International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 39, No. 1, February, 2022, pp. 119-124
Journal homepage: http://iieta.org/journals/ts

Comparative Analysis of OpenPose, PoseNet, and MoveNet Models for Pose Estimation in Mobile Devices

BeomJun Jo[1], SeongKi Kim[2*]

[1] Dept. of Game Design and Development, Sangmyung University, Seoul 03016, Korea
[2] Division of SW Convergence, Sangmyung University, Seoul 03016, Korea

Corresponding Author Email: skkim9226@smu.ac.kr

https://doi.org/10.18280/ts.390111

Received: 30 December 2021
Accepted: 26 February 2022

Keywords:
mobile devices, MoveNet, OpenPose, pose estimation, PoseNet

ABSTRACT

Pose estimation is a significant strategy that has been actively researched in various fields. For example, the strategy has been adopted for motion capture in moviemaking, and character control in video games. It can also be applied to implement the user interfaces of mobile devices through human poses. Therefore, this paper compares and analyzes four popular pose estimation models, namely, OpenPose, PoseNet, MoveNet Lightning, and MoveNet Thunder, using pre-classified images. The results show that MoveNet Lightning was the fastest, and OpenPose was the slowest among the four models. But OpenPose was the only model capable of estimating the poses of multiple persons. The accuracies of OpenPose, PoseNet, MoveNet Lightning, and MoveNet Thunder were 86.2%, 97.6%, 75.1%, and 80.6%, respectively.

1. INTRODUCTION

Pose refers to the movement of a part of the human body, such as the hands, arms, head, and face, that expresses an idea or a feeling. It is a form of non-verbal communication often used consciously, or unconsciously in daily life. The common poses include handshaking, clapping, and dancing. If these poses are taken as the user interface of devices, e.g., desktops, laptops, and smartphones, it is possible to express intentions more directly and obtain an extended experience. Apart from that, poses can be applied to virtual reality (VR), augmented reality (AR) and mixed reality (MR) to enhance the immersion and presence of users. The pose-based user interfaces boast another advantage: numerous poses can be defined, and used to interact with the devices.

Pose estimation, the key to pose recognition in user interfaces, can be utilized in various manners. In the field of movies and video games, poses are commonly estimated by motion capture. Actors are required to either wear a suit with multiple sensors or attach markers to their bodies. During the performance, computer graphics are added to the captured media, making the expression vivid and lively. In the field of video games, Nintendo released the Wii, a home video game console, in 2006 [1]. The controller of the console uses an infrared sensor, and allows operations on the screen, such as cursor movement on a personal computer (PC). Besides infrared sensors, various built-in sensors enable us to play realistic games, such as tennis, golf, and boxing. In 2010, Microsoft launched Kinect, a peripheral device of Xbox 360 [2]. Kinect relies on a camera to estimate poses and a microphone to recognize voices. As a result, users can play games without the aid of additional controllers. Although not embedded in the latest Xbox, Kinect continues to be used for military and research purposes. Similarly, the PlayStation camera [3], move controller [4], and VIVE tracker [5] are employed in video games.

In addition to motion capture and video games, pose estimation plays a significant role in smartphones. Despite the expansion of the screen and smartphones itself, and the reduction of the bezel, the current size of smartphones is not enough for various expressions. To overcome the limitation, pose estimation could be adopted to provide a convenient input method, in replacement of the traditional touch approach [6]. For example, Samsung Health [7] and Apple Fitness+ [8] could adopt pose estimation to inform users about the correctness of their poses via a mobile device. Pose estimation can also be employed to interact with AR contents [9] on smartphones.

OpenPose, PoseNet, and MoveNet are available for pose estimation on mobile devices. All these methods estimate poses through deep learning of camera inputs, eliminating the need for sensors, and therefore apply to most devices.

To the best of our knowledge, this paper is the first attempt to summarize and compare the features and prospects of OpenPose, PoseNet, and MoveNet, which are common tools for pose estimation on mobile devices. The remainder of this paper is organized as follows: Section 2 summarizes the functions of OpenPose, PoseNet, and MoveNet for a better understanding of the research; Section 3 evaluates the three models in terms of the features and performance in the same environment; Section 4 puts forward the conclusions.

