首都师范大学

人体姿态估计

第七组 王以恒 吴悠 袁忻泽 董子萱 陈云龙

目 录

CONTENTS

01

算法讲解

视频可视化分析

这个模块中，我们会用到4个.py文件，每个文件的功能不一样，其基本原理是通过获取人体某一个具体的关键点的坐标，进行绘图，然后将绘图结果进行整合，最后转换为视频，已实现对某一个具体的点进行可视化，以便于更好的进行运动姿态检测。

这部分我们主要是对已有的视频进行处理，然后将视频中人物动作变化通过一个可视化的曲线表示出来。在这个模块中，我们实现这个功能的原理大概是先对视频人物的动作进行分析，识别出关键点。然后再对视频进行逐帧处理，将其转化为多张图片进行储存。最后，根据识别出的关键点画出对应的点图，再将图与原图片统一之后再还原成视频状态。

视频可视化分析的步骤如下：

可视化步骤1：视频预处理

通过identity_processvideo.py这个文件里面的process_frame函数以及generate_video函数对原视频进行分析。

process_frame这个函数主要是通过对视频逐帧图片的处理，区别并分类了33个关键点，并进行了处理。

generate_video函数这个函数主要是把process_frame这个函数生成的含有33个关键点的坐标生成一个新的视频，取名为‘out-video_name.mp4’。

这个函数无返回值的。

```
if results.pose_landmarks:
    mpDraw.draw_landmarks(img, results.pose_landmarks, mpPose.POSE_CONNECTIONS)
    for i in range(33):
        cx = int(results.pose_landmarks.landmark[i].x * w)
        cy = int(results.pose_landmarks.landmark[i].y * h)
        if i == 0: #鼻尖
            img = cv2.circle(img, (cx, cy), 10, (0,0,255), -1)
        elif i in [11, 12]: #肩膀
            img = cv2.circle(img, (cx, cy), 10, (223, 155, 6), -1)
        elif i in [23, 24]: #髋关节
            img = cv2.circle(img, (cx, cy), 10, (1, 240, 255), -1)
        elif i in [13, 14]:
            img = cv2.circle(img, (cx, cy), 10, (140, 47, 240), -1)
        elif i in [25, 26]:
            img = cv2.circle(img, (cx, cy), 10, (0, 0, 255), -1)
        elif i in [15, 16, 27, 28]:
            img = cv2.circle(img, (cx, cy), 10, (223, 155, 60), -1)
        elif i in [17, 19, 21]:
            img = cv2.circle(img, (cx, cy), 10, (94, 128, 121), -1)
        elif i in [18, 20, 22]:
            img = cv2.circle(img, (cx, cy), 10, (16, 144, 247), -1)
        elif i in [27, 29, 31]:
            img = cv2.circle(img, (cx, cy), 10, (29, 123, 243), -1)
        elif i in [28, 30, 32]:
```

```
while cap.isOpened():
    success, frame = cap.read()
    frame_count += 1
    if not success:
        break
cap.release()
print('视频总帧数为', frame_count)
cap = cv2.VideoCapture(input_path)
frame_size = (cap.get(cv2.CAP_PROP_FRAME_WIDTH), cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
# 视频解码，解码器选成mp4格式
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
fps = cap.get(cv2.CAP_PROP_FPS)

out = cv2.VideoWriter(output_path, fourcc, fps, (int(frame_size[0]), int(frame_size[1])))
```

可视化步骤2：视频逐帧转换图片

通过identity_videointopicture.py这个文件里面的video_into_picture函数对上个步骤中生成的‘out-video_name.mp4’视频进行逐帧转换为图片，并把图片存入pic文件夹目录。

