 0:
        e = TfPoseEstimator(get_graph_path(args.model), target_size=(w,
h))
    else:
        e = TfPoseEstimator(get_graph_path(args.model), target_size=(432
, 368))
```

2.4 数据获取

读取导入的视频，并将视频切分成多帧画面图像

```
    cam = cv2.VideoCapture('ikun.mp4')
    ret_val, image = cam.read()
```

图 1 视频切分成多帧画面图像

获得视频帧率，通过视频解码器来设定输出视频的格式，得到生成的新视频的帧宽和高，并设置生成的新视频的属性

```
    fps = cam.get(cv2.CAP_PROP_FPS)
    fourcc=cv2.VideoWriter_fourcc(*'XVID')
    frame_size = (int(cam.get(cv2.CAP_PROP_FRAME_WIDTH)),int(cam.get(cv2
.CAP_PROP_FRAME_HEIGHT)))
    videoWriter = cv2.VideoWriter('ikun.avi', fourcc, fps, frame_size)
```

进入循环

通过模型的推理操作，将读取的视频中的某一帧输入到模型中，得到包含关节坐标的 humans 这个列表

```
        print(image.shape)
        print('-------')
        humans = e.inference(image, resize_to_default=(w > 0 and h > 0),
 upsample_size=args.resize_out_ratio)
```

2.5 数据处理：humans 列表的实现

对模块读取某一帧视频并输出 humans 列表中的关节坐标的实现。

```
def draw_humans(npimg, humans, imgcopy=False):
        if imgcopy:
            npimg = np.copy(npimg)
        image_h, image_w = npimg.shape[:2]
        centers = {}
        for human in humans:
            # draw point
            for i in range(common.CocoPart.Background.value):
                if i not in human.body_parts.keys():
                    continue
                body_part = human.body_parts[i]
                center = (int(body_part.x * image_w + 0.5), int(body_par
t.y * image_h + 0.5))
                centers[i] = center
                cv2.circle(npimg, center, 3, common.CocoColors[i], thick
ness=3, lineType=8, shift=0)
            # draw line
            for pair_order, pair in enumerate(common.CocoPairsRender):
                if pair[0] not in human.body_parts.keys() or pair[1] not
 in human.body_parts.keys():
                    continue
                # npimg = cv2.line(npimg, centers[pair[0]], centers[pair
[1]], common.CocoColors[pair_order], 3)
                cv2.line(npimg, centers[pair[0]], centers[pair[1]], comm
on.CocoColors[pair_order], 3)
        return npimg
```

补充：在 draw points 过程中若增加储存关键点的列表，后续也可以通过坐标轴的翻转，在 matplotlib 中实现关键点镜像。我们在另一个模型中做了尝试，具体代码部分见 2.6，完整见代码文件 flipped-matplotlib.py

2.6 数据呈现

2.6.1 Version1

新建一张空图，在新建的空图上画出得到的人体骨架

```
        emptyImage = np.zeros(image.shape, np.uint8)
        emptyImage[...] = 0
```

按照 humans 列表中的内容将关节图画出来，并通过 OpenCv 中 cv2.flip()函数，通过设定 flipcode = 1 来实现水平翻转

```
        pose_img = TfPoseEstimator.draw_humans(emptyImage, humans, imgco
py=False)
        pose_img = cv2.flip(pose_img, 1)
```

显示通过以上处理后的这一帧的图像，并将图像插入视频作为一帧，设定时间，完成后通过循环继续读取输入的视频中的某一帧

设定等 1ms 刷新图像，27 为键盘 Esc 键的 Ascii 码，通过 Esc 键可以跳出循环继续后续内容的运行

```
        cv2.imshow('tf-pose-estimation result', pose_img)
        videoWriter.write(pose_img)
        fps_time = time.time()
        ret_val, image = cam.read()
        if cv2.waitKey(1) == 27:
            break
    cv2.destroyAllWindows()
```

图 2 version1 input

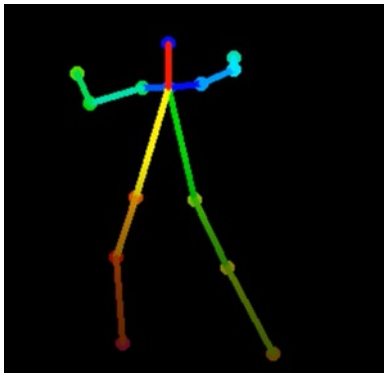


图 3 version2 output

2.6.2 Version 2

通过对 x、y 轴方向的设置，间接实现关键点的镜像绘制（关键点坐标的 x, y 信息分别储存在 x_p, y_p 列表中）

