Grasping Hand Pose Estimation from RGB Images using Digital Human Model by Convolutional Neural Network

Kentaro INO*[1], Naoto IENAGA[1], Yuta SUGIURA[1], Hideo SAITO[1], Natsuki MIYATA[2], Mitsunori TADA[2]
[1] Keio University, Yokohama, Japan;
[2] Digital Human Research Group, Human Informatics Research Institute,
National Institute of Advanced Industrial Science and Technology, Japan

DOI: 10.15221/18.154 http://dx.doi.org/10.15221/18.154

Abstract

Recently, there has been an increase in research estimating hand poses using images. Due to the hand's high degree of freedom and self-occlusion, multi-view or depth images are often used. Our objective was to estimate hand poses specifically while grasping objects. When holding something, the hand moves in many directions. However, if the camera is too distant from the hand, it may move out of range. Widening the viewing angle, however, reduces the resolution beyond usable limits. One possible solution was developed by Kashiwagi - by setting the camera on an object, the hand's pose can be estimated regardless of its position. However, Kashiwagi's method cannot be used without estimating the fingertips' positions. Recently, another method using a convolutional neural network (CNN), useful for estimating complex poses, has been proposed. Unfortunately, it is difficult to collect the large number of images with ground truth needed for learning. In this research, we focused on creating a large dataset by generating hand pose images using a digital human model and motion-captured data using DhaibaWorks. We evaluated the model by calculating the distance of the estimated pose and ground truth of the test data, which was approximately 12.3 mm on average. In comparison, the average distance in related work was 18.5 mm. We also tested our method with ordinary camera images and confirmed that it can be used in the real world. Our method provides a new means of dataset generation: annotations are done automatically with motion capture technology, which reduces the time required. In future work, we will improve the architecture of the CNN and shorten the execution time for real-time processing.

Keywords: 3d hand pose estimation, convolutional neural network, motion capture

1. Introduction

Hand pose estimation has recently become a hot topic in computer vision. Although a hand is a tool for handling objects, most research in this area has not involved handling objects. In addition, a majority of prior research has used depth images as input, but depth cameras are not widely used in society. If it is possible to use RGB images as input, pose estimation can be easily done and utilized to evaluate the development of the grip of daily necessities.

The pose estimation method that is widely used is OpenPose [1]. OpenPose is an algorithm for skeleton detection, using a single camera and deep learning for estimation. It detects the whole (or parts of) the human body and displays its joints. However, considering it as a hand pose estimation method, the issue is that if the body cannot be seen from the input image, the whole estimation cannot be completed. When the camera is on an object, in most cases only hands can be seen from the image.

Kashiwagi [2] proposed a method for grasping hand pose estimation with a fisheye camera embedded in an object. It detects the fingertips with the common RGB threshold and computes their positions, on the premise that viewable fingers in the input images are known. Then, it displays the estimated pose by synthesizing a pre-prepared hand reference model to the computed position of fingers. The input is an RGB image, and estimation is done in real-time; thus, it can be easily used. The main limitation is that it requires a reference model, which restricts how the hand is grasping. For example, when the model is a pose of holding a ball with 4 fingers, the output will be similar, even if the real hand is not.

Considering the aforementioned methods, this study focuses on hand pose estimation with an RGB image taken from a camera embedded in the object and using deep learning to remove the hand-grasping limitation.

2. Construction of Convolutional Neural Network (CNN) Model

2.1. Flowchart of method

The flowchart of the proposed method is shown in Fig. 1.
As a pre-process, we generated the dataset and learned using a CNN [3]. Using 3D positions of markers on the hand (we define this as a keypoint) obtained from motion-captured data, we generated the CG models of the hand frame by frame. After changing the hand colors and the camera position, we constructed a machine learning network to predict the 3D position of 28 keypoints from the RGB image taken from the same position of the dataset, and the output was the estimated 3D position of the keypoints.

