COMP6211 Biometrics Coursework

Name: Haoyu Wang ID: 31623875

***Abstract*—This Coursework ask us to make a body gait and pose recognition method. We use body pose detect to generate human skeleton, compare between different skeleton to distinguish different people with only one image per people. And finally get a pretty good recognition performance.**

I. Introduction

By reviewing the gait image data [1] provided and the assignment requirements, we can draw some conclusions. First, we only have one training set image per object as a reference. This increases the influence of randomness, thus increasing the probability of error and the difficulty of judgment. Second, we are asked to recognise people by images of their body shape rather than recognise them by facial recognition or any other way. So we can't use existing models such as facial recognition models and image similarity comparison models. Finally, we can see that the images in the test set and the training set are different. The same person will wear different clothes. Therefore, any color/shape related recognition technology will not be used.

II. Model To Use

Based on the above questions and requirements. One conclusion we can draw is that we must use recognition of the skeletal structure and standing posture of the human body. Because these factors are not obviously affected by clothing, color, lighting, film angle, face looking, accessories, etc.

In the industry, among the existing pose and gait recognition models, the best results are as follows: Open Pose [2] from Carnegie Mellon University, AlphaPose [3] from Shanghai Jiaotong University, Deep Pose from Google, and Carnegie Mellon University's The convolutional pose machines.

In the above implementation method, first we exclude Deep Pose and convolutional pose machines. Because Deep Pose does not have open source code that can be used directly, although the Pose estimation can be used in TensorFlow Lite from Google. But the accuracy is low. As to the CMU's convolutional Pose machines. Its effect and popularity are not as good as Open Pose, which also comes from CMU. In the end, due to the poor compatibility of Alpha Pose with the Windows operating system and my computer's GPU performance, I finally choose Open Pose as the extraction model of the posture and posture in the picture.

III. Feature Choose

There are three of the most used types of human pose models: skeleton-based model, contour-based, and volume-based.

Among those three method, we choose to use skeleton-based model. It consists of a set of key points to form the human skeleton structure. This structure won't get a huge effect by the different clothing.

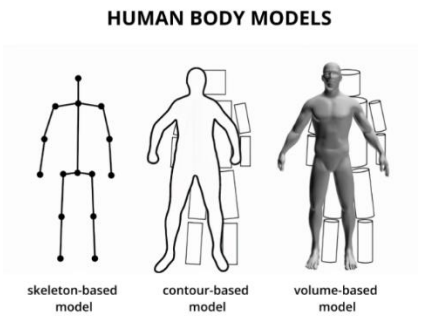

Figure I. Different Human Body Models

So, finally we choose to use Open Pose with 2D Skeleton-based model to recognize our image.

And in our test image, we have people face front or face another direction. These poses of standing will have a effect on the possibility of recognition of body's key points(joints). So, we decided to make some trade-offs in the generated data. Delete the recognition points with lower accuracy due to the influence of the stance. This step will be expanded in detail in the next section.

So how do we compare different skeletons? According to the logic of the human brain when comparing different pictures, we will first put the two pictures together virtually in the brain, and then look for the differences between them. We want the computer to implement the same logic. We want to overlap the two skeletons Together, then observe the position difference between the various joints(or key points), so that the computer can distinguish different skeletons.

According to the data provided by Open Pose, we can directly use the position of point 0 and use this point as the reference point (coordinate axis origin). Find a relative position relative to point 0 for all the remaining points.

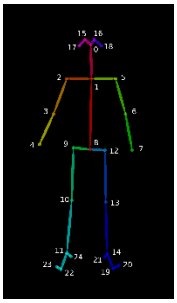

Figure II. Key Points Number By Open Pose

The reason for this is that we want to remove the influence of the camera angle. Because in the picture set, although the angle, direction, and lighting conditions may be different. But the distance between people and the camera is always constant. Therefore, the size of the human skeleton is won't change accordingly. So we don't need to scale the bone size.

After that, we can compare the skeleton generated by the pictures in the test set with all the skeletons in the training set, and finally get the prediction result.

IV. Implementation Approach

A. *Generate Data and Image*

After we decide which model and feature to use. We need to extract them.

Although Open Pose can be run by GPU or CPU. According to the prediction results, the GPU operation of my computer is only 0.2% faster than the CPU operation. Therefore, I did not use CUDA, which significantly reduced the time cost of restoring the Open Pose project.

In the end, I found a version on the Internet that only runs on the CPU and perform skeleton recognition on our image data.

