Final Project. Report

Anna Boronina

I. Introduction

HUMAN pose and shape estimation is an ongoing challenge in the Computer Vision and Deep Learning area of study. The researches create models for human mask detection [1], human pose estimation [2], human keypoint detection [3], human model reconstruction [4]. There are 3 most common types of human pose modeling: skeleton-based [3], contour-based [1] and volume-based [4]. The first model outputs a list of key points it detected in a given image. These points can be used to reconstruct a human skeleton. Contour-based human detection model is related to instance segmentation; it outputs one mask per detected human. The last type of human detection is volume-based. It captures a human shape and gives its volumetric representation.

All of these methods can be used for abnormal behavior detection, person counting and identification, sign language translation, and accidents detection, such as a person falling or getting hit by a car [5].

II. Related Work

Due to the popularity of human detection and segmentation, researchers constantly strike to improve existing solutions. For sign language, Ko et al. [6] created detection of sign language using key points; Later, the same authors [7] trained a skeleton-based model to translate sign language. In 2019 Cao et al. released OpenPose library [8]. It estimates human pose with up to 20 classes of body parts. Later, the same authors released real-time gesture grading [9] based on OpenPose. The competitor of OpenPose is Resnet50, which is a bit older than the former. Nevertheless, Oztel [10] noticed that the first 40 depths of the model give a better performance than the first 46 depths, and they used it for human detection. Similarly, Xiao et al. [2] used an improved Resnet50 for human pose estimation.

Many researchers compare models for human pose estimation such as DeepCut, DeeperCut, OpenPose, AlphaPose for different body parts: head, shoulder, elbow, etc. According to Li [11], OpenPose and AlphaPose outperform DeepCut and DeepedCut.

III. Idea Explanation

The idea of the project is to use human keypoints estimation for workout sessions. During the pandemic, most people got locked up in their houses without access to gyms, personal trainers, or group workout sessions. In this case, the only choice is to work out from home using YouTube tutorials. Unfortunately, an incorrect way of doing an exercise might be useless or, even worse, cause trauma.

My project is targeting the problem explained above. Given two images or one image and one video, the algorithm will calculate the accuracy of the correctness of some physical exercise or yoga pose performance.

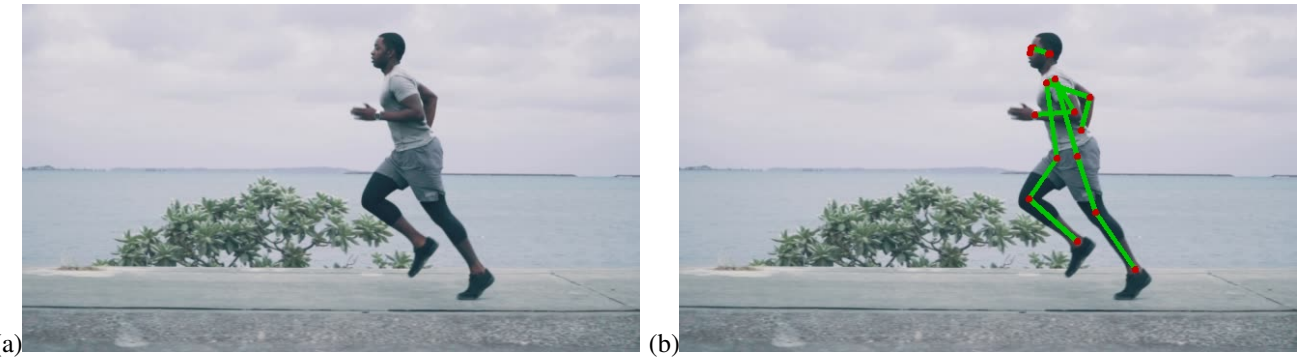
IV. The current state

For human pose estimation, I use KeypointRCNN. The main reason is that PyTorch has the implementation and is easy to use. I have attempted to use OpenPose, yet it has to be imported from someone's repository, and most authors didn't provide a convenient way to do it.