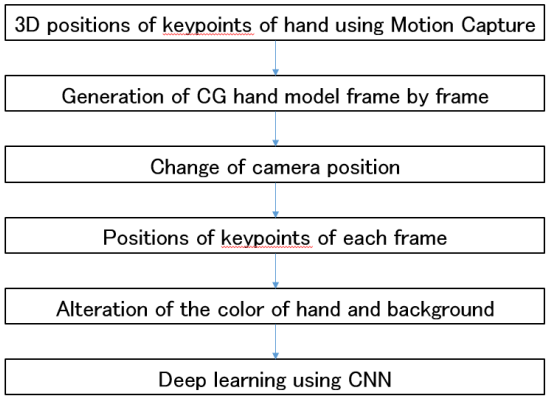

Fig. 1. Flowchart of proposed method

2.2. Motion Capture

The dataset was created using DhaibaWorks (DW) [4]. DW is a software program developed by AIST for utilizing digital human technology in product design. Its advantage is that it can generate poses of body parts as a CG model referring to the motion capture data. There are certain hand pose datasets [5, 6] that can be used publicly; however, because the annotation of keypoints is done manually, some points are not labeled, and labeling itself is also unstable.
Nevertheless, simply applying motion capture data will not improve the accuracy of pose estimation. Several issues with motion capture is the sync method, tracking error, and processing lost keypoints. For that, DW has a “fitting” function, which completes and adjusts the keypoints according to the constraint condition of hands. Vicon Nexus (VN) [5], an auto-labeling software program that also improves the tracking error. It reads multi-frame data, and if the same markers exist between the adjacent frames, it automatically labels them as the same. Thus, there is no need to label each frame. These software programs will solve such issues.
In the Digital Human Group of AIST, the marker positions are defined, shown in Fig. 2. Leveraging their position, DW will output the suitable hand pose.

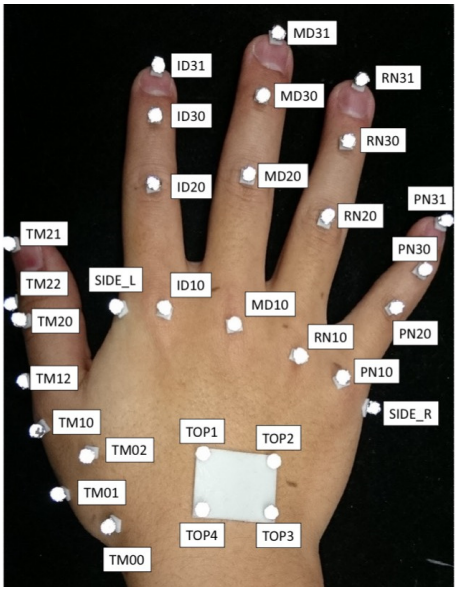

Fig. 2. Marker position

2.3. Creating the Datasets

For motion capture, we put markers on participants' hands and recorded their position when grasping objects. We defined them as keypoints, and our goal was to estimate their position. DW output the hand pose model. We then moved the viewpoints on the object so the hand could always be seen from the camera. In reality, the skin color and background color differed in most cases, so we changed the colors of the model. This process can be seen in Fig.3. We repeated this to create more than 5000 images for the datasets.

Fig. 3. Process for generating images for dataset

2.4. Neural Network

This method will use the CNN for learning. CNN added convolutional layers to an ordinary neural network to enable region-based feature extraction. This makes it robust to transform images. There are convolutional layers for feature convolution, pooling layers for reducing parameters and the amount of computation to control overfitting, and fully-connected layers for making final decision according to features. By putting these layers in a certain order, researchers try to improve the estimation accuracy. We used Inception V4 (IV4) [7], which performed well in image classification competition [8]. According to one study that compared the performance of CNN architecture [9], as of 2016 it is the best network. In the proposed method, the input will be the RGB image and the keypoint positions are the instruction signal. The output will be 84 values, 3 values (x, y, z) for each of the 28 keypoints.

3. Estimation and Its Performance Evaluation

Using motion capture data and DW, we created 5810 images for the datasets and 113 images for evaluation and then compared the estimation error with Zimmermann's method [10]. Further, we compared the estimation accuracy and time with different network architecture.
We put 28 markers on our participants' right hand and let them hold a transparent ball with a radius of 5cm. We created 2 datasets, one with a noised background (Dataset A) and one without (Dataset B) (Fig. 4). We used a ball the size of a baseball and asked participants to hold it in 3 ways: with 3 (thumb, index, middle finger), 4 (without little finger), and 5 fingers. We recorded the marker positions in 50 fps, but because there was little difference between adjacent frames, we set to obtain images in every 2 frames. We set the viewpoint at the opposite of the hand (red point in Fig. 5).