I found that my computer is very fast in recognizing a single picture, but we have too many pictures, and I want to save time. Therefore, each picture has been compressed to a certain extent to make the recognition faster.

Finally, we output the point data recognized by the model and save all pictures' info into a csv file. And draw the recognized skeleton on the original picture, and regenerate as a new picture.

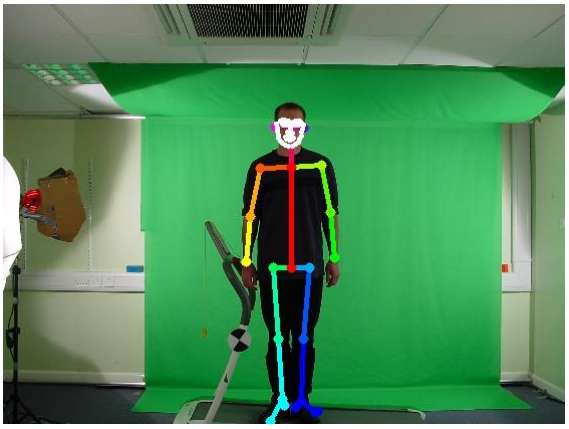

Figure III. Example Image Generated

B. *Code to get data*

Next step. We need to extract the data we want from the csv file. In this step, I wrote nine functions for Test data and Train data. These nine functions can be divided into the following categories: 1. Extract All data of a certain picture. 2. Extract the relative position data of a point in a certain picture. 3. Extract the data of a point in a picture in different poses. 4. Extract the relative position data of a certain point in all pictures.

1. Extract All data of a certain picture:

In this function, we get info from corresponding csv file. Then choose a certain image we want to use and extract all data from that.

2. Extract the relative position data of a point in a certain picture:

In this function, we extract the data of point 0, then find the relative coordinates of the remaining points. And filter out the relative coordinates of the specific point we need.

Then we construct them into a numpy.array for future calculation.

3. Extract the data of a point in a picture in different poses.

Because our people stand in different directions. And even between the test set and the training set with the same person and the same direction, will have a difference in whether or not they're holding the sign. Therefore, we need to optimize for these specific movements and orientations. For positive standing, we can boldly use all points except the hand that holds the sign (i.e. points 4 and 7). But we noticed that some people's feet were out of the picture. And even within the range of images, the accuracy of foot recognition is relatively low. So we don't include points 19 to 24 in the skeleton.

Similarly, people standing on their side need to filter out blocked spots and those with low accuracy. Including number 2,3,4,8,9,10,11,12,13,14,15,17,19,20,21,22,23,24 point.

This function is actually based on the last one and finally grow from it. At the end, they will combine together and separate based on the facing side.

4. Extract the relative position data of a certain point in all pictures:

This is actually an abandoned function, but I want to talk about it a little bit because it represent another way idea to implement the skeleton recognition.

The logic we use in the current approach is to compare the entire skeleton data of the test image with the skeleton data of all the training sets. But we can actually go the other way, compare each point in the skeleton to this point in the skeleton of all the training sets. The number of predictions will be the number of midpoints in the bone. And then filter the one that predicted the same number of times the most. Then we get our final result.

But, this method will increase the error of our result. Because then we're going to have a lot of different predictions for each skeleton. We even have cases where both outcomes are predicted the same number of times. Therefore, we should compare the whole skeleton as the human brain does, that is, calculate the cost of all the data in a skeleton.

C. *Compare Skeleton and make Prediction*

At first I want to use the KNN method. But since each type of training set has only one training data. This means that we have 88 types of data, each of which has only one number. So, There is no need and no way to use the KNN algorithm.

So, what we actually need to do is compare the relative distances between each individual piece of data in the test

skeleton data and all of the same pieces of data in the training set skeleton data. This step is repeated until all the points in a skeleton have been calculated from all the corresponding points in train set skeletons. And then we sum the cost(the Euclidean distance) of all the points in a skeleton. Finally, the degree of difference between each training set skeleton and the current test set skeleton is obtained.

Then we can easily make predict that the skeleton has the less cost with current skeleton will be the final result.

In order to better achieve this function, I also divided the image into two categories, the image standing forward and the image standing side. This allows the skeleton to be calculated only for the skeleton data in the same direction.

Then, we forecast all the test set's skeleton data and output the forecast results. At the same time, I matched the real results corresponding to each test data and outputted them together. Finally, the accuracy of the whole algorithm is calculated.