KeypointRCNN accepts an image and returns, mainly, scores and keypoints for each detected human. Score is just number which describes the model's certainty about one human. A set of keypoint has maximum length of 17 and contains one class per one human skeleton point, e.g. left or right eye, ear, shoulder, knee, etc.

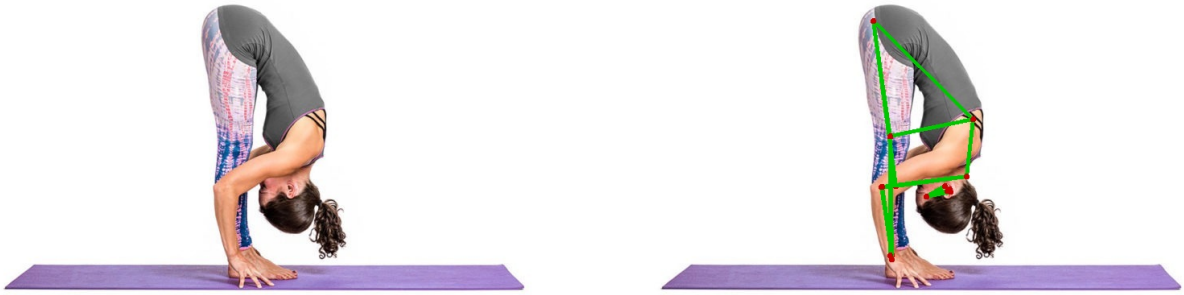
The experiments have shown that the model is better at detecting a human if the input image was not normalized.

The code contains documentation and comments, therefore I will not discuss many implementation details in the report.



(a) (b)

Fig. 1. (a) Original image (b) Detected human



(a) (b)

Fig. 2. (a) Original image (b) Detected human

Link to the Colab Document.
Link to the results - images and videos.
The structure of the folder:

```
images
  comparison
  detection
videos
  detection
  comparison
```

In the *images* folder, there are pairs: (*image*, *image_detected*), where the first one is the original image, and the second one is the original image with skeleton drawn.

The *videos* folder contains two folders: *comparison* and *detection*. *Comparison* folder contains triples (*image*, *video*, *video_compared*), where the first one is the image of a correctly performed exercise; the second one is *how* the exercise was performed by someone else. The last one is the result of my algorithm. In the top left corner it shows the overall accuracy of performing the exercise, and to each skeleton point there is an accuracy score attached. The same applies to *images* folder.

A. Detection

1) Images: For images, it's simple: for a given image, KeypointRCNN detects all people it is sure about. It gives a skeleton description, which I use to draw points and edges on the image. The pseudocode:

You can see the short version of the result in Fig. 1 and Fig. 2. Unfortunately, the model didn't give very good results for the the second image.

2) Videos: A video is a sequence of images, therefore the algorithm for people detection is the same as for a photo. It requires some additional processing for video though.

The examples can not be attached to the reports but they are available in the results folder mentioned above.

B. Comparison

For comparison it's important that an image or a video contains exactly one person. If it has more, then the results are unpredictable.

In this part of the project I do not use face keypoints anymore because they are unimportant for comparison of workout activity.

Fig. 3. (a) Target image (b) Skeleton of target (c) Original human (d) Skeleton of original (e) Comparison