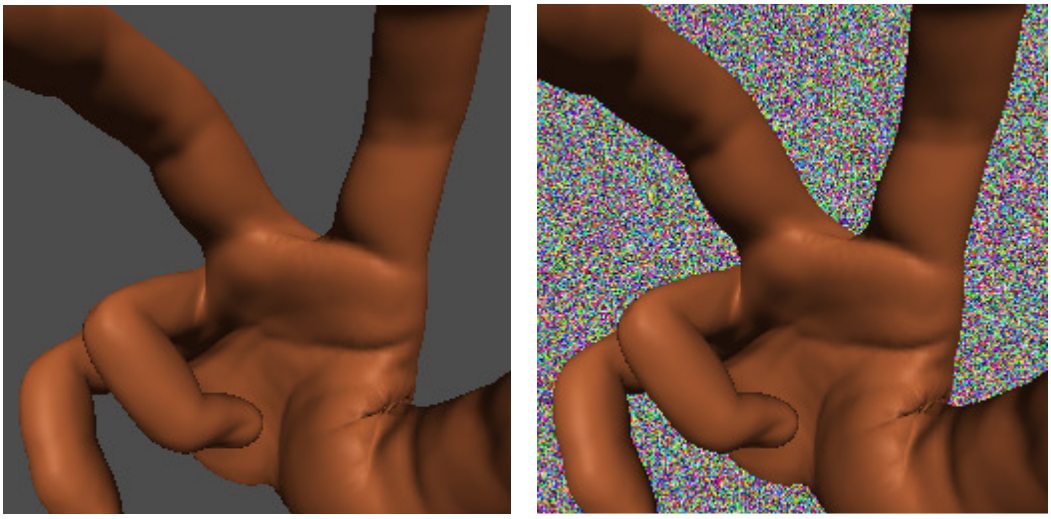

Fig. 4. Example image of datasets (left: Dataset A, right: Dataset B)

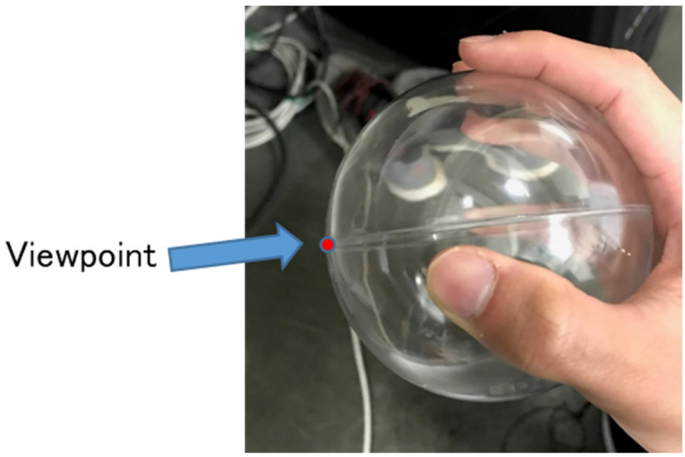

Fig. 5. Viewpoint

We compared the estimated position of each dataset with the ground truth obtained by motion capture. (Table 1). Zimmermann's method can be estimated in a free viewpoint, but our method performed better in both datasets, even if not all the keypoints were seen in the image.

Table 1. Average estimation error of keypoints (mm)

Dataset A	Dataset B	Zimmermann [10]
12.31	12.69	18.8

In addition, keypoint which had the largest error was at the tip of the little finger. We consider this is since this marker was hardly viewable in most of the images in the dataset. In contrast, error of the estimated position at index, middle finger and ring finger was small. Since result in dataset A was better, in the following experiment we used the model trained with dataset A.

