Sensor signal processing

Head Posture Estimation by Deep Learning Using 3-D Point Cloud Data From a Depth Sensor

Seiji Sasaki[1] and Chinthaka Premachandra[1,2,*]

[1] *Department of Electronic Engineering, School of Engineering, Shibaura Institute of Technology, Tokyo 135-8548, Japan*
[2] *Graduate School of Engineering and Science, Shibaura Institute of Technology, Tokyo 135-8548, Japan*
[*] *Senior Member, IEEE*

Manuscript received June 3, 2021; accepted June 20, 2021. Date of publication June 22, 2021; date of current version July 6, 2021.

Abstract—Head posture estimation is performed by capturing characteristic areas of the face, such as the eyes and nose, in images acquired from a camera installed in front of the subject. However, with this method, parts of the eyes and nose are hidden when the subject turns away and faces the side, making estimation difficult. In this letter, we aim to realize a more effective head estimation method than previous research using 3-D point cloud data from a depth sensor. We pursued the estimation of five head posture classes. In the proposed method, first, the 3-D point cloud data of the postures in the five classes are learned by a deep learning model. Next, the posture of the head is estimated using the model. In this letter, many verification experiments confirmed that the proposed method is very effective for head posture estimation with five posture classes.

Index Terms—Sensor signal processing, deep learning, depth sensor, head pose estimation, 3-D point cloud data.

I. INTRODUCTION

Head posture estimation technology is used for driving support systems that confirm the condition of automobile drivers and for measuring the points of interest of consumers in front of product shelves [1], [2]. It is also used to measure the effectiveness of digital advertising signage using a liquid crystal display. Furthermore, the use of virtual avatars in the entertainment industry and communication has increased in recent years, such as by Vtubers and applications as VRChat [3], which use head posture estimation to link the movements of performers and avatars. Thus, head posture estimation technology is very useful in a wide range of fields, including driving assistance, marketing, and entertainment. In the conventional head posture estimation method, posture is estimated by capturing characteristic areas of the face, such as the eyes and nose, using images acquired from a camera placed in front of the subject [4]–[7]. However, with that method, the eyes and nose are hidden when the subject turns sideways, making estimation difficult. In addition, this method cannot be used when the camera cannot be placed in front of the subject. Therefore, in this letter, we aimed to estimate posture over a wider range than achieved before by estimating the direction of the face using 3-D point cloud data. When 3-D point cloud data are used, the user's face does not appear as an image, so posture estimation is effective even in an environment where the user must remain anonymous. The 3-D point cloud data are a set of points with coordinate information in 3-D space that can represent the shape of an object. In this study, we used a RealSense depth sensor to acquire the 3-D data of a face. These kinds of 3-D depth sensors have been used to develop different consumer applications [8]–[11]. In this letter, we confirmed that the estimation is effective for the task of dividing the posture into five classes by the verification experiment using a deep learning model.

Corresponding author: Chinthaka Premachandra (e-mail: chintaka@sic.shibaura-it.ac.jp).
Associate Editor: R. Vida.
Digital Object Identifier 10.1109/LSENS.2021.3091640

Table 1. Computer Specifications.

CPU	Intel Core i5-7500 @3.40GHz
GPU	NVIDIA GeForce RTX2060
RAM	8GB

Table 2. Head Posture Classes.

①	Head tilted 90 ° to the right on the Y axis
②	Head tilted 90 ° to the left on the Y axis
③	Head tilted 45 ° to the right on the Y axis
④	Head tilted 45 ° to the left on the Y axis
⑤	Head from the front (0°)

II. PROCESS FLOW OF HEAD ESTIMATION

We conducted the study using one depth sensor and a computer. We coded with Python, which is widely available for software development machine learning libraries, and used PyTorch [12] as the deep learning library. In addition, to estimate the head posture using deep learning, we used a GPU to speed up the processing. Table 1 lists the specifications of the computer.