首先进行对pic文件夹进行文件判断，有的话就把文件夹里面清空。

这里的interval就是我们要按多少帧进行转换，num初始为1，进行逐帧转化，同时也把每一张图片对应的33个关键点坐标进行保存，赋值给了变量lmlist。

这个函数是有返回值的，返回值为列表lmlist。

```
def video_into_picture(video_path = 'F:/human identity/main/out-6.mp4',
                       output_path = 'F:/human identity/pic/',
                       interval = 1):
    file_name = glob.glob("F:/human identity/pic/*")
    # 检测文件夹里面是否为空，如为空，则file_name = [],  进入if判断
    if not file_name:
        print("File doesn't exit")
    else:
        for file in file_name:
            os.remove(file)
        print("所有文件删除成功")
```

```
            try:
                if num % interval == 1 or num % interval == 0:
                    file_name = '%d' % num #给转换的图片进行命名
                    cv2.imwrite(output_path + str(file_name) + '.jpg', frame)  # 以.jpg的格式输出图片，保存在指定的目录下


                    img_RGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                    results = pose.process(img_RGB)
                    for id, lm in enumerate(results.pose_landmarks.landmark):
                        lmlist.append((int(file_name), id, lm.x, lm.y, lm.z))
                    cv2.waitKey(1)
                num = num + 1
```

可视化步骤3：将lmlist写入csv文件

通过identity_Visualization.py这个文件里面的write_csv函数将上一个步骤的返回值lmlist进行处理，存成一个csv文件。

write_csv这个函数将原lmlist列表的各个值重新进行了分类，从一个一维列表转换成了一个二维的csv文件，方便我们在后续获取某一个点的关键点坐标。

这个函数无返回值的。

```python
#通过视频进行姿态估计， 将33个关键点坐标生成csv文件，无返回值
def write_csv(self, lmlist):
    text = ['picture_name', 'points_id', 'x', 'y', 'z']
    with open('landmark.csv', 'w', newline="") as f:
        csvwriter = csv.writer(f)
        csvwriter.writerow(text)
        csvwriter.writerows(lmlist)
    print("csv文件生成完成")
```

```
picture_name,points_id,x,y,z
1,0,0.48973017930984497,0.31701746582984924,-0.13903971016407013
1,1,0.4921774864196773,0.31017038226127625,-0.11738558113574982
1,2,0.4939887225627899,0.3110363781452179,-0.11738327890634537
1,3,0.495878666639328,0.31212398409843445,-0.11744804680347443
1,4,0.4880838990211487,0.3107142746448517,-0.11512835323810577
1,5,0.4869706332683563,0.31169408559799194,-0.11512477695941925
1,6,0.4858059287071228,0.3127855062484741,-0.11508777737617493
1,7,0.4995543360710144,0.327092707157135,-0.028648369014263153
1,8,0.48392340540885925,0.3273980915546417,-0.017927294597029686
1,9,0.49245789647102356,0.33051833510398865,-0.10870551317930222
1,10,0.4872632920742035,0.3309757709503174,-0.10567786544561386
```

可视化步骤4：读取csv文件，获取坐标

通过identity_Visualization.py这个文件里面的readcsv_get_landmarks函数将上一个步骤的生成的csv文件进行处理，获取我们想要的某一个关键点的x、y、z坐标。

readcsv_get_landmarks这个函数的参数变量有两个，一个是csv_path，指的是csv文件的路径，另一个是points，范围是[0,32]，也就是33个关键点的编号。

这个函数是有返回值的，返回值为某一个关键点的x、y、z坐标（当然，这里为了处理问题，获取了y轴坐标）。

```python
#读取csv文件，并且提取某一个关键点的值，返回值是特点关键点的y轴坐标
def readcsv_get_landmarks(self, csv_path, points):
    with open(csv_path, 'r') as f:
        reader = csv.reader(f)
        header_row = next(reader)
        lm = []
        # enumerate()获取每个元素的索引及其值
        # for index, column_header in enumerate(header_row):
        #     lm.append([index, column_header])
        for row in reader:
            if row[1] == str(points):
                lm.append(row)
        lm1 = []
        for i in lm:
            res = list(map(float, i))
            lm1.append(res)
        x, y, z = [], [], []
        for i in lm1:
            x.append(i[2])
            y.append(i[3])
            z.append(i[4])
        return y
```