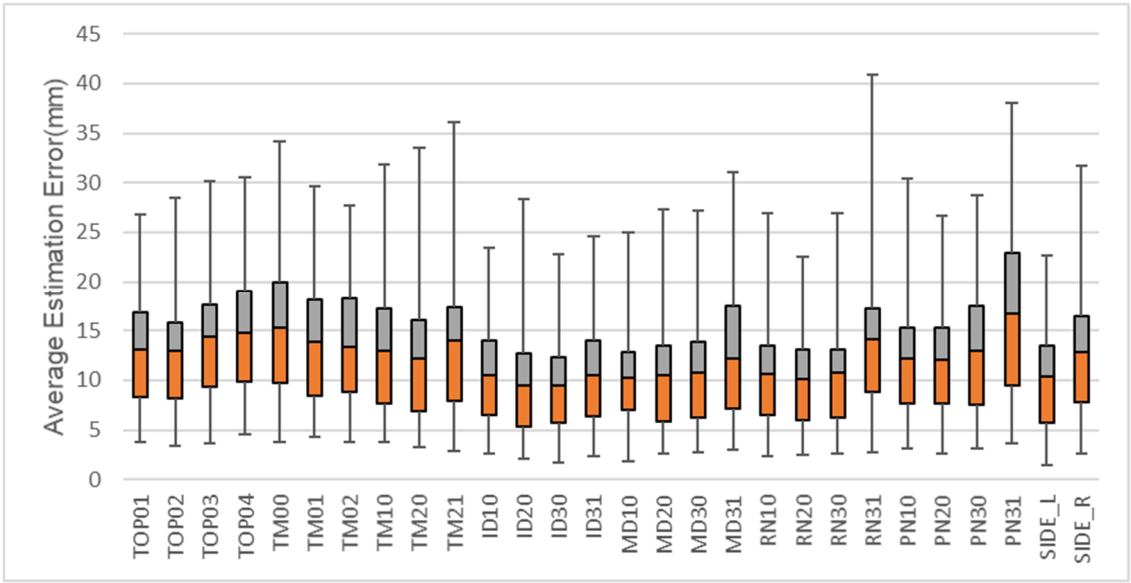

Fig. 6. Average estimation error of each keypoints (marker names refer to Fig. 2)

As mentioned in the last chapter, Inception V4 was used for the model architecture, but we also referred to other architecture, VGG16 [11] and the model (Fig. 7) to confirm that this was an adequate choice. VGG16 is also one of the networks that made a significant contribution in image classification competition, with fewer layers and less effort than IV4. VGG16 and IV4 have many layers. Thus, to investigate whether the number of layers and execution time are correlated, we created another model (Fig. 7), which has a simple structure [12] and a small number of layers. We used dataset A for comparison.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 128, 128, 3)	0
conv2d_1 (Conv2D)	(None, 126, 126, 32)	896
conv2d_2 (Conv2D)	(None, 124, 124, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 62, 62, 32)	0
dropout_1 (Dropout)	(None, 62, 62, 32)	0
conv2d_3 (Conv2D)	(None, 60, 60, 64)	18496
conv2d_4 (Conv2D)	(None, 58, 58, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 29, 29, 64)	0
dropout_2 (Dropout)	(None, 29, 29, 64)	0
flatten_1 (Flatten)	(None, 53824)	0
dense_1 (Dense)	(None, 512)	27558400
dropout_3 (Dropout)	(None, 512)	0
output1 (Dense)	(None, 1)	513

Fig. 7. Model for comparison

Table 2. Average estimation error of keypoints and execution time

	Average estimation error (mm)	Execution time per image (s)
Model in Fig. 7	19.42	2.01
VGG16	17.35	2.28
IV4	12.31	7.41

From Table 2, we see that number of layers and execution time are correlated. The keypoint position was based on the definition by AIST, which has 4 markers on the back of the hand and 4 markers on the thumb at a similar position. The execution time is affected by the number of parameters to estimate; thus, these markers may be partially ignored in the future.
We also investigated the accuracy in (a) various numbers of dataset images and (b) camera positions. For (a), we made 3 datasets. The first one (A) is the same as the last experiment, the second (B) has 1 / 3 images from the first one, and the third (C) has 1 / 9 images of the same.

Table 3. Result of experiment (a) (mm)

(A)	(B)	(C)
12.31	20.77	265.52

Referring to Table 3, it seems that the more images there are, the better the accuracy. However, (C), which had approximately 600 images in total, dropped drastically.
For (b), we set the viewpoint closer to the hand (Fig. 8) and compared the estimation results. The results are shown in Table 4.

Table 4. Result of experiment (b) (mm)

Regular	Viewpoint at Fig. 8
12.31	11.31

The results were better when the viewpoint was closer. Therefore, if part of the finger and palm can be seen, good accuracy can be obtained.

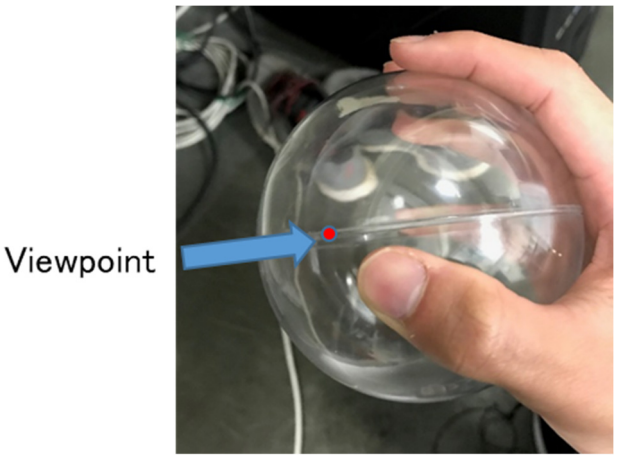

Fig. 8. Viewpoint in experiment (b)

For public use, an image taken with a real camera is necessary. Thus, we obtained such a photo (Fig. 9) for an additional experiment. The result output is shown in Fig. 10.