A. Classification of Head Posture Estimation

Fig. 1 shows the overall process flow of the proposed head posture estimation. First, a depth image is taken with the depth camera, and point cloud data are generated based on the depth. Next, only the head data are extracted from the generated point cloud data. The extracted head data are input to a neural network for estimation. In the system proposed here, the head posture is estimated as one of five posture classifications. Each class is given in Table 2.

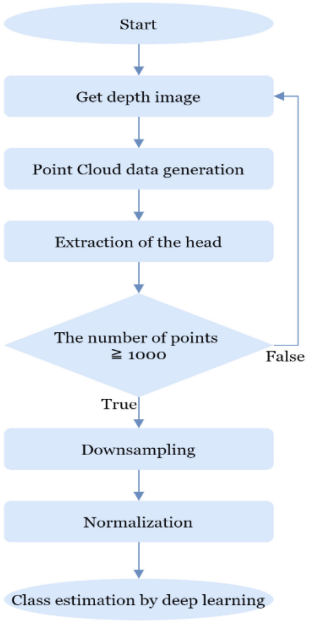

Fig. 1. Overview of the system.

Fig. 2. Intel RealSense D435 depth sensor.

Table 3. RealSense D435 Specifications.

Depth Stream Output Resolution	Up to 1280×720
Depth Stream Output Frame Rate	Up to 90fps
Depth Field of View (FOV) (Horizontal × Vertical × Diagonal)	91.2°×65.5°×100.6°（±3°）

B. Depth Sensor

A depth camera can measure the distance to the subject and takes a depth image. The distance data are stored in each pixel. The RealSense D435 depth sensor (Intel) [13] was used. This sensor is shown in Fig. 2. RealSense D435 is an active stereo depth camera equipped with two infrared cameras and an infrared projector. It projects an infrared dot pattern and calculates the depth value of each pixel from the parallax. The specifications of RealSense D435 are given in Table 3.

C. Acquisition of Head Data

Point cloud data were generated using Open3D [14], [15] from the depth image taken by the depth camera. Open3D is an open-source 3-D data processing library distributed under the MIT license. In this study, the resolution of the depth camera was set to 640 × 480 pixels. The captured depth image and the generated point cloud data are shown in Figs. 3 and 4, respectively.

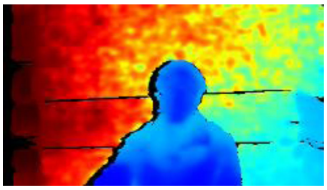

Fig. 3. Depth image.

Fig. 4. Generated point cloud data.

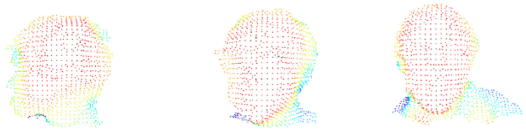

Fig. 5. Examples of generated head position data.

Each point in the generated point cloud is first classified using a clustering method called DBSCAN [15]. Next, the data corresponding to the head are selected from the classified point cloud, and the head data are generated. The head data are downsampled so that the number of points becomes 1000 for input to the neural network. The generated head data are shown in Fig. 5.

D. Normalization of Point Cloud Data

In this study, we examined posture estimation with classification in which the position of the head is divided into five postures as described above. PointNet by Qi *et al.* [16] was used as the classification method. PointNet is a deep learning model that uses point cloud data and can take invariant and local features with respect to the order and orientation of point clouds. The experiments and evaluations were performed using head data acquired in advance.

First, the PCD file in which the head point cloud data are recorded is read, and normalization is performed based on the maximum vector distance from the origin in the sample. The formula for preprocessing is

$$x_{\text{norm}} = \frac{x_i - \sigma}{\max\{|x_i - \sigma|\}} \tag{1}$$

where x_i is the input, σ is the average of the point cloud on the x-axis, y-axis, and z-axis, and x_{norm} is the output. Normalization is performed by finding the difference from the average of each axis for the point cloud data and dividing by the maximum value of the distance at each point. Next, the results are input to PointNet of the deep learning model for class estimation.