可视化步骤5：绘图坐标图片并储存

通过identity_Visualization.py这个文件里面的get_plt_picture函数将上一个步骤的返回的某个关键点的坐标进行绘图，并将图片保存。

get_plt_picture这个函数将视频总帧数为x轴坐标，将某个关键点坐标作为纵坐标进行绘图，并将所有的图片储存在plt文件夹下。

这个函数无返回值的。

```
#通过读取某一个关键点的坐标集合，生成逐帧坐标图片，无返回值
def get_plt_picture(self, frames, landmark_lmlist):
    frame = np.arange(1,frames+1)
    fig = plt.figure()
    plt.xlim(xmin=0, xmax=frames)  # x轴的范围[0,frames]
    plt.ylim(ymin=0, ymax=1)  # y轴的范围[0,2]
    plt.xlabel('X')
    plt.ylabel('Y')
    # 调整x轴刻度
    plt.xticks(np.linspace(0, frames, 10))
    # 调整y轴刻度
    plt.yticks(np.linspace(0, 1, 11))
```

```
    for i in frame:
        if i == 1:
            ax = plt.gca()
            plt.scatter(frame[0], landmark_lmlist[0], color="red")
            plt.savefig('F:/human identity/plt/' + str(i) + '.jpg')
            pbar.update(1)
        else:
            ax = plt.gca()
            plt.plot(frame[0:i], landmark_lmlist[0:i], color="red")
            plt.savefig('F:/human identity/plt/' + str(i) + '.jpg')
            pbar.update(1)
```

可视化步骤6：合并图片并储存

通过identity_Visualization.py这个文件里面的paste_picture函数将保存在pic以及plt文件夹下的图片利用PIL库中的Image模块进行合并，并储存在plt+pic文件夹中。

paste_picture这个函数输入参数有4个，视频的帧数、两个图片文件的储存路径、输出文件的储存路径。

这个函数无返回值的。

```
#将pic文件夹下的图片与plt_pic下的图片进行整合，合并为一个图片，存储在plt+pic文件夹下
def paste_picture(self, frames, input_path1='F:/human identity/pic/'
                  ,input_path2='F:/human identity/plt/'
                  ,output_path='F:/human identity/plt+pic/'):
    print("准备合并图片")
```

```
        with tqdm(total=frames - 1) as pbar:
            for i in range(frames):
                try:
                    img1 = Image.open(input_path1 + str(i+1) + '.jpg')
                    img2 = Image.open(input_path2 + str(i+1) + '.jpg')
                    img1.paste(img2,(0,0))
                    img1.save(output_path + str(i+1) + '.jpg')
                    pbar.update(1)
                except:
                    print('中途中断')
                    pass
```

可视化步骤7：将图片转换为视频

通过identity_Visualization.py这个文件里面的picture_into_video函数将plt+pic文件夹目录下的图片再转换成为视频。

picture_into_video这个函数输入参数有4个，输出视频的名字name、视频的总帧数、图片输入路径、视频输出路径。

这个函数无返回值的。

```
#将plt_pic文件夹下所有的图片转换为一个视频：无返回值
# a. 使用循环获取路径下的所有图片；
# b. cv2.imread()读取所有图片；
# c. 将读取的图片存于列表中；
# d. 使用cv2.VideoWriter()创建VideoWriter对象，注意参数的设置；
# e. 使用cv2.VideoWriter().write()保存每一帧图像到视频文件；
# f. 释放 VideoWriter对象；
def picture_into_video(self, name, frames
                       ,input_path='F:/human identity/plt+pic/'
                       ,output_path='F:/human identity/video/'):...
```

```
    img_array = []
    file = []
    for num in range(frames):
        fname = input_path + str(num+1) + '.jpg'
        file.append(fname)
        # file = sorted(glob.glob(input_path + '*.jpg'), key=os.path.getsize)
    print("准备将图片转换为视频，现生成图像数组")
```

```
    with tqdm(total=frames - 1) as pbar:
        for filename in file:
            img = cv2.imread(filename)
            height, width, layers = img.shape
            size = (width, height)
            img_array.append(img)
            pbar.update(1)
    print("图像数组生成完毕，下面转换为视频")
    with tqdm(total=frames - 1) as pbar:
        out = cv2.VideoWriter(output_path + str(name) +'.mp4', cv2.VideoWriter_fourcc(*'DIVX'), 15, size)
        for i in range(len(img_array)):
            out.write(img_array[i])
            pbar.update(1)
```