V. Recognition Performance

This is the final result that this algorithm output.

```
test image index is 1   guess result is 48   real result is 48    True
test image index is 2   guess result is 47   real result is 47    True
test image index is 3   guess result is 50   real result is 50    True
test image index is 4   guess result is 49   real result is 49    True
test image index is 5   guess result is 52   real result is 52    True
test image index is 6   guess result is 53   real result is 51    False
test image index is 7   guess result is 54   real result is 54    True
test image index is 8   guess result is 53   real result is 53    True
test image index is 9   guess result is 56   real result is 56    True
test image index is 10   guess result is 5   real result is 55    False
test image index is 11   guess result is 58   real result is 58    True
test image index is 12   guess result is 73   real result is 57    False
test image index is 13   guess result is 20   real result is 60    False
test image index is 14   guess result is 33   real result is 59    False
test image index is 15   guess result is 48   real result is 62    False
test image index is 16   guess result is 61   real result is 61    True
test image index is 17   guess result is 68   real result is 64    False
test image index is 18   guess result is 37   real result is 63    False
test image index is 19   guess result is 26   real result is 66    False
test image index is 20   guess result is 27   real result is 65    False
test image index is 21   guess result is 52   real result is 88    False
test image index is 22   guess result is 87   real result is 87    True
50.0 %
```

Figure IV. Final Predict Result

I think it's fair to say that this method achieve a pretty solid result in predict different skeletons. Among all the 22 test images, we successfully predict 11 of them. And get the Correct Classification Rates at exactly 50%. The randomly choose predict Correct Classification Rates would be 4.5%.

We can take the first one as an example to see the different cost between skeletons, and see how much similarity of each skeleton has compared with the test skeleton.

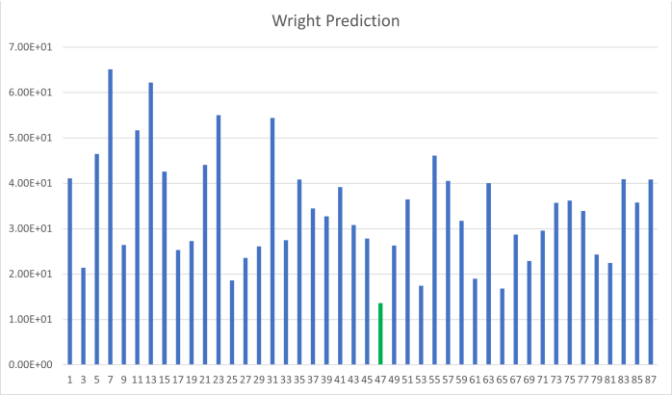

Figure V. Right Prediction Cost

We can see the right skeleton was selected with least error cost.

But obviously, why there are still half of the prediction is wrong? Let's take the test image index 6 as an example to better illustrate this question.

The people in this image is the same as image 51, but the prediction is image 53. Let's take a look at those three images with skeletons on them.

