Human Pose Estimation and Matching

Jiawen Ou Hongning Shangguan Hao Yuan

The University of New South Wales

Abstract

Referring to Part Affinity Fields (PAFs), human parts can be presented with linked key points that detected by VGG network. Once all parts of one person in image or video are correctly presented, an angular comparison of this person to input model would tell if this person is doing a specific pose that we want to detect. With COCO 2016 key points dataset [1], this approach provides a good detect rate. Although matching if two people are doing the same pose is somehow a subjective judge, yet our algorithm gives over 93.33% match that we satisfied.

1. Introduction

In this project, we are going to estimate a 2D multi-person pose in an image or a video using deep neural networks (DNNs) by OpenCV and then design an algorithm to detect the similarity of an image model with the input image. Human pose estimation is used to detect the position and orientation of a given human model. And it recently becomes a more and more important task in computer vision which is widely used in broad range of scientific and commercial aspects including gaming, human-robot integration, sports performance analysis. More specific, human pose estimation could track and recognize human activity for suspicious behaviour like burglary, and thus people could minimize their losses and improve safety. It could also use in a AR/VR game by having the camera recognize human body language to interact with others. Although recent approaches have dealt with several problems like identify multi-person on an image, it may not efficiently identify the pose of each people on the given image.

Multi-person pose estimation has recently improved by using large-scale datasets [6] and deep convolutional neural networks [5]. Although these approaches have dealt with several challenges, some problems like person to person occlusions and partial visible of person remain far from being solved. Current DNNs could explicitly train the data to identify all the Keypoints on a given image, but it often output incorrect association of body parts when different people with merges of body joints or missing joints occur. This challenge which is to correctly localize each body parts of all people in the scene is the main task of our project.

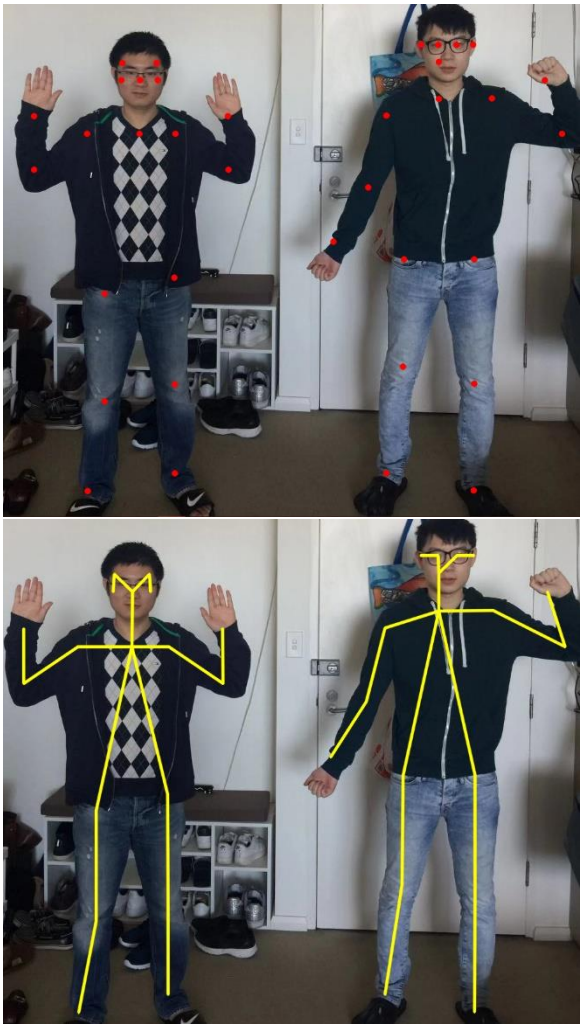

Figure 1. TOP: Red points illustrates the Keypoints of all person joints, each person has 18 Keypoints without occulation. Bottom: Example of Multip-person pose estimation. The skeleton in yellow shows that the body parts belong to the same person.

In this work, we first focus on identifying the location of 18 body parts which includes nose, neck, right and left eyes, ears, shoulders, elbows, wrists, hips, knees, ankles on a single person. Second, we improve our approach by working on multi-person pose estimation. In this step, we need

to find all the Keypoints on the input image, and then determine which set of Keypoints belong to a same person. Last step is to determine the similarity of the given model and input image.

We evaluate our approach on comparing two different human poses on both image and video. The output image and video show our approach could identify similar human pose when person is not in the side position.