可视化步骤8：读取视频并进行计数

通过identity_Visualization.py这个文件里面的put_number_into_video函数读取视频并且计数，然后实时显示在视频上。

put_number_into_video这个函数一共有6个输入参数，视频路径、视频总帧数、肩、手、肘、嘴的关键点坐标。

这个函数无返回值的。

```python
#读取视频，进行计数
def put_number_into_video(self, video_path, frames, lmlist_shou, lmlist_jian, lmlist_zhou, lmlist):
    pTime = 0
    cap = cv2.VideoCapture(video_path)
    for i in range(frames):
        success, img = cap.read()
        if success == False:
            break
        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
```

```python
if lmlist[i]<lmlist_shou[i] and lmlist_jian[i]<lmlist_zhou[i]:
    self.logical_up = 1
elif lmlist[i]>lmlist_zhou[i] and self.logical_up:
    self.logical_down = 1

if self.logical_up and self.logical_down:
    self.count += 1
    self.logical_up, self.logical_down = 0, 0
```

```python
# 将计数结果实时反馈至屏幕上
cv2.putText(img, "FPS:", (1600, 50), cv2.FONT_HERSHEY_PLAIN, 2, (0, 0, 255), 3)
cv2.putText(img, str(int(fps)), (1700, 50), cv2.FONT_HERSHEY_PLAIN, 3, (0, 0, 255), 3)
# cv2.putText(img, "The number is :", (0, 100), cv2.FONT_HERSHEY_PLAIN, 2.5, (255, 0, 0), 3)
cv2.putText(img, str(int(self.count)), (1700, 250), cv2.FONT_HERSHEY_PLAIN, 3, (0, 0, 255), 3)
#由于cv2.putText在输入中文后显示了？？？，不能输出中文，故在Chinese.py文件中封装函数，在对其使用，起到输入中文的目的
img = cv2AddChineseText(img, "引体向上数量为：", (1200, 300), (255, 0, 0), 60)
cv2.imshow("Image", img)
cv2.waitKey(1)
```

可视化额外：count_frame函数与初始值

通过identity_Visualization.py这个文件里面的count_frame函数来统计视频帧数。

__init__函数进行初始值设置，设置了计数变量count以及两个逻辑变量logical_up以及logical_down。

这个count_frame函数有返回值，返回值为视频的总帧率。

```
#设置初始值，定义两个逻辑值变量均为0，用于计数
def __init__(self,logical_up=0, logical_down=0, count=0):
    self.logical_up = logical_up
    self.logical_down = logical_down
    self.count = count
#计算输入视频帧率，返回值是视频的帧率
def count_frames(self, video_path):...
```

实时摄像头分析

这个模块中，我们会用到2个.py文件，每个文件的功能不一样，其基本原理是通过获取人体某一部分的具体的关键点的坐标，进行角度运算，然后将计算的角度结果进行整合，最后实现计数，以便于更好的进行运动姿态检测。

这部分我们主要是对实时摄像头下进行人体姿态关键点的局部处理，然后在通过实时计算角度进行分析，最终实现计数。在这个模块中，我们实现这个功能的原理大概是先对视频人物的动作进行分析，识别出关键点。由于大多数运动都是一个二分类（up-down-up），我们设计了一个ui界面，可以支持5种运动。

实时摄像头运动姿态分析的步骤如下：

分析计数模块

这个模块中，我们会先预设一些默认值，方便我们计数时使用，我们对已经识别出来的人体关键点进行分析，通过不同关键点之间的位置，来判断运动运动时动作是否标准，并对其进行计数

```
def default(self):  # 一些默认值
    self.filename = []
    self.keypoint = {0:[(23,11,23,25),(24,12,24,26)], 1:[(13,11,13,15), (14,12,14,16)], 2:[(11,23,11,13), (12,24,12,14)]}
    self.angle = {0:[90,100], 1:[110,140], 2:[40,160]}  # 角度范围 0仰卧起坐躯干-大腿角度, 1俯卧撑肘关节角度, 2.....以此类推
    self.countBtn = False
```