```
x_p = [s[0] for s in flat]
y_p = [s[1] for s in flat]
ax = plt.gca()

ax.xaxis.set_ticks_position('top')  #将x轴的位置设置在顶部
ax.invert_xaxis()  #x轴反向
ax.yaxis.set_ticks_position('right')  #将y轴的位置设置在右边
ax.invert_yaxis()  #y轴反向

plt.scatter(x_p, y_p)
# 暂停
plt.pause(0.2)
plt.clf()
```

图 4 version2 input

图 5 version2 关键点可视化

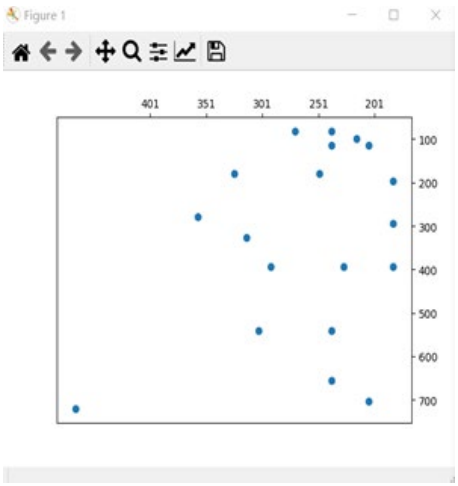


图 6 version2 output

第三章 局限与优化

3.1 局限性

将 3D 的输入转化为 2D 的输出，深度信息丢失，并将前后翻转简化为水平镜像翻转，仅适用于人物正面的镜像学习，而当视频中的人物有较多的侧身动作时，则会导致不同关节区域重叠，影响学习效果。

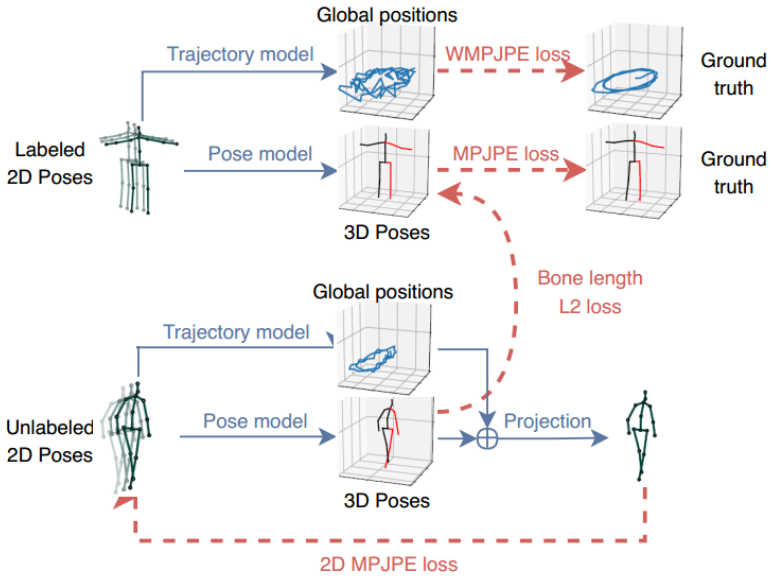
输出模型较为简单粗糙，细节信息缺乏，在解决“同手同脚”现象等方向性动作问题时效果较佳，而没办法解决深度学习、精度学习中的问题。

在导入视频时需要修改源代码，这对于用户来说比较麻烦。

识别精度还可以进一步提高。

3.2 优化设想

在二维关键点上执行时间卷积，将 3D 的姿态估计问题转化为 2D 关键点的检测问题，利用未标记数据进行半监督学习，使用现成的 2D 键点检测器预测未标记视频的 2D 键点，预测 3D 姿态，然后将它们映射回 2D 空间(只需要摄像机的内部参数) 参见开源项目 https://github.com/facebookresearch/VideoPose3D



增加人体姿态估计中识别的关键点个数，提高精度，增强细节识别能力。

第四章 总结

在本次项目中，我们主要利用 OpenCv 函数方法对图片进行读取和呈现，利用以 Tensorflow 为基础实现的 tf-pose 模块识别人体关键点，并连接成骨架。其中的突破点和实用价值来源于：采用了两种方式，分别借助了 cv2 函数/matplotlib，分别对人体骨架/关键点进行了镜像翻转。

在探索实践过程中，我们应用了课堂学习的 python 知识，也对这些包有了更深层次的理解。

在小组成员的探讨和沟通中，明确分工，相互配合，学会合作与交流，包容与理解。

在项目独立完成模块，不断突破边界，探索可能。

懂得多方案进行，在对一种方案坚持完善实现，排除困难的同时，触类旁通，能够思考其他方案实现的可能性，从而保证项目的进展。

在项目进行中，注重结果更注重过程中试错排错的经验积累，明确当前方案的局限性，并制定优化方案和未来预期目标，做到过程完整，具有发展空间。

总之，这次学习让我们受益匪浅，也认识到，还有更多属于 python 的未知精彩等待我们去探索……

参考文献

Documents:

[1] https://docs.opencv.org/master/de/d7a/tutorial_table_of_content_core.html

[2] https://docs.OpenCv.org/master/df/d2c/tutorial_table_of_content_videoio.html

[3] https://tensorflow.google.cn/tutorials/text/nmt_with_attention?hl=zh_cn

Github:

[4] https://github.com/CMU-Perceptual-Computing-Lab/openpose

[5] https://github.com/facebookresearch/VideoPose3D

[6] https://github.com/OpenCv/OpenCv

[7] https://github.com/ildoonet/tf-pose-estimation

CSDN:

[8] https://blog.csdn.net/Horizonhui/article/details/89243449