Gesture Recognition with non-contact sensor for Natural User Interface in the COVID-19 era

Yoshihisa Nitta
Department of Computer Science
Faculty of Liberal Arts
Tsuda University
nitta@tsuda.ac.jp

Himawari Ichikawa
Graduate School
Tsuda University
m18hichi@gm.tsuda.ac.jp

Yuko Murayama
Institute for Mathematics and Computer Science
Tsuda University
murayama@tsuda.ac.jp

Abstract

COBID-19, an infectious disease transmitted by droplet and contact, is prevalent. In order to reduce the risk of contact infection, various operations should be performed in silence and non-contact. A user interface using non-contact sensors is effective in such an environment. Among them, Natural User Interface based on Gesture Recognition using non-contact sensors are useful, we think.

We have developed our NUI system in which the user instructs the computer in a full-body gesture. In this paper, we discuss several methods available for gesture recognition based on skeleton recognition. And, for some of the gesture recognition systems we have implemented with the combination of such methods, the design policy and experimental results of each are presented.

1. Introduction

The infectious disease COVID-19 is prevalent. There is an urgent need to take steps to prevent the spread of the disease [1]. The COVID-19, an infectious disease is transmitted by droplet and contact. The risk of contact infection increases due to touch screens and buttons that are touched by many people. For this reason, it has become desirable to perform various operations silently and without contact.

What is most effective in such an environment is a user interface using non-contact sensors. Among them, natural user interfaces based on gesture recognition using non-contact sensors are useful, we think. We have developed our NUI system in which the user instructs the computer in a fully-body gesture [2, 3, 4]. These studies are becoming increasingly important in environments where contact infection should be avoided.

In this paper, we discuss several methods available for gesture recognition based on skeleton recognition. And, for some of the gesture recognition systems we have implemented with the combination of such methods, the design policy and experimental results of each are presented.

In addition, although we have conducted research using an RGB-D camera, Microsoft Kinect for Windows V2 (hereinafter called "Kinect V2") [5], the RGB camera is preferable for practical application. Therefore, we are currently moving to the method using OpenPose [6], and describe the design policy and plan of the new system.

2. Skeleton Recognition

Kinect V2 is a RGB-D camera device that recognises the human skeleton and gets the positions of joints. It can get 3-D coordinates of each 25 joints for up to 6 people at the same time. Kinect V2 was discontinued in October 2018, but there is no problem in prototyping our system.

We have developed NtKinect [7] and NtKinectDLL [8] to make Kinect V2 available with many programming languages and development environments. We released them as Open Source Software. With NtKinect, you can use Kinect V2 in Python's machine learning system with ease.

Intel's RealSense [9] is also an RGB-D camera. By using the official SDK, the 3D positions of the face and hands can be acquired. It can also be used in combination with software such as Nuitrack [10] to recognize the whole body skeleton.

OpenPose [6] is a study of skeleton recognition with RGB camera. It can recognize skeletons of any number of people at the same time, and it can recognize skeletons with high accuracy even if a part of the body is hidden. OpenPose uses Deep Learning to calculate the possibility that each pixel in a 2D RGB image is a specific joint, and creates a heatmap for the entire image. Then, it recognizes the skeleton for each person by looking for possible pairs of detected joints. The advantage is that it does not require depth data.

URI: https://hdl.handle.net/10125/70675
978-0-9981331-4-0

3. Gesture Recognition

Visual Gesture Builder [11] is a tool for full body gesture recognition using Kinect V2. In order to recognize a certain gesture, training is performed by given information in which the part of the gesture in the video is labeled as positive(+) and other part is labeled as negative(-). Two types of gestures, *discrete* gesture and *continuous* gesture, can be learned. The former uses AdaBoost and the latter uses Random Forest. The gesture we want to recognize corresponds to continuous gesture, but we did not adopt this tool because its recognition rate is not so high.

LeapMotion [12] is a device that operates applications with gesture, and uses an infrared sensor to recognize hand and finger movement. We did not use LeapMotion because we want to recognize the whole body gestures.

Cippitelli [13] has conducted research on behavior recognition based on skeleton data acquired by RGB-D cameras. They used the k-means method to select poses and Support Vector Machine (hereinafter called "SVM") to classify the feature vectors for the behavior recognition.

Taha [14] studied behavior recognition to monitor using RGB-D camera. The data is pre-processed by "rotating the joint data around the y-axis to formulate the data facing frontally" and "converting joints position to polar coordinates to eliminate the influence of pysical features". They use the Markov Model to recognize behavior.

Researches on gesture recognition based on pose recognition have made remarkable achievements. Many of them recognize human skeletons using RGB-D cameras and extract the temporal features of human skeletal joints using Deep Learning techniques as Recurrent Neural Network (hereinafter called "RNN") and Long Short Term Memory (hereinafter called "LSTM").

Yan [15] proposed Spatial-Temporal Graph Convolutional Networks (ST-GCN) for automatically learning both the spatial and temporal patterns from data.