E. Deep Learning Using Point Cloud Data

To handle point cloud data with machine learning, it is necessary to consider order invariance and movement invariance. Order invariance means that the result is invariant to the order of the point cloud. For example, a gray image can be represented by a 2-D matrix with each pixel regularly arranged. However, unlike pixels, point cloud data do not change the information represented by the data position, even if the order of points changes. Also, when input to the machine learning model, the point cloud may be presented in a different order each time. Therefore, it is necessary to build a model that does not depend on the order of data. As a method for this, a symmetric function is used in the model. A symmetric function is a function whose value does not change even if the order of variables is changed. In PointNet, forward invariance is obtained by performing MaxPooling, which is a symmetric function, in the pooling layer. The MaxPooling function outputs the maximum value element in the input element. The function equation is

$$f\ (x_1, x_2, \ldots, x_n) = \max\{x_1, x_2, \ldots, x_n\}. \tag{2}$$

The same result can be obtained by exchanging x_1 and x_2.

Movement invariance means that invariant results can be obtained for point cloud data that have been translated or rotated. The invariance for translation is expressed by the following equation:

$$f\ (x_1 + \boldsymbol{r}, x_2 + \boldsymbol{r}, \ldots, x_n + \boldsymbol{r}) = f\ (x_1, x_2, \ldots, x_n) \tag{3}$$

where r represents an arbitrary vector. Even if the input x_n is moved by an arbitrary vector r, in this way, the output does not change. Invariance with respect to rotational movement is expressed by the following equation, where the rotation matrix $\boldsymbol{R}$ is used:

$$f\ (\boldsymbol{R}x_1, \boldsymbol{R}x_2, \ldots, \boldsymbol{R}x_n) = f\ (x_1, x_2, \ldots, x_n). \tag{4}$$

To satisfy these equations, PointNet estimates the affine transformation matrix for the input point cloud and multiplies the matrix to obtain the movement invariance approximately. A neural network with forward invariance called T-Net is used to estimate the affine transformation matrix. Each network is expressed by the following equation:

$$f\ (\{x_1, x_2, \ldots, x_n\}) \approx g\,(h\,(x_1),\ h\,(x_2), \ldots h\,(x_n)) \tag{5}$$

where $h()$ is approximated by a multilayer perceptron, independent transformation is performed for each point, and MaxPooling is applied by $g()$ to obtain the overall features. In addition, local features for each point can be obtained by combining the overall features obtained here with the features to which the affine transformation [17]–[19] of each point is applied.

Different types of networks are proposed in the literature for different classification problems [20]–[22]. The structure of the network in this study is shown in Fig. 6. If the number of points is n, the input is $n \times 3$ data. The invariance with respect to the orientation is obtained by estimating the matrix for performing the affine transformation in the transform layer and calculating the product with the input. Next, by applying MaxPooling through convolution and multilayer perceptron, 1024-dimensional features are acquired.

III. EXPERIMENTS

A. Experimental Environment

We prepared unique learning and verification datasets for head posture estimation. Depth images were taken with a tripod-mounted RealSense D435 at a distance of about 40 cm from the head and at the same height as the subject's line of sight. Fig. 7 shows the environment at the time of image acquisition.

We prepared 120 learning data points and 50 verification data points for five classes, and the accuracy was measured with 600 learning data points and 250 verification data points. The data prepared in this study were collected from several subjects.

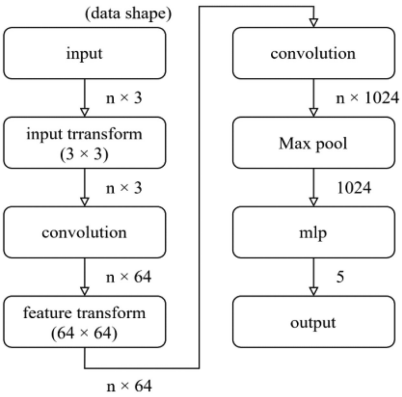

Fig. 6. PointNet network structure.

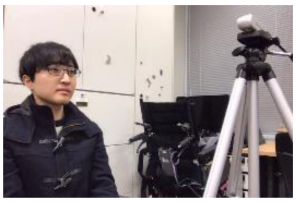

Fig. 7. Imaging environment.