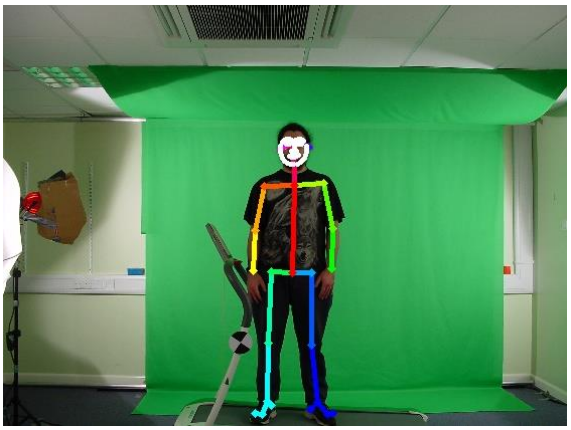

Figure VI. Wrong Prediction: Test Image

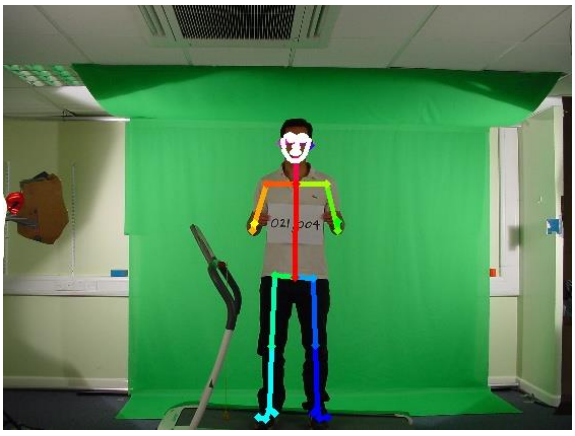

Figure VII. Wrong Prediction: Prediction Image

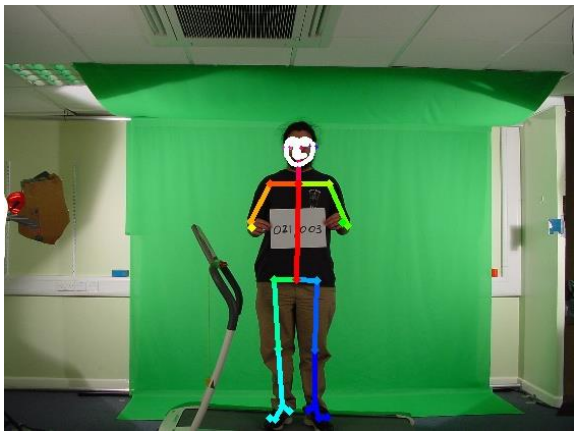

Figure VIII. WRONG PREDICTION: RIGHT IMAGE

If we just know the skeleton data, or skeleton image. I thought we will definitely make the same predictions as the computer makes. Because as you can see, This person in the test set and in the training set, the spacing between their legs was very different. At the same time, our predictions were very close to the test data, both in terms of height and posture.

We can also have a look at the cost result:

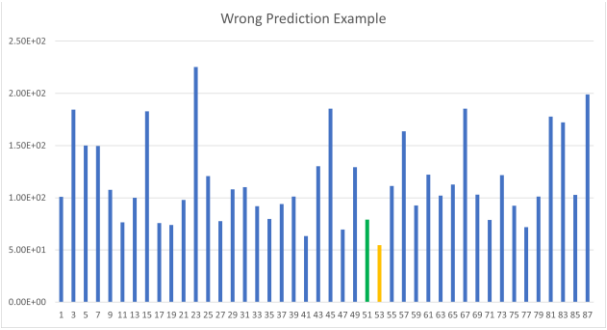

Figure IX. WRONG PREDICTION: COST FIGURE

As the figure shows, The right result still get a relatively small cost, but the body shape of the prediction's is much closer with the test one.

And this actually tells us a problem. A training set with only one sample is too small. This greatly increases the random probability error.

VI. METHOD ANALYSIS

There are several indisputable advantages to using this approach, many of which have been mentioned previously:

1. Our posture and bone recognition can ignore the influence of clothing, human face, illumination and shooting Angle. Identify a person directly from the body and gait levels.

2. Compared with the comparison of a single data point, this method can effectively reduce errors.

3. Although only one sample means a large random error, it does mean that only one sample is needed for each type of data. This greatly reduces the amount of work required to enter new classes.

Again, there are some problems with this approach. Such as:

1. There is only one sample, so the random error is too large. A person can make the result wrong simply by changing the position that is not used often.

2. A very important point of our method is that it ignores the judgment accuracy of each point position. Open Pose can provide us with this data to better judge the credibility of each point and change its importance ratio in the overall skeleton data. Therefore, we can theoretically add the weight information of each point accordingly. However, in the current method, we manually filter the points with accuracy rate of more than 70% and they are usually the point we choose to remain in the method.

Our future work is not just to solve these two problems. Also consider the impact of addressing changes in location.

Because at the moment, everyone is the same distance from the camera, which means that if two people have the same skeleton shape, we can use the size of the skeleton instead of the height to make a further distinction. But if the two people are in different positions, it means that factors such as height can no longer be used as a predictor.

At the same time, another step we need to take in the future is to take into account the body types of different people. In another word. the difference between fat and thin. Because our current posture recognition only identifies the skeleton. There was no way to get fat or thin data. If we have this data, we can differentiate even people with similar skeletons by being fat or thin.

REFERENCES

[1] J. G. M. N. M. A. C. J. SHUTLER, "ON A LARGE SEQUENCE-BASED HUMAN GAIT DATABASE," IN APPLICATIONS AND SCIENCE IN SOFT COMPUTING, PP. PP. 339-346, 2004.

[2] Z. H. G. S. T. W. S. A. S. Y. CAO, "OPENPOSE: REALTIME MULTI-PERSON 2D POSE ESTIMATION USING PART AFFINITY FIELDS," IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, PP. 43(1), PP.172-186, 2019.

[3] H. X. S. T. Y. A. L. C. FANG, "RMPE: REGIONAL MULTI-PERSON POSE ESTIMATION," IN PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION , PP. PP. 2334-2343, 2017.