Omran [16] proposed Neural Body Fitting (NBF). It integrates a statistical body model within a CNN, leveraging reliable bottom-up semantic body part segmentation and robust top-down body model constraints.

Li [17] proposed the actional-structural graph convolution network (AS-GCN), which stacks actional-structural graph convolution and temporal convolution.

4. Target of Our Research

Our goal is to develop an NUI system that operate computers by natural movement using the whole body. A natural motion is one intuitively performed by a human without wearing some devices like markers and sensors. However, it should be noted that "natural move" can vary depending on the culture and background of the users. Therefore, it is desirable that the gestures to be recognized are not fixed, but can be changed and extended by the user.

In our research, skeleton recognition is performed by the device such as Kinect V2, Intel RealSense or OpenPose. In other words, we are in a position to use existing tools for skeleton recognition. Thus, the procedure of our study is as follows.

- Selecting gestures to recognize
- Creating classifiers that detect gestures from time-series skeleton position data
- Providing APIs to other applications to use gesture recognition

A gesture classifier is created by machine learning.

We treat gestures as poses arranged in chronological order. Among poses that make up a gesture, we define some poses as *distinctive pose*s each of which is crucial to the identification of the gesture. We recognise a specific gesture by detecting the appearance of the distinctive poses in order within a short period of time.

Therefore, the following work is required to create a classifier that detects a gesture from time-series skeleton position data.

- Selecting distinctive poses
- creating classifiers to detect distinctive poses
- detecting the appearance of time-series sequence of dictinctive poses

We have developed four systems and conducted experiments to investigate various methods. Table 1 shows the design policy of each system. The expression 'n-stage relative' in the table means that the relative positional relationship between the joints is expressed in n stages.

5. Distinctive Pose

We define each pose that determines the gesture as a *distinctive pose*. Gestures are recognized by detecting the appearance of *destinctive poses* in time series.

Figure 1 shows an example of n skeleton data S_k ($k = 1, 2, \cdots, n$) in gesture G. In order to select L skeleton data as *distinctive poses* from n skeleton data,

Table 1. Policies of each system

Gesture Recognition System	version 1	version 2	version3	version 4
Skeleton Recognition Tool	Kinect V2	Kinect V2	Kinect V2	OpenPose
Data Representation	3-stage relative	Polar Coordinates	5-stage relative	5-stage relative
Selection Distinctive Pose	time tick	-	time span + k-means	time span + k-means
Detection Distinctive Pose	SVM	RNN	SVM	NN
Gesture Recognition	Exact Match		RNN	RNN

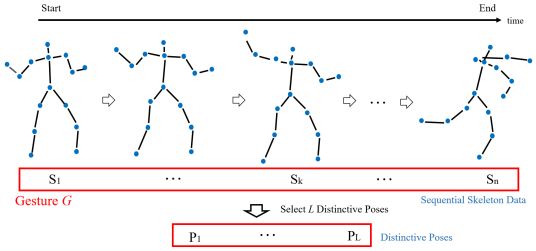

Figure 1. Time Sequence of Distinctive Poses

Figure 2. Non-Flexible Selection of Distinctive Pose

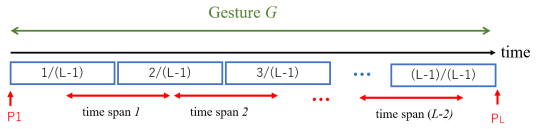

Figure 3. Flexible Selection of Distinctive Pose

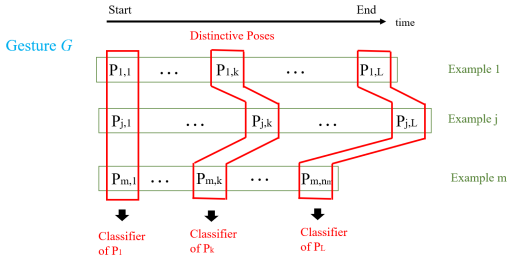

Figure 4. Classifier for Each Distinctive Pose

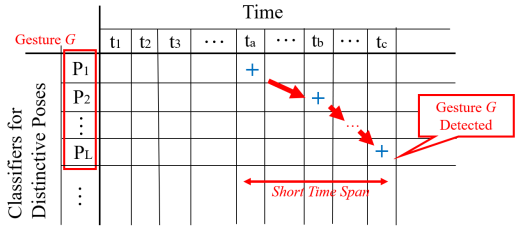

Figure 5. Sequential Outputs of Each Classifier

you can select $L-2$ *distinctive poses* between S_2 and S_{n-1} if the first S_1 and last S_n must be adopted.

5.1. How to Select Distinctive Pose

There are two ways to select L distinctive poses from n skeleton data.

- Non-Flexible selection: equally divided time tick
- Flexible selection: equally divided time span and k-means

Non-Flexible selection divides the elapsed time T of the gesture equally into $L-1$, and select the pose at the each time tick as distinctive pose (See Figure 2).