Table 4. Accuracy of Validation Data.

Nunber of trials	Accuracy
#1	96%
#2	98%
#3	97%
#4	96%
#5	95%
Average	96.4%

Table 5. Analysis Result of Each Class by Confusion Matrix.

Target angle	0	Left−45	Left−90	Right−45	Right−90
Precision	90.9%	100%	100%	100%	100%
Recall	100%	90.0%	100%	100%	100%
F1−Score	95.2%	94.7%	100%	100%	100%

B. Experimental Results

Table 4 lists the accuracy and average of the verification data when the number of epochs in one training round was 15, and the training was performed five times. Fig. 8 shows a sample of the correct label and the estimated label for the point cloud data. Table 5 lists the precision, recall, and F1 score for each class based on this confusion matrix.

Table 6 lists the processing time from imaging to the output of the estimation result.

In this letter, five classes of head posture were determined by deep learning, and the models taught using data from several heads were

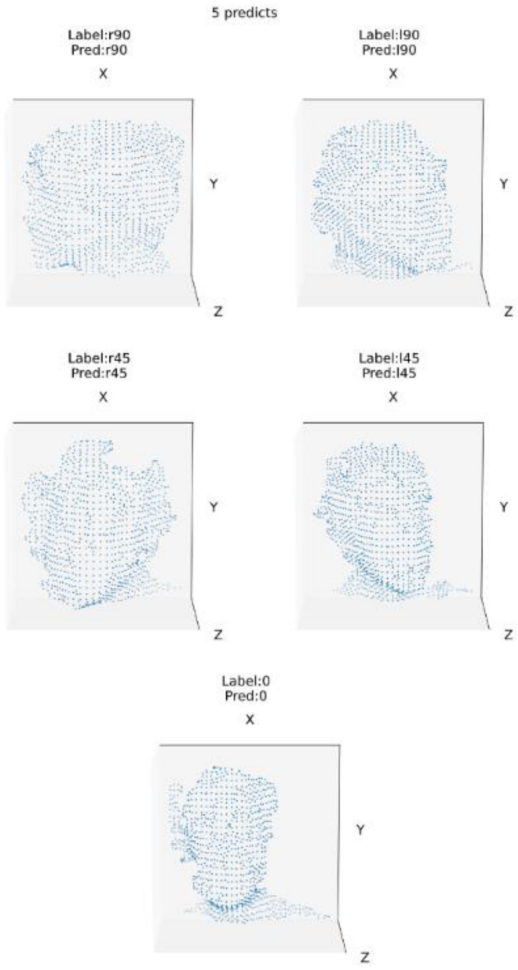

Fig. 8. Validation data sample and estimation results.

Table 6. Processing Time.

Number of trials	Processing time[ms]
#1	3096.816ms
#2	3327.370ms
#3	3245.507ms
#4	3118.869ms
#5	3201.047ms
Average	3197.922ms

able to classify with high accuracy. In addition, in view of the mixed matrix of the estimation results, the number of elements in the matrix appears diagonally, and it can be said that the estimation was possible with high accuracy from this figure as well.

DBSCAN was evaluated for extracting head data in this research, but this clustering method makes real-time estimation difficult because of its high calculation cost and time. In addition, because the distance from the camera to the person is fixed, it is necessary to give flexibility to the camera placement position for practical use. This increases speed and flexibility by measuring the distance from the camera after detecting the face and hands of a person before starting the estimation. Furthermore, it uses that distance as a threshold value when extracting the head from the point cloud data.

IV. CONCLUSION

In this letter, we investigated head posture estimation by deep learning using the 3-D point cloud data. Point cloud data were generated from depth images, head extraction was performed by DBSCAN, and the number of points was downsampled to 1000. These data were used as input to the PointNet deep learning network. Learning was performed by dividing postures into five classifications. The results confirmed that the attitude estimation in the horizontal direction is effective.