Fig. 9. Input image

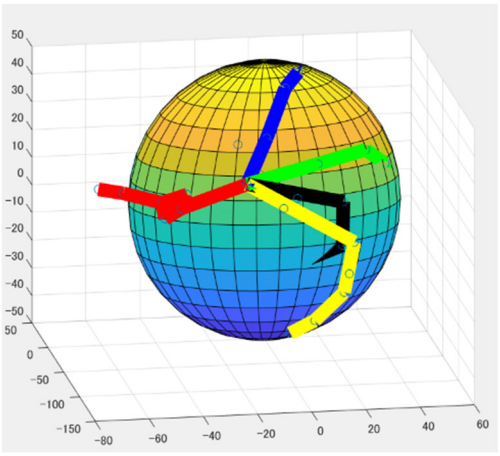

Fig. 10. Estimated pose

In the input image, a ball was held with 3 fingers, and the ring finger and little finger were bent. The black and yellow lines in Fig. 8 are the ring finger and little finger, respectively, and it shows that the pose is correctly estimated.

4. Conclusion and Future Work

This paper presented a method for grasping hand pose estimation. Moreover, by leveraging motion capture, we created a dataset without manual annotation or the use of a depth camera. In the evaluation section, we compared the accuracy with some CNN architecture and confirmed that Inception V4 performed the best.

In this research, the output was shown merely by plotting the estimated keypoints, referring to Zimmermann's method [10]. However, in the plotting stage, some of the keypoints were ignored for clear visualization. For learning efficiency, these points may be skipped, which may reduce the computational cost.

For future work, estimating the region of the hand before estimating a pose itself may improve the results. We also plan to conduct similar research with other objects so this method can be applied widely. Using DhaibaWorks and motion capture, we enabled better camera positions to be considered, even after the data were collected.

References

[1] Z. Cao, T. Simon, S. E. Wei and Y. Sheikh, “Realtime Multi-person 2D PoseEstimation Using Part Affinity Fields,” in IEEE Conference on Computer Vision and Pattern Recognition, pp. 1302-1310, 2017.

[2] N. Kashiwagi, Y. Sugiura, N. Miyata, M. Tada, M. Sugimoto and H. Saito, “Measuring Grasp Posture Using an Embedded Camera,” in *IEEE Winter Applications of Computer Vison Workshops*, pp.42-47, 2017.

[3] Y. Lecun, P. Haffner, L. Bottou, Y. Bengio, “Object Recognition with Gradient-Based Learning,” in Shape, Contour, and Grouping in Computer Vision, pp.823-823, 1999.

[4] *DhaibaWorks,* www.dhaibaworks.com., accessed 2018.

[5] S. Sridhar, F. Mueller, M. Zollhofer, D. Casas, A. Oulasvirta and C. Theobalt, “Real-time joint tracking of a hand manipulating an object from RGB-D input.” in Proceedings of the European Conference on Computer Vision, 2016.

[6] J. Zhang, J. Jiao, M. Chen, L. Qu, X. Xu and Q. Yang, “3D Hand Pose Tracking and Estimation Using Stereo Matching,” in arXiv preprint arXiv:1610.07214, 2016.

[7] C. Szegedy, S. Ioffe, V. Vanhoucke and A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” in AAAI, pp.4278-4284, 2017.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge.” in International Journal of Computer Vision. Vol 115, Issue 3, pp. 211252, 2015.

[9] A. Canziani, A. Paszke and E. Culurciello,“An Analysis of Deep Neural Network Models for Practical Applications,” in arXiv preprint arXiv:1605.07678, 2017.

[10] C. Zimmermann and T. Brox, “Learning to Estimate 3D Hand Pose from Single RGB Images,” in *IEEE International Conference on Computer Vision*, pp.4913-4921, 2017.

[11] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in arXiv preprint arXiv:1409.1556, 2014.

[12] Keras Github, https://github.com/keras-team/keras, accessed 2018.