In the Flexible selection, the elapsed time T of gesture is divided equally into $L-1$. Then, the k-means method is applied to the the second half of the ith and the first half of the $(i+1)$th period. The data which is the centroid of each cluster of k-means is selected as the distinctive pose P_{i+1} (See Figure 3). If there are multiple leading clusters in the same time span, then multiple Distinctive Poses can be selected, which would make it a more appropriate choice.

5.2. How to Train the Classifier of Distinctive Poses

We sample m examples of gesture G which contains time-series skeleton data. The elapsed time for the gesture is different for each sample, so the number of skeleton data S_i also vary. For each sample data, L skeleton data are selected as distinctive poses P_{jk} ($1 \leqq j \leqq m$, $1 \leqq k \leqq L$). Figure 4 shows how to create a classifier that detects P_k from sampling data, $P_{1k}, \cdots, P_{mk}$. This is a classifier that detects k-th distinctive pose in the gesture G.

In order to recognize gesture G, it is necessary to detect the appearance of $P_1, P_2, \cdots, P_L$ in this order

(See Figure 5). This can be done in two ways.

- Non Flexible Detection: Exact Match
- Flexible Detection: RNN

6. Pre-processing of Skeleton Data

When working with skeleton data, the data must be pre-processed to reduce the difference between data for the following three points.

- user orientation to the sensor
- distance between sensor and user
- Physical characteristics of users

6.1. Rotation

To eliminate the effect of the user's orientation to the sensor, rotate the skeleton data of the user around the y-axis. The orientation of the user's front can be determined from the left and right positions of the waist for example. To rotate the coordinates (x, y, z) by θ around y-axis, use the following formula.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

6.2. Normalization

The size of the skeleton changes according to the user's height. The coordinates of measured joints may be affected by the distance between the sensor and the user. Therefore, using the coordinates of the joints as they are will cause problems.

In such cases, it is necessary to normalize the coordinates of the joints, rather than using them as they are. For example, the skeleton data are normalized to coordinates with the lower part of the human spine (SpineBase) as the origin point and the length of the spine as the basic unit system.

6.3. Polar Coordinates

Different users have different physical characteristics. For example, tall or short, long or short limbs, wide or narrow shoulders, etc.

Therefore, if the 3D coordinates of the joints are used as they are, individual differences will have a negative impact on perception. This effect remains even after applying the normalization process described in section 6.2.

The expression of the state of each joint can be converted from the position coordinates (x, y, z) in the 3D coordinates to the polar coordinates (θ, ϕ, r), which is the bending degree.

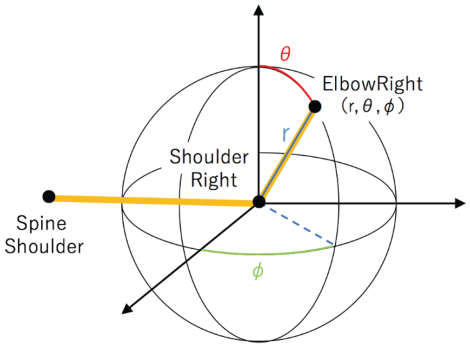

Figure 6. Polar Coordinates System

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = \cos^{-1}(\frac{z}{r})$$

$$\phi = \tan^{-1}(\frac{y}{x})$$

Because human bones do not change its length, the parameter r can be omitted. By removing parameter r, it is possible to eliminate the influence of the difference in the physical characteristics of the user. Also, since the number of input data is reduced, there is an advantage that the time for training is shortened.

For example, the state of the right shoulder is defined by the positional relationship of the three joints, Spine Shoulder, Shoulder Right, and Elbow Right. Figure 6 shows an example of this right shoulder state expressed in polar coordinates.

6.4. Quantization

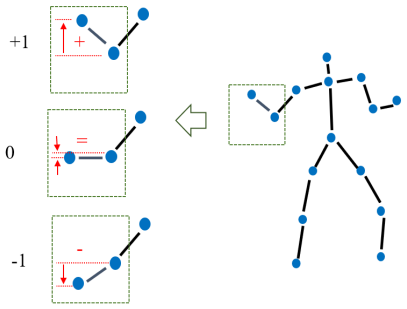

Figure 7. 3-stage Relative Relation of y-value between 2 joints

To train the classifier by using the joint data coordinates as they are, it needs a lot of examples and time. Quantizing coordinate data allows learning with a smaller number of examples and also reduces learning

time. It is also expected to eliminate the effects of individual physical characteristics, such as limb length.

Instead of using the coordinates of the joints as they are, there is a way to represent the pose using the relative positions between the joints.

Figure 7 shows an example of the y coordinates of the right elbow and the right wrist as a relative position of the 3-stage. The relative positional relationship between the y coordinates of the wrist and that of the right elbow is represented by a value of 1, 0, or -1.

In general, the value can be expressed in n-stage by taking into account the degree of difference.

7. Our System: Version 1