2. MODELS

Poses are generally estimated by top-down or bottom-up methods. The top-down method first discovers a person in the input, and derives the pose in his/her bounding box. The bottom-up method infers the pose from all the key-points in the input, as well as the relationship between these points.

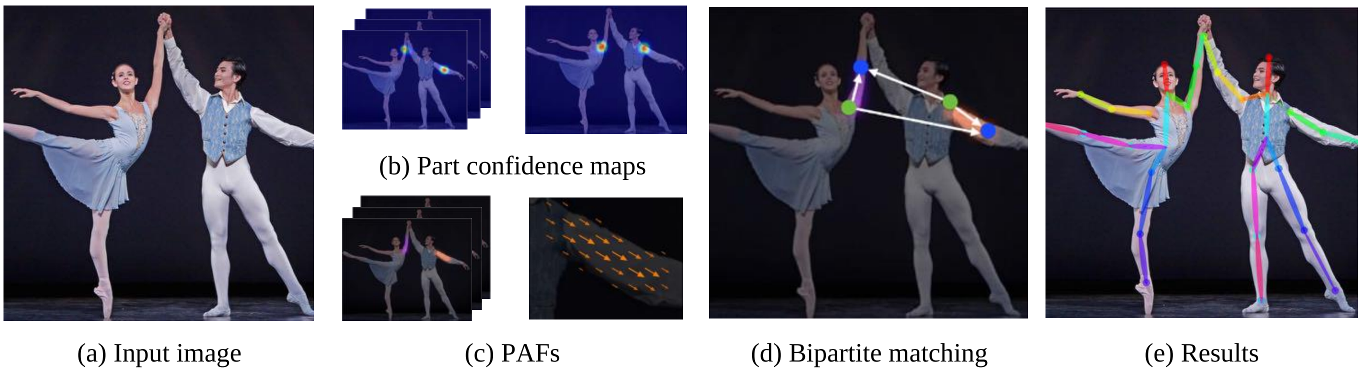

Figure 1. Overall process of OpenPose

2.1 OpenPose

In 2017, Cao et al. [10] at Carnegie Mellon University proposed OpenPose. This open source model marks the first attempt to estimate multi-person poses in the real time. It uses the bottom-up method to overcome the limitations of the top-down method. The flow of OpenPose is illustrated in Figure 1.

As shown in Figure 1 (a), two-dimensional (2D) images/videos are imported to the model. Then, a confidence map is obtained from the input. Through non-maximum suppression (NMS), the candidates for the body are identified in the confidence map. After identifying an object, a bounding box is created to contain the object, and the probability of the object is set as a score. Next, the scores are sorted in descending order, and the redundant bounding boxes are removed by the following criterion: the intersection of union (IoU) is greater than the threshold, which means the two bounding boxes identify the same object. The IoU refers to the ratio of the overlapping region to the combined region. The removal of redundant bounding boxes is known as NMS. On this basis, OpenPose creates part affinity fields (PAFs), a set of flow fields representing the relationships between parts of many persons. Finally, bipartite matching is performed on the candidates using confidence maps and PAFs, producing full-body poses.

OpenPose and Alpha-Pose are still under research [11, 12]. OpenPose was initially written in C++, and recently in Python. The applicability of different versions depends on central processing unit (CPU) or Unity [13].

2.2 PoseNet

PoseNet is an open source machine learning model created by Google Creative Lab. Capable of estimating human poses in the real time [14], the model works on the recently released COCO person key-point detection dataset, which tracks the key-points of the entire body.

Single-person pose estimation uses four elements as input: an input image, an image scale factor, a horizontal flip, and an output stride. The single-person detection algorithm is faster and simpler than the multi-person detection algorithm. Besides the four elements above, three other elements are required in the input of multi-person pose estimation: the maximum number of detected poses, the threshold for pose confidence score, and the NMS radius. These additional elements improve the accuracy of multi-person pose estimation. The estimation results include pose confidence scores and key-points. Figure 2 illustrates the flow of PoseNet.