分析计数模块

所谓默认值，比如这个keypoint，便是对人进行动作捕捉时的33个关键点中用于计数的几个关键点，比如0便是躯干大腿角，1是肘关节角等等。

而angle，便是用于判断动作标准的角度范围，0是仰卧起坐躯干-大腿的角度，1是俯卧撑肘关节的角度等等。

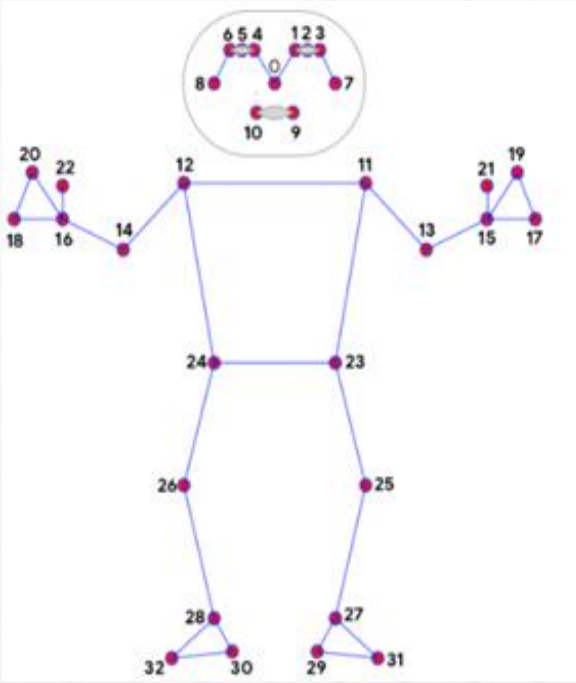
```
def default(self):  # 一些默认值
    self.filename = []
    self.keypoint = {0:[(23,11,23,25),(24,12,24,26)], 1:[(13,11,13,15), (14,12,14,16)], 2:[(11,23,11,13), (12,24,12,14)]}
    self.angle = {0:[90,100], 1:[110,140], 2:[40,160]}  # 角度范围 0仰卧起坐躯干-大腿角度, 1俯卧撑肘关节角度, 2.....以此类推
    self.countBtn = False
```

分析计数模块

有了设置默认值，我们通过选择不同的keypoint和angle数值来对不同的动作进行计数，我们用引体向上来举例，那我就选择act 2 ，那么对应的值为（11,23,11,13）、（12,24,12,14），对应右图我们可以看到取得是肩肘腰的点，再通过连线之间的夹角角度来实现计数我们设置了一个最大角度160°，当人垂直吊在单杠上，那便会大于160°，当人向上发力直至下巴过杠这个角度会持续变小，我们将这个最小角度设置为40°，当角度小于40°时，算是经过一次角度的变换，我们便让计数+1。

```
def analyze(self):  # 开始计数
    fileName = QFileDialog.getSaveFileName(sel
    # print(fileName) 获取一个元组，一共有两个数据，
    if fileName[0]:  #进行if判断，增加容错
        act = self.comboBox.currentIndex()
        # print(act) act = 字典keypoint的key值，
        if act == 1:
            joint1 = self.keypoint1[act]
            joint2 = self.keypoint2[act]
        else:
            joint1 = self.keypoint1[act]
        angleRange = self.angle[act]
```

```
# 判断并计数
if act == 1:
    if joint1 > angleRange[1] and joint2 > 155:
        down = True
    elif down == True and joint1 < angleRange[0]:
        up = True
    if down == True and up == True:
        count += 1
        down = False
        up = False
else:
    if joint1 > angleRange[1]:
        down = True
    elif down == True and joint1 < angleRange[0]:
        up = True
    if down == True and up == True:
        count += 1
        down = False
        up = False
```