1) Images: After processing each image there are just two skeletons, and we know coordinates of each point of the skeleton. For detection, I connect them by lines and draw. For comparison, it is useless. I predefined the following "bends" (a bend is two vectors sharing one point): an arm is a bend of two vectors (or lines, or edges): shoulder-to-elbow and elbow-to-wrist. Another example is a shoulder, e.g.: right-shoulder-to-left-shoulder and left-shoulder-to-left-hip. Since a bend is a pair of two vectors (or lines, or edges) it's represented as four numbers, two of which are the same number. The number is the keypoint number. For example, for shoulder-to-elbow and elbow-to-wrist bend it would be $((6, 5), (5, 11))$, where 6 - right shoulder, 5 - left shoulder, 11 - left hip. And each keypoint is represented with two numbers - x and y coordinates.
Now, when the bends are defined, there should be a way to find an angle between two lines of one bend. To do that, I change the basis to another one with point $(0, 0)$ being at the repeated keypoint. For example, for the case of $((6, 5), (5, 11))$ the new basis zero would be at the coordinates of keypoint 5.
There are two bends in comparison. One from the target image (how the exercise *should be* performed) and another one is from the original image (how it *is* performed). I find an angle for each bend and compare the two. If the numbers are close to each other, this bend gets a high accuracy. Overall, I have 12 bends, and the visualization is in the code.
After I know all 12 accuracies (one per bend), I take an average and return it as the whole performance accuracy. Each bend central point has its own accuracy number attached to it. The examples are Fig. 3 and Fig. 4.
Fig 3. is a success because the model correctly detected both humans and then calculated reasonable accuracy. Indeed, legs are wrists are not exactly the same. Fig 4. shows weakness of the algorithm, mainly because of the model I chose.

2) Videos: As was mentioned before, video is just a sequence of frames, each of which is an image, so the idea stays the same.

V. Further work

There are several unresolved issues with the currect state of the project:
1. The model has to be substituted with a better one. The candidates are OpenPose and AlphaPose. Nevertheless, they have different interface, so the code will need a readjustment. It would be enough to change the lists that represent keypoints and bends, everything else can stay as is.
2. It would be perfect to have it working in real-time for exercising but it requires computational resources I do not have at the moment.
3. The algorithm is sensitive to viewpoints. Since it heavily relies on bends angles, it will not handle a situation when target and original images are mirrored. Solving this requires some either preprocessing or more advanced computer vision strategies.
4. Instead of processing one frame at a time I could use sequence-to-sequence RNN. It could also make real-time processing easier and faster. Moreover, an RNN at time t can remember things that happened before t which could improve the accuracy skeleton detection.

Fig. 4. (a) Target image (b) Skeleton of target (c) Original human (d) Skeleton of original (e) Comparison

REFERENCES

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.
[2] Xiao Xiao. Human pose estimation via improved resnet-50. *IET Conference Proceedings*, pages 24 (5 .)–24 (5 .)(1), January 2017.
[3] Chen Z. Zhang, J. and D. Tao. Towards high performance human keypoint detection. volume 129, June 2019.
[4] Riza Alp Guler and Iasonas Kokkinos. Holopose: Holistic 3d human reconstruction in-the-wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
[5] Haque S.M.E. Paul, M. and S. Chakraborty. Human detection in surveillance videos and its applications - a review. November 2013.
[6] Sang-Ki Ko, Jae Gi Son, and Hyedong Jung. Sign language recognition with recurrent neural network using human keypoint detection. In *Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems*, RACS '18, page 326–328, New York, NY, USA, 2018. Association for Computing Machinery.
[7] Sang-Ki Ko, Chang Jo Kim, Hyedong Jung, and Choongsang Cho. Neural sign language translation based on human keypoint estimation. *Applied Sciences*, 9(13), 2019.
[8] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021.
[9] Sen Qiao, Yilin Wang, and Jian Li. Real-time human gesture grading based on openpose. In *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–6, 2017.
[10] Ismail Oztel. Human detection system using different depths of the resnet-50 in faster r-cnn. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–5, 2020.
[11] Bingyi Li, Jiaqi Zou, Luyao Wang, Xiangyuan Li, Yue Li, Rongjia Lei, and Songlin Sun. The overview of multi-person pose estimation method. In Songlin Sun, Meixia Fu, and Lexi Xu, editors, *Signal and Information Processing, Networking and Computers*, pages 600–607, Singapore, 2019. Springer Singapore.