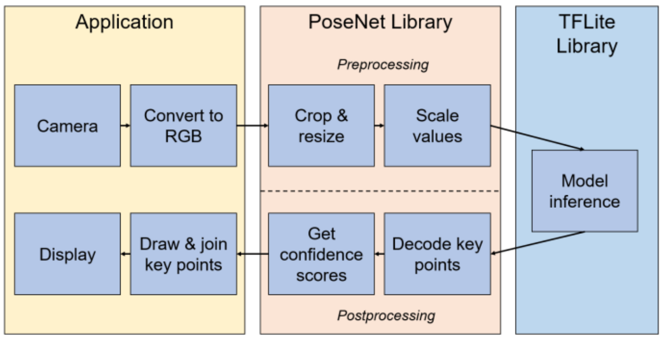

Figure 2. Flow of PoseNet [15]

As shown in Figure 2, PoseNet for mobile devices operating on Android and iOS has been developed under TensorFlow Lite [15-17]. The overall pipeline is similar to the JavaScript version. Currently, there is a face tracker called Facemesh, which combines PoseNet into a pose animator [18]. Facemesh can animate full-body characters, similar to motion capture.

2.3 MoveNet

MoveNet [19, 20] is a Google-based inference model developed by IncludeHealth, a digital health company [21]. IncludeHealth unveiled the model in 2021, and solicited help from Google to support remote treatment of patients. Similar to PoseNet, the web version of MoveNet uses TensorFlow.js, and the mobile version uses TensorFlow Lite. There are two versions of MoveNet: the performance-oriented Lightning [22], and the accuracy-oriented Thunder [23]. The two models differ in input size and depth multiplier. In terms of input, Lightning receives a video or an image of a fixed size (192×192) and three channels, and employs 1.0 depth multiplier. In contrast, Thunder receives an input of the size 256×256 and three channels, and employs 1.75 depth multiplier. The depth multiplier changes the number of channels of the input video/image, which generally adopts the red-green-blue (RGB) format. Yet feature maps can also be regarded as one channel in each layer. Meanwhile, Thunder has 1.75 times more layers for deep learning than Lightning. Hence, it performs 1.75 times more calculations.

MoveNet is a bottom-up model relies on TensorFlow object detection API and MobileNet V2 as a feature extractor. In the TensorFlow object detection API, there are multiple detection models supporting TensorFlow 1 and TensorFlow 2. MoveNet follows CenterNet [24, 25], a detection API [26]. Different from the standard anchor (bounding box)-based detection model, CenterNet takes the center point as the only anchor, and searches for and classifies objects by processing regional proposals, rather than inferring objects from the IoU value. Without requiring NMS, CenterNet recognizes the difference between objects in one stage, and exhibits a high performance.

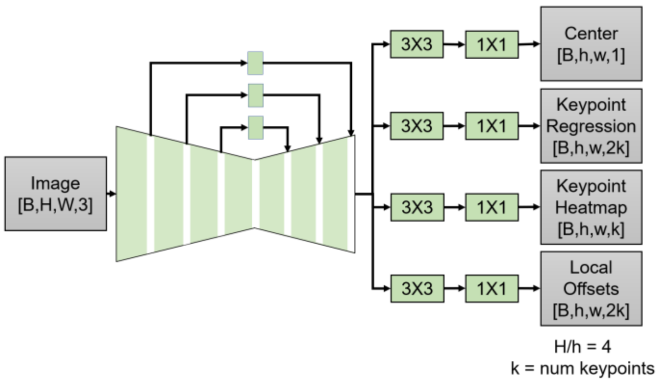

Figure 3. Flow of MoveNet [19]

As shown in Figure 3, MoveNet [19] calculates all four processes simultaneously. Once a person center heatmap is prepared to identify each person, the location with the highest score is selected. Then, a set of key-points for the person is initialized based on the key-points obtained through regression. A person is identified, when the regression fits the arrangement of prepared key-points into a person. Furthermore, each pixel is multiplied by a weight, which is inversely proportional to the distance from the regressed key-point. In this way, the key-points from the persons in the background are excluded from the computation. In the end, the set of key-points is finalized according to the maximum heatmap values in each key-point channel.