分析计数模块

通过判断关键点的位置来计数

02

优缺点分析

优缺点分析

视频可视化：

优点：条理清晰、步骤明了，代码构成结构较为简单，实现原理较为简单，可视化让人体姿态检测变得生动具体、能够很直观的反映出人体运动的变化，进而设置标准值。

缺点：代码运行时间较长，因为涉及到视频、图像的混合处理，运行一次的耗时较大。同时计数模块还是存在一定的逻辑缺陷，不能很好地适应多个视频

实时摄像头分析：

优点：运动种类多，支持5种运动类型的实时分析；ui界面清晰，一目了然，使用方便；可调节性高，计数模块可以很方便的修改，应对不同情况。

缺点：对关键点实时检测要求高，如果关键点不在摄像头中，或者关键点出现问题，导致计数不成功；角度计算要求设置范围需要精确，不然会有较大的计数偏差

03

反思与总结

反思与总结

王以恒：这一次大作业，对我们组来说是一个很大的挑战，因为大家都是对这方面不是非常的了解，甚至说是一无所知，在刚开始的时候不知道该如何下手。但是随着老师的讲解以及自身的不断学习，每天在电脑上进行不断的研究，终于完成了这个挑战，虽然有瑕疵，但是我认为瑕不掩瑜。

作为第七组的组长，我积极带领我的组员们进行大作业的研究，经常通过腾讯会议进行python学习、机器学习的一些交流，通过这些交流，我巩固了已学习的知识，了解了我不太会的知识。可以说收获很多，在这里，也向我的组员说一声谢谢，感谢你们的陪伴，我们一起解决困难。

但是这次大作业还是存在着瑕疵：比如在视频可视化中还是没有根据阈值的设定来计数，可视化模块是通过图片合并生成的，不是实时生成的等等。

反思与总结

袁忻泽：这次大作业的经历，时间线拉得比较长，要完善的东西也比较多。我从零开始认识mediapipe并见识到了其强大的功能，也意识到了合理运用这些功能强大的库能做出很多对我们生活便利的工具程序。

在对人物动作计数这方面也想了很多种方法，比如将识别出的关键点转化成为像素点，再比较像素点之间的位置关系，然后完成计数。但之后我们还是选择了计算关键点之间的角度，看做运动时角度是否达到标准，通过角度的变化来判断运动人的运动姿态。然后我们又加上了摄像头实时识别，使功能更加多样化。到后来也加上了可视化图像，能更加直观的看出运动人的运动姿态的变化。

在这个项目中，我学习到了很多的东西，也通过小组合作交流提升了我的团队合作能力，学会了在团队中贡献出我自己的一份力量。经过这一系列的经历，我也认识到了自己的不足，有很多设计我并不能独自实现，今后也要加深自己知识的学习，提高自己的能力。

反思与总结

陈云龙：这次大作业我们做的是人体姿态识别，第一次触碰这个领域，开始也是一头雾水，在组长的带领下，我们慢慢学习，了解了mediapipe的用法以及功能，中途虽然也遇到了很多困难，但有组长带领下，我们攻克了难关。也是这次让我深刻体会到了注释的重要性，其他人完成的部分都要靠着注释去理解。通过此次大作业项目，我感受颇多，完全不像平时的课堂练习那么简单，都是些自己未知的领域，需要自己探索。

反思与总结

董子萱：本次大作业，我们通过mediapipe实现了对人体姿态的识别，完成了对引体向上，俯卧撑等运动动作的计数功能，在体会到mediapipe功能之强大和多样性的同时，业增长了我们自己处理问题的能力，在最开始拿到课题时觉得无从下手，但是在组长的带领下，我们每个人分配了任务，有条不紊的去把每一个步骤完成，将一个大难题拆解成一个个小难题去攻克，才有了最后的成果，大家一起学习，协同进步，总的来说是一次难得的经验，通过这一次大作业，我也认识到自己的不足，以后一定也增强自己能力上的提升，知识上的积累。

反思与总结

吴悠：之前没有接触过计算机视觉这个领域，刚上课时觉得很新奇，随着之后上课越发明白了计算机视觉在现今和未来有着重要作用。这个大作业算是让我初探计算机视觉。我跟其他组员通力协作，分工明确，在完成大作业过程中，不仅学到了很多计算机视觉相关知识，学会了mediapipe库的调用，也明白了出现什么error时如何去解决。经过这次大作业，令我受益匪浅。

Thank You

感 谢 聆 听

学生：第七小组

指导老师：刘铁