2. Methodology

2.1. Problem Decomposition

To match two poses, key points of human parts need to be pointed out. Once we have all key points that we interested in, since an image of a video frame may include more than one person, identifying which parts belong to which person is one of problems that we faced. After identified all parts to all people in a frame, an algorithm of getting similarity of two poses is necessary. Therefore, we decomposed the whole project into three sub goals: find key points interested, identify parts of people and find a way to calculate similarity of two models.

2.2. Literature Survey

To find key points in a frame, a multi-stage CNN is needed. Following an architecture of the two-branch multi-stage CNN by Cao [2] (Fig. 2), a good prediction of confidence map can be presented. Each stage in the first branch predicts confidence maps S^t, and each stage in the second branch predicts PAFs L^t. After each stage, the predictions from the two branches are concatenated for next stage.

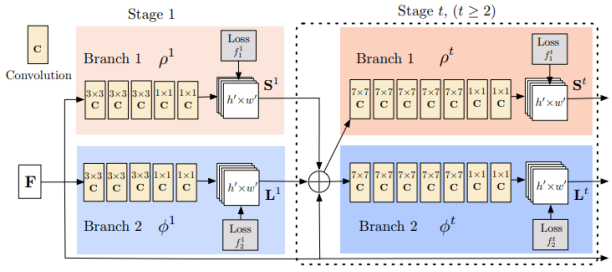

Figure 2. Network Architecture of two-branch CNN.

Refer to Wei et al. [3], refining the predictions over successive stages, $t \in \{1, \ldots, T\}$, with intermediate supervision at each stage. Input image will be analyzed by a VGG convolution network [4] first, providing confidence maps S^t and PAFs L^t that defined as,

$$\mathbf{S}^t = \rho^t(\mathbf{F}, \mathbf{S}^{t-1}, \mathbf{L}^{t-1}), \ \forall t \geq 2, \quad (1)$$

$$\mathbf{L}^t = \phi^t(\mathbf{F}, \mathbf{S}^{t-1}, \mathbf{L}^{t-1}), \ \forall t \geq 2, \quad (2)$$

where ρ t and φ t are the CNNs for inference at Stage t.

At end of each stage, loss functions spatially feedback to network indicating confidence maps S^t and PAFs L^t.

$$f_{\mathbf{S}}^t = \sum_{j=1}^{J} \sum_{\mathbf{p}} \mathbf{W}(\mathbf{p}) \cdot \|\mathbf{S}_j^t(\mathbf{p}) - \mathbf{S}_j^*(\mathbf{p})\|_2^2, \quad (3)$$

$$f_{\mathbf{L}}^t = \sum_{c=1}^{C} \sum_{\mathbf{p}} \mathbf{W}(\mathbf{p}) \cdot \|\mathbf{L}_c^t(\mathbf{p}) - \mathbf{L}_c^*(\mathbf{p})\|_2^2, \quad (4)$$

Received key points from confidence maps generated, applying vector field to link up each part of a person. The part affinity is a 2D vector field for each limb, for each pixel in the area belonging to a limb, a 2D vector encodes the direction that points from one part of the limb to the other. Each type of limb has a corresponding affinity field joining its two associated body parts (Fig. 3).

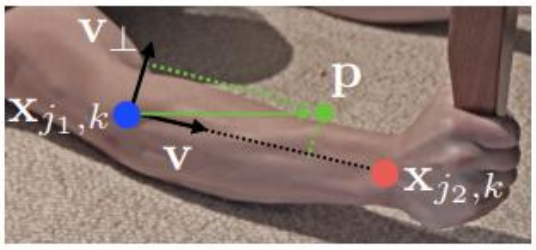

Figure 3. Vector Field

Vector field defined by PAFs gives a value indicates unit vector that points form j_1 to j_2.

$$L_{c,k}^*(p) = \begin{cases} v & if\ p\ on\ limb\ c,k \\ 0 & otherwise \end{cases} \quad (5)$$

Where v = (xj2, k − xj1, k)/||xj2, k −xj1, k|| is the unit vector in the direction of the limb.

Confidence of detect candidate limb calculated by integral of all PAFs in candidate key points:

$$E = \int_{u=0}^{u=1} \mathbf{L}_c(\mathbf{p}(u)) \cdot \frac{\mathbf{d}_{j_2} - \mathbf{d}_{j_1}}{||\mathbf{d}_{j_2} - \mathbf{d}_{j_1}||_2} du, \quad (6)$$

where $\boldsymbol{p}(u)$ interpolates the position of the two body parts d_{j1} and d_{j2},

$$\mathbf{p}(u) = (1-u)\mathbf{d}_{j_1} + u\mathbf{d}_{j_2}. \quad (7)$$

This confidence of vector field gives us method of matching two limbs, while we still need to find way of distinct parts of different people in the same frame and how we identify if two people are doing the same pose.

2.3. Algorithms

After steps of literature work, feed COCO datasets into VGG deep neural network, we can detect key points of a single person from an image, as sample in Fig. 4

Following PAFs vector field, we have confidence values of each key points to other points. Network output set of detected parts classified in parts, i.e. set of heads, set of necks, set of shoulders, etc.

To correctly classify parts to a person, linking key points that within shortest distance but with largest vector field confidence,

$$C(i,j) = \underset{d_{i,j}}{\operatorname{argmin}} \underset{E}{\operatorname{argmax}} E_{i,j} \quad (8)$$

Where $C(i,j)$ is the confidence of key point i and key point j belong to same person and they are two ends of a reasonable part.

Once we have all people in a frame with all their parts linked together correctly, similarity of two people could be presented.

Since the key of identifying if two people are doing the same pose is how their limbs behave, angular similarity of limbs is a good approach to detection poses. In this way distance of taking photo would not affect result of matching since length of limbs are not interested.

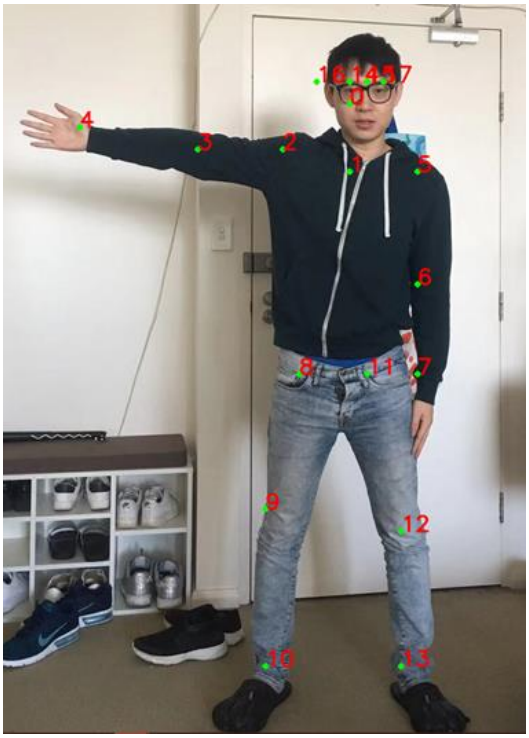

Figure 4. Keypoints Detection of each human joints. Without occupation, a person has 18 Keypoints.

A single angular comparison of two limbs are define as cosine similarity of comparing two key points tuples.

$$S_{single} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (9)$$

Where S_{single} is similarity of behavior of two limbs, A_i and B_i is vector in x and y direction of key points tuples. Since we only care about angle of limbs, we divide each vector by their second normal,

$$A_i = \frac{I_x - J_x}{\|I_x - J_x\|} \quad (10)$$

$$B_i = \frac{I_y - J_y}{\|I_y - J_y\|} \quad (11)$$

Where I and J are key points tuple detected from vector field.

Since there is a good chance that only part of a person displayed in a frame, an instance would be we can see arms behavior of a person, but legs of this person were not taking into this frame. Parts of a person that cannot be seen should not affect

similarity of comparison. Adjusted similarity would be:

$$S_{adj} = \begin{cases} S_{single} \; if \; both \; tuple \; can \; be \; seen \\ 0, m-1 \qquad\qquad\qquad otherwise \end{cases} (12)$$

Where m is total key points detected of a person.

Therefore, similarity of poses of two people are:

$$S = \frac{1}{m}\sum_{i=1}^{m} S_{adj} \qquad (13)$$

By using this similarity, if similarity above a threshold, we identify two poses as the same.

3. Implementation

The neutral network that we build will have two types of data return to us. One is the confidence map.

3.1. Confidence map

Figure 5. The green area illustrates the Confidence map for Left shoulder of multi-person. [7]

This is a map full of probability of 18 human parts (0-Nose, 1-Neck, 2-Right Shoulder, 3- Right Elbow. 4-Right Wrist, 5-Left Shoulder, 6-LeftElbow, 7-Left Wrist, 8-Right Hip, 9-RightKnee, 10-Right Ankle, 11-Left Hip, 12-Left Knee, 13- Left Ankle, 14-Right Eye, 15-Left Rye, 16-Right Ear, 17-Left Ear) and One background. Base on that data, we can resize that to our input image, and set a threshold which we use is 0.2. This means when the probability of the part in that position. By highlight the picture of right shoulder at Figure 1 we can see that the network did an excellent job. Then we get all the points that pass the threshold and keep them in NumPy array for feature use.

3.2. Part Affinity Map:

When the neutral network tries to identify various kinds of human part, it also gives back the vector of the pixel in the image, where the identity parts are. The simplest thought is we can always choose the most closet part to be one person's parts. However, that not always true, that's why we need the Affinity Map. As we know that two parts belong to one human should have same vector in first derivative region. So, based on that, we choose two body that need to be connect. We get the vector of all the two parts that we have, and iterator them. In each iterator, we put 10 points along the two points Then we compare those vectors with the one we get from the affinity map, see if they are in the same direction. We set that point as true, overall if we have 70% out of the 10 points then we know that is belong to the same human. After all the parts, be connect. We have many pair of human parts. We feed that to a function, which return us an array first layer is human and the second is position of the parts. The function is to go through all the pairs, and when two are connect sign them to one person. Then we will have Multiple human parts' position.

Figure 6. The coloured area illustrates Part Affinity map for Left Shoulder - Left wrist of multi-person. [8]

3.3. Vector Compares:

Then we get the model vector and go through each human to compare the model. By use our own create algorithm. We believe that human pose can be affect by image size, different human body. Since we want to get right compare value, angle is our best way. And the way people take picture can be different which may have some part miss, so we decide to ignore the part that miss by now. By using the formula that we decriable in the Design part, we have few poses detected correctly. Since

the time limitation we didn't do too much test case.

4. Results

We present comparative results on multi-person image and video to evaluate our approach on the task of articulated multi-person pose estimation and calculate the similarity of given model with input image.

4.1. Experimental Setup

We evaluate our method by comparing different human pose on a given model and an input image. As Figure 7 shows below on the left, the model is a human pose of a single person. First the algorithm will detect all the Keypoints of this person pose (Fig. 7 central). Then each human part will be connected by line to form a skeleton (Fig. 7 right). Next step is to repeat the step for input image with multiply persons. The results of Keypoints and skeleton human pose estimation are shown at Figure 8. Once all the keypoints are assigned to a specific person, we can compare the given model and input image by calculating the angular of each connected joint like the angle between right wrist to right elbow and right elbow to right shoulder. If the human pose is matched, a wine-red rectangle will appear with the score of similarity of the top of rectangle like Figure 8 bottom central image. The similarity of this pose has the score of 99%.

Figure 7. Left: Human pose estimation model. Central: Highlight the Keypoints of 18 human parts by red dots. Right: Link associate pair of human parts to form a skeleton-look human pose estimation.

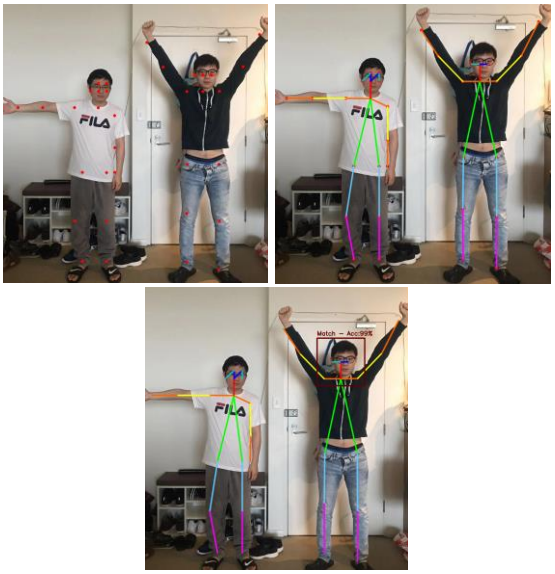

Figure 8. Top Left: Totally 36 Keypoints for 2 persons parts. Top Right: Find the association of keypoint that belongs to the same person and link these parts to form skeleton-look human pose estimation. Bottom Central: Comparison of model and input image with similarity

In order to eliminate the effect of comparing the same pose with the same person, we also use another person with different gender as model for comparison. As Figure 9 shown, the image model is a female and the image input for comparison are both males. It slightly decreases the similarity score to 96%.

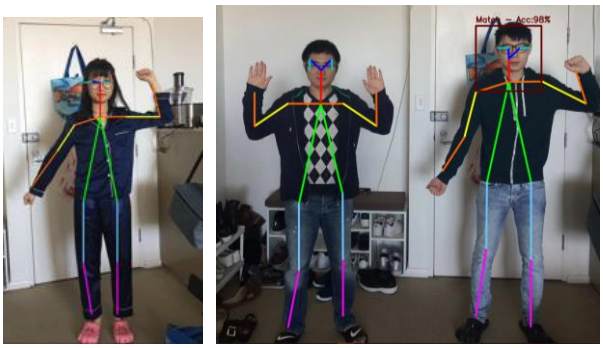

Figure 9. Left: Model image by a female. Right: Input image by two men.

We also consider the effect of invisible parts of human pose like bust shot. In this case, we use a full body shot as model where 18 keypoints are detected. And input image of a bust show where each person only has 14 keypoints are detected. The result show on Figure 10 illustrates that our method can also construct the skeleton human pose estimation and find the similar pose by ignoring the missing keypoints. The accuracy can be reach 99% in this case.

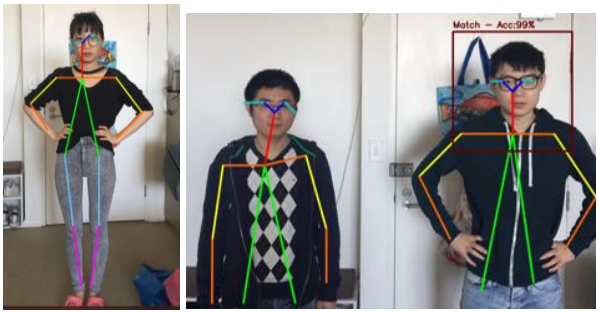

Figure 10. Left: Model image by a full body shot. Right: Input image by a bust shot.

We also evaluate our method by comparing human pose on a given image model and a short dancing video. When the human pose on video match with the image model, the wine-red rectangle will be shown on the video frame. The left figure on Figure 11 illustrate the model of human pose which will be used to compare with each frame on video.

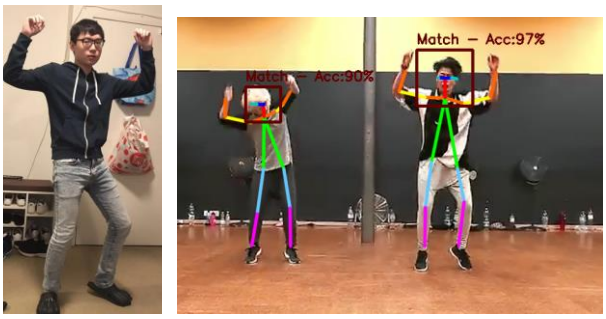

Figure 11. Left: Model image. Right: A screenshot from video where match is found. The left dancer has similarity score of 90% and the right dancer has similarity score of 97%.

5. Analysis and Conclusion

5.1. Problem Still Facing

5.1.1. The program only works in 2-D:

We only had this application in 2-D, when we have an image with human side way, it's hard for us to compare, with will give the wrong compare data.

5.1.2. The accuracy range is not good enough:

Since we compare all the parts that appear in that image. But when human standing, their neck to head, chest to feet is almost the same, that make our accuracy up to 70% which I think we should not focus on that.

5.1.3. Process speed:

Our algorithm may take 2 seconds for one frame, on our own laptop. Which is too slow for real-time (30 farm per second). It is better for us the improve the speed of our algorithm to speed up.

5.2. Feature Improvement:

5.2.1. Build 3-D Models:

We will try to build 3-D models out of few pictures from 2-D image, and do the same compression algorithm, see if we still match the pose, and give more specific result. Which can be use teach people how to work out, how to train themselves doing right Yoga, without a teacher.

5.2.2. Compare in Different Environment:

We believe that the accuracy is depend on different environment. For normal pose compare, we may want to focus on arm, for dance maybe the leg position is more important. So, we should create different compression algorithm to fit different environment that need our application. Also, the match accuracy should also fit the environment, with enough data, we can use a logits regression to train out what accuracy can be considered as match

5.2.3. Smaller Neural network:

I believe the main reason our program compute one image takes 2 second is that we build a large neural network, and the process time is too long. The improve of compute speed on device may help us a lot. Still, some trade-off can happen. If cut down the size of the Neural Network, the accuracy may drop down, but the speed can raise. In some special environment, like baby monitor, we want to know if they are doing some dangerous movement quick, so there is when we need to speed up. But for pervious environment, like teach yourself Yoga, will need more accuracy. So, a network like that may be ok.

References

[1] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár. Microsoft COCO: Common Objects in Context. ECCV 2014. 3

[2] Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. CVPR 2017. 1

[3] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In CVPR, 2016. 1, 2, 3, 6

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015. 2

[5] Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M., & Schiele, B. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In ECCV, 2016.

[6] Lin, T-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L., Microsoft coco: Common objects in context. In ECCV, 2014.

[7] Mallick, S. Deep Learning based Human Pose Estimation using OpenCV. 2018.

[8] Mallick, S. Multi-Person Pose Estimation in OpenCV using OpenPose. 2018.