3. RESULTS AND DISCUSSION

Table 1 compares the basic specifications of OpenPose, PoseNet, and MoveNet. It can be observed that the number of key-points is the most prominent difference between these models. PoseNet provides a total of 17 key-points: 5 in the face and 12 in the body. MoveNet has the same key-points as PoseNet. OpenPose supports 137 key-points: 25 in the body, including the foot, 21 in each hand, and 70 in the face. Overall, OpenPose can track the body in greater details than the other two models. Another difference lies in the pose estimation method: PoseNet uses the top-down method, while OpenPose and MoveNet adopt the bottom-up method. The top-down method estimates a pose within a person's bounding box after detecting the person. On the contrary, the bottom-up method predicts the key-points of persons in the input, and estimates the pose from the correlation between the key-points.

Table 1. Basic specifications of OpenPose, PoseNet, and MoveNet

	OpenPose	**PoseNet**	**MoveNet**
Detection Parts	Body, Foot, Hand, Face	Body, Part of face	Body, Part of face
Detection No.	Many	One	One
Key-points	137	17	17
Operation	Real-time	Real-time	Real-time
Method	Bottom-up	Top-down	Bottom-up

These models can be applied to mobile devices in the following steps:

Step 1. The outside view is imported via the camera of the mobile phone, where the screen presents the preview of the application.
Step 2. The application captures a short moment of the preview.
Step 3. The captured contents are entered into the estimation model.
Step 4. The results of the above step are displayed on the preview screen.
Step 5. The application shuts down if the user presses the back button.
Step 6. Otherwise, Steps 2-5 are repeated.

To compare the three models fairly and accurately, the input method of comparative applications was changed, and the same scenes were delivered to OpenPose, PoseNet, and MoveNet. The images of the COCO and MPII datasets [27, 28] (Figure 4) were grouped, and used for the comparative analysis.

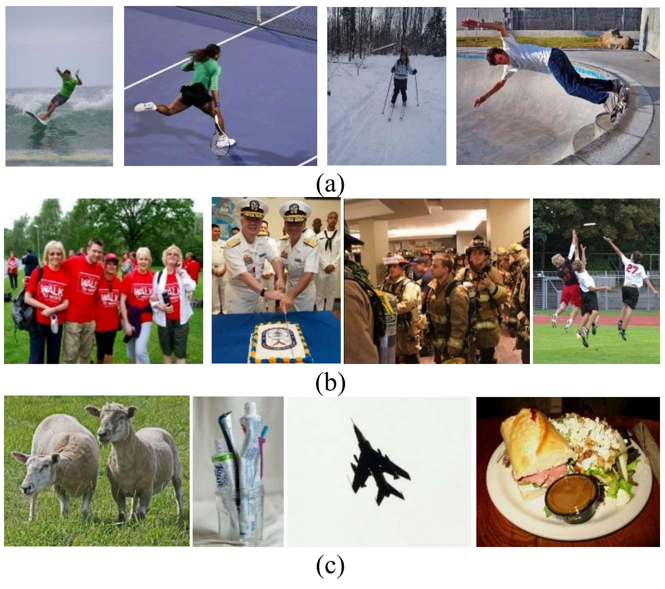

Figure 4. Examples of image groups
(a) First group with a single person (b) Second group with multiple persons (c) Third group without any person

As shown in Figure 4, the first group consists of images with only one person. This group was used to compare the performance of OpenPose, PoseNet, and MoveNet.

The second group consists of images with multiple persons. Since the mobile versions of PoseNet and MoveNet only apply to single-person pose estimation, the two models cannot be compared accurately with OpenPose on the second group of images.

The third group comprises images, which contain no person, but include elements like animals, dolls, food, and natural environment. This group was adopted to verify whether the three models can estimate poses, when there is no person in the image, and check if their performance would be affected.

Each of the three groups includes 1,000 different images. The performance of each model was measured by:

$$T_r^g = \sum_{k=0}^{n-1} E(I_k^g) \qquad (1)$$

where, g is the group number; T_r^g is the total time to estimate all images in each run; n is the number of images; $E(I_k^g)$ is the time to estimate an image I_k^g within group g. Formula (1) represents the time to estimate the poses in 1,000 images within group g. The T_r^g value of each model was measured 10 times.

The sum S^g of T_r^g can be expressed as:

$$S^g = \sum_{r=0}^{9} T_r^g \tag{2}$$

The standard deviation SD^g of T_r^g can be expressed as:

$$SD^g = \sqrt{\frac{\sum(T_r^g - \overline{T^g})}{10}} \tag{3}$$

Table 2 summarized the results of all three models. Note that all results were rounded to the third decimal place.