REFERENCES

[1] H. Han, H. Jang, and S. W. Yoon, "Driver head posture monitoring using MEMS magnetometer and neural network for long-distance driving fatigue analysis," in *Proc. IEEE Sensors*, Oct. 2019, pp. 1–4.

[2] Y.-W. Chuang, S.-W. Sun, and P.-C. Chang, "Driver posture recognition for 360-degree holographic media browsing," in *Proc. 10th Int. Conf. Ubi-Media Comput. Workshops*, 2017, pp. 1–6.

[3] L. Bredikhina, T. Kameoka, S. Shimbo, and A. Shirai, "Avatar driven VR society trends in Japan," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces Abstr. Workshops*, Mar. 2020, pp. 497–503.

[4] Y. Gu, L.-T. Hsu, L. Xie, and S. Kamijo, "Accurate estimation of pedestrian orientation from on-board camera and inertial sensors," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E99-A, no. 1, pp. 271–281, 2016.

[5] Y. Gu and S. Kamijo, "Bicyclist recognition and orientation estimation from on-board vision system," *Int. J. Automot. Eng.*, vol. 6, no. 2, pp. 67–73, 2015.

[6] G. Borghi, M. Fabbri, R. Vezzani, S. Caldera, and R. Cucchiara, "Face-from-depth for head pose estimation on depth images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 3, pp. 596–609, Mar. 2020.

[7] J. D. Ortega *et al.*, "DMD: A large-scale multi-modal driver monitoring dataset for attention and alertness analysis," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 387–405.

[8] H. Tang, J. Ohya, and T. Ohkawauchi, "Basic study of line-of-sight estimation method from the position of the nose of a person using near-infrared images," in *Proc. IEICE Gen. Conf.*, 2008.

[9] Y. Ito, C. Premachandra, S. Sumathipala, H. W. H. Premachandra, and B. S. Sudantha, "Tactile paving detection by dynamic thresholding based on HSV space analysis for developing a walking support system," *IEEE Access*, vol. 9, pp. 20358–20367, 2021.

[10] X. Ning and G. Guo, "Assessing spinal loading using the kinect depth sensor: A feasibility study," *IEEE Sensors J.*, vol. 13, no. 4, pp. 1139–1140, Apr. 2013.

[11] M. Zhang and W. Cai, "Energy-efficient depth based probabilistic routing within 2-hop neighborhood for underwater sensor networks," *IEEE Sensors Lett.*, vol. 4, no. 6, Jun. 2020, Art. no. 7002304.

[12] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," *Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 8024–8035, 2019.

[13] "Intel RealSense camera D400 series product family datasheet," Jan. 2, 2021. [Online]. Available: https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discov. Data Mining*, 1996, pp. 226–231.

[15] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3-D data," 2018, *arXiv:1801.09847*.

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3-D classification and segmentation," 2016, *arXiv:1612.00593*.

[17] S. Ono and C. Premachandra, "Generation of panoramic images by two hemispherical cameras independent of installation location," *IEEE Consum. Electron. Mag.*, to be published, doi: 10.1109/MCE.2020.3031090.

[18] O.-S. Kwon and Y.-H. Ha, "Panoramic video using scale-invariant feature transform with embedded color-invariant values," *IEEE Trans. Consum. Electron.*, vol. 56, no. 2, pp. 792–798, May 2010.

[19] L. Brun and A. Gasparini, "Enabling 360° visual communications: Next-level applications and connections," *IEEE Consum. Electron. Mag.*, vol. 5, no. 2, pp. 38–43, Apr. 2016.

[20] A. S. Winoto, M. Kristianus, and C. Premachandra, "Small and slim deep convolutional neural network for mobile device," *IEEE Access*, vol. 8, pp. 125210–125222, 2020.

[21] Y. Yamazaki, C. Premachandra, and C. J. Perera, "Audio-processing-based human detection at disaster sites with unmanned aerial vehicle," *IEEE Access*, vol. 8, pp. 101398–101405, 2020.

[22] M. Demirhan and C. Premachandra, "Development of an automated camera-based drone landing system," *IEEE Access*, vol. 8, pp. 202111–202121, 2020.