Table 2. Total time (unit: s) to estimate 1,000 images within a group and its standard deviation

Group	**OpenPose**		**PoseNet**	
	Total time	**SD**	**Total time**	**SD**
1st	649.368	13.757	78.289	0.252
2nd	643.384	21.915	77.732	0.630
3rd	637.856	16.960	69.675	0.464

Group	**MoveNet Lightning**		**MoveNet Thunder**	
	Total time	**SD**	**Total time**	**SD**
1st	55.362	0.652	141.435	8.186
2nd	56.122	1.086	141.005	3.832
3rd	48.891	1.746	137.482	0.717

As shown in Table 2, all models achieved the best performance on the images of the third group, for the images do not contain any person. PoseNet had the smallest standard deviation, i.e., the smallest performance fluctuation between runs.

The performance required for each image was measured by:

$$A^M = \frac{1}{n^1+n^2+n^3}(T^1 + T^2 + T^3) \tag{4}$$

where, M is the model used to estimate the pose; n^i is the number of images within group i; T^i is the total time to estimate all images within group i; A^M is the mean time to estimate poses in all images within all groups by model M. The A^M was measured ten times, and the average was taken as the final result (Table 3).

As shown in Table 3, the models can be ranked in descending order of performance as MoveNet Lightning, PoseNet, MoveNet Thunder, and OpenPose. This is because OpenPose is a multi-person pose estimation model, and the bottom-up method is slower than the top-down method.

Table 3. Mean time to estimate 1,000 images (unit: s)

Models	**OpenPose**	**PoseNet**	**MoveNet Lightning**	**MoveNet Thunder**
Average time	643.536	75.232	53.458	139.974

Aside from performance, the accuracy of each model was measured by:

$$C^g = \frac{e^g}{n^g} \tag{5}$$

where, g is the group number; n^g is the number of images within group g; e^g is the number of the estimated correctly images within group g. The pose estimation accuracies of the models on the first and third groups are recorded in Table 4. The model accuracies on the second group are not shown, for PoseNet and MoveNet cannot estimate poses in images with multiple persons. All models, except for OpenPose, only output zeros and ones as the result of human identification, although the image groups have different objects and treatments. Therefore, the results between the first and third groups can be shared, and have meaningful results.

Table 4. Pose estimation accuracies on images within the first and third groups

Model	**OpenPose**	**PoseNet**	**MoveNet Lightning**	**MoveNet Thunder**
Group 1	78.5%	96.7%	78.7%	79.5%
Group 3	93.8%	98.4%	71.5%	81.6%
Average	86.2%	97.6%	75.1%	80.6%

As shown in Table 4, PoseNet had the highest accuracy, followed by OpenPose, and then MoveNet Thunder/ Lightning.

Next, the multi-person pose estimation effect of OpenPose was verified. The relevant results are displayed in Figure 5.

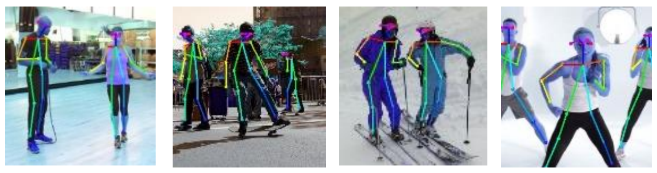

Figure 5. Results of OpenPose

As shown in Figure 5, OpenPose successfully identified the multiple persons, and accurately estimated the poses within each image. But PoseNet and MoveNet failed to find such persons.

A part of the execution results is reported in Figure 6, where the columns are in the order of original, OpenPose, PoseNet, and MoveNet Lightning/ Thunder, the first and second rows are the images, where the poses are successfully estimated, and the third row is the results of the third group. As shown in Figure 6, PoseNet and MoveNet Thunder outputted human poses, although no pose is included in the third group.

Figure 7 shows the images in the first group, where the poses are incorrectly estimated. The results of PoseNet are presented in subgraphs (a) and (b): In both cases, PoseNet and MoveNet successfully inferred the poses, but OpenPose failed. The results of OpenPose are presented in subgraphs (c) and (d): In the case (c), both MoveNet models successfully inferred the pose, while OpenPose estimated only a part of the arm, and PoseNet failed to estimate the pose, as it could not recognize the person due to rotation; In the case (d), OpenPose recognized the doll as a human, and estimated the doll poses in addition to human poses, yet PoseNet and MoveNet estimated only human poses.

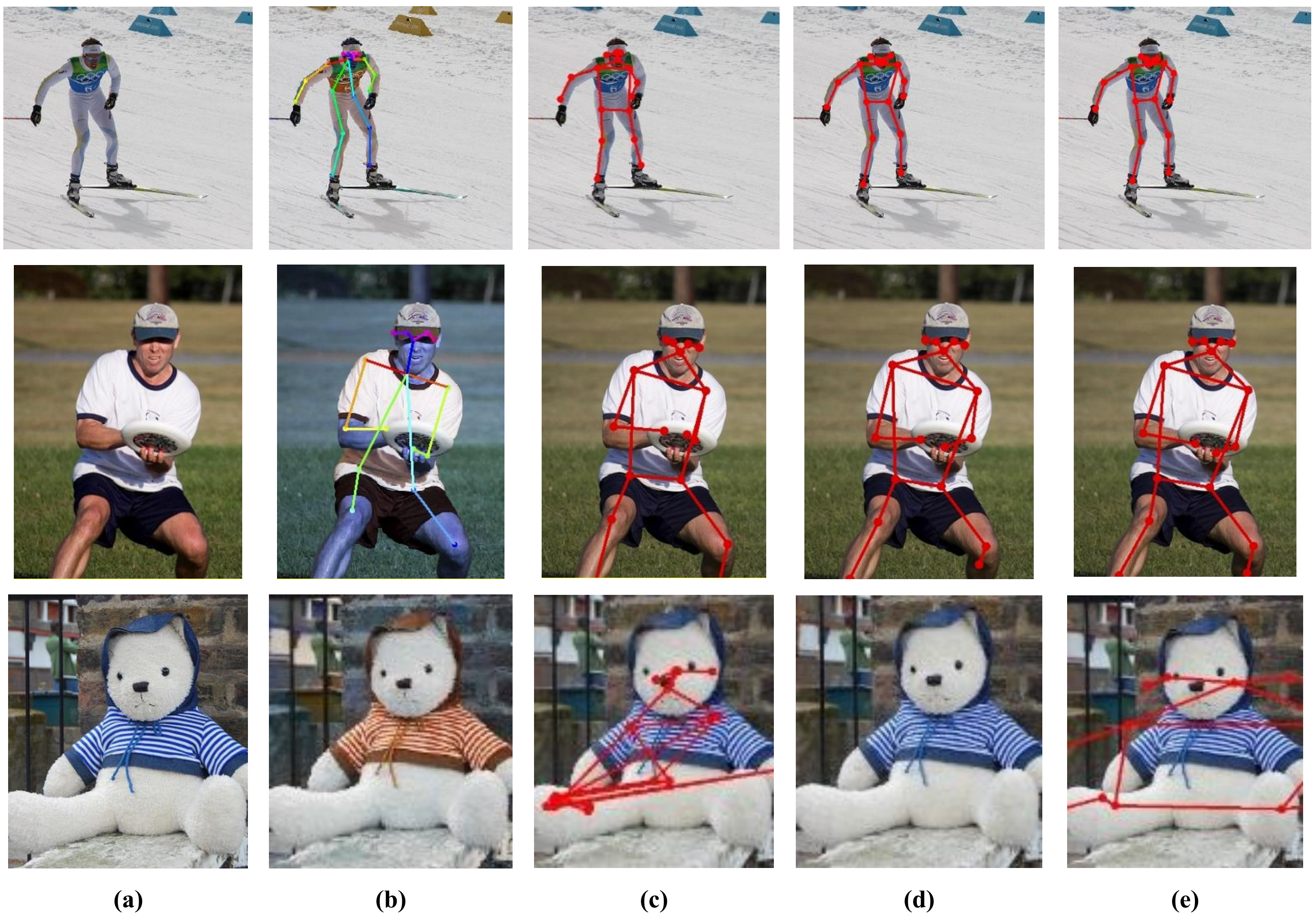

Figure 6. A part of the execution results (a) Original image (b) Results of OpenPose (c) Results of PoseNet (d) Results of y MoveNet Lightning (e) Results of MoveNet Thunder

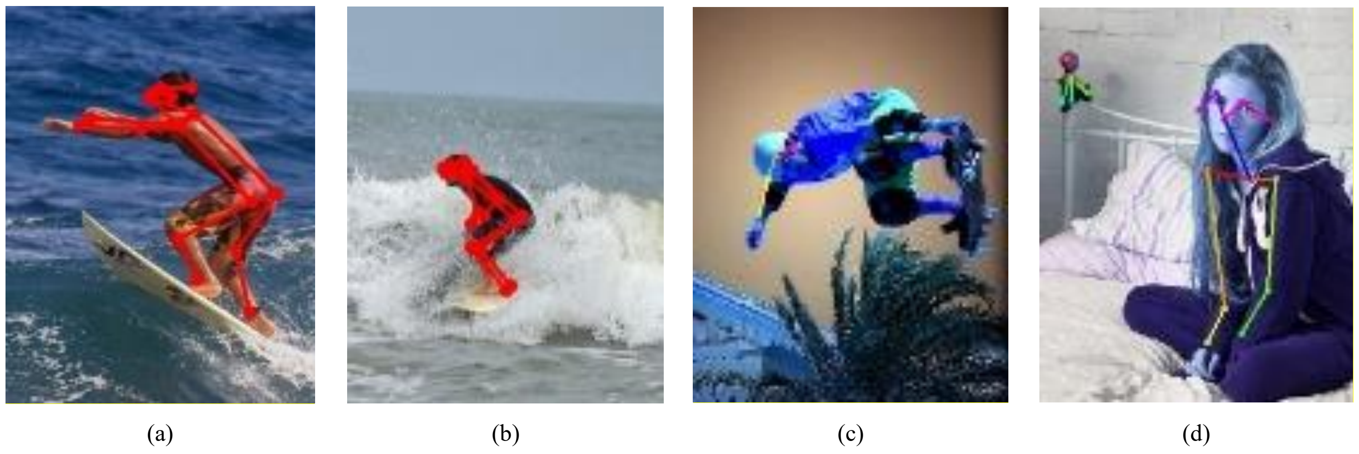

Figure 7. Incorrect outputs on the first group (a)(b) Results of PoseNet (c)(d) Results of OpenPose

4. CONCLUSION

The pose estimation on mobile devices is crucial, because it supports manipulations that cannot be performed with only a touch interface. For example, smartphones with Android 4.3 Jelly Bean or newer versions have an additional function called Smart Stay, which tracks user eyes on the phone. Another emerging example is the function of Smart Scrolling, which captures user eyes and phone tilt. Moreover, the Motion Sense function of Pixel 4 enables media control without touching the phone, even if the screen is turned off. In addition, many other functions can be realized through using pose estimation. As a result, OpenPose, PoseNet, and MoveNet boast a great potential in mobile device applications. However, their advantages and disadvantages have not yet been investigated.

This paper compares the mobile version of OpenPose, PoseNet, and MoveNet in terms of features and performance, using pre-classified images. All models could use normal cameras without any specific sensors. The comparison reveals that MoveNet Lightning was the fastest model. OpenPose was 12 times slower than MoveNet, but it could estimate the poses of multiple persons. Meanwhile, PoseNet achieved the highest accuracy. The limitation of our research is that the model performance was merely compared on images, and the results of the camera input were not measured, which will be considered in future research. Also, MoveNet MultiPose was recently developed and was unable to be compared in this paper. We consider it as the next research.

REFERENCES

[1] Wikipedia. Wii. https://en.wikipedia.org/wiki/Wii, accessed on Feb. 17, 2022.

[2] Microsoft. Azure Kinect DK. https://azure.microsoft.com/en-us/services/kinect-dk/, accessed on Feb. 17, 2022.

[3] SONY Playstation. PlayStation Camera. https://www.playstation.com/en-us/accessories/playstation-camera/, accessed on Feb. 17, 2022.

[4] SONY Playstation. Playstation Move Motion Controller. https://www.playstation.com/en-us/accessories/playstation-move-motion-controller/, accessed on Feb. 17, 2022.

[5] Vive. VIVE tracker (3.0). https://www.vive.com/us/accessory/tracker3/, accessed on Feb. 17, 2022.

[6] Samsung Newsroom. [Infographic] A rich life with Samsung Galaxy S4. https://news.samsung.com/kr/4306, accessed on Feb. 17, 2022.

[7] Youtube. How to use Samsung Health with Neo QLED | Samsung. https://www.youtube.com/watch?v=7i3q1tw7Et0, accessed on Feb. 17, 2022.

[8] Apple. Apple fitness+ - Apple. https://www.apple.com/apple-fitness-plus/, accessed on Feb. 17, 2022.

[9] Youtube. Minecraft Earth: Official reveal trailer. https://youtu.be/AQEizp-VrVU, accessed on Feb. 17, 2022.

[10] Cao, Z., Simon, T., Wei, S.E., Sheikh, Y. (2017). Realtime multi-person 2D pose estimation using part affinity fields. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1302-1310. http://dx.doi.org/10.1109/CVPR.2017.143

[11] Github. CMU-Perceptual-Computing-Lab/openpose. https://github.com/CMU-Perceptual-Computing-Lab/openpose, accessed on Feb. 17, 2022.

[12] Github. MVIG-SJTU/AlphaPose. https://github.com/MVIG-SJTU/AlphaPose, accessed on Feb. 17, 2022.

[13] Github. CMU-Perceptual-Computing-Lab/openpose_unity_plugin. https://github.com/CMU-Perceptual-Computing-Lab/openpose_unity_plugin, accessed on Feb. 17, 2022.

[14] TensorFlow. Real-time human pose estimation in the browser with TensorFlow.Js. https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5, accessed on Feb. 17, 2022.

[15] TensorFlow. Pose estimation. https://www.tensorflow.org/lite/examples/pose_estimation/overview, accessed on Feb. 17, 2022.

[16] Github. TensorFlow lite pose estimation Android demo. https://github.com/tensorflow/examples/tree/master/lite/examples/pose_estimation/android, accessed on Feb. 17, 2022.

[17] TensorFlow. Track human poses in real-time on Android with TensorFlow Lite. https://medium.com/tensorflow/track-human-poses-in-real-time-on-android-with-tensorflow-lite-e66d0f3e6f9e, accessed on Feb. 17, 2022.

[18] Tensorflow. Pose Animator - An open source tool to bring SVG characters to life in the browser via motion capture. https://blog.tensorflow.org/2020/05/pose-animator-open-source-tool-to-bring-svg-characters-to-life.html, accessed on Feb. 17, 2022.

[19] TensorFlow. MoveNet: Ultra fast and accurate pose detection model. https://www.tensorflow.org/hub/tutorials/movenet, accessed on Feb. 17, 2022.

[20] Tensorflow. Pose estimation and classification on edge devices with MoveNet and TensorFlow Lite. https://blog.tensorflow.org/2021/08/pose-estimation-and-classification-on-edge-devices-with-MoveNet-and-TensorFlow-Lite.html, accessed on Feb. 17, 2022.

[21] Tensorflow. Next-generation pose detection with MoveNet and TensorFlow.Js. https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html, accessed on Feb. 17, 2022.

[22] Tfhub.dev. movenet/singlepose/lightning. https://tfhub.dev/google/movenet/singlepose/lightning/4, accessed on Feb. 17, 2022.

[23] Tfhub.dev. movenet/singlepose/thunder. https://tfhub.dev/google/movenet/singlepose/thunder/4, accessed on Feb. 17, 2022.

[24] Zhou, X., Wang, D., Krähenbühl, P. (2019). Objects as Points. ArXiv, https://arxiv.org/abs/1904.07850.

[25] Github. xingyizhou/CenterNet. https://github.com/xingyizhou/CenterNet, accessed on Feb. 17, 2022.

[26] Github. tensorflow/models. https://github.com/tensorflow/models/tree/master/research/object_detection, accessed on Feb. 17, 2022.

[27] Cocodataset.org. COCO - common objects in context. https://cocodataset.org/, accessed on Feb. 17, 2022.

[28] Mpg.de. MPII Human Pose Database. http://human-pose.mpi-inf.mpg.de/, accessed on Feb. 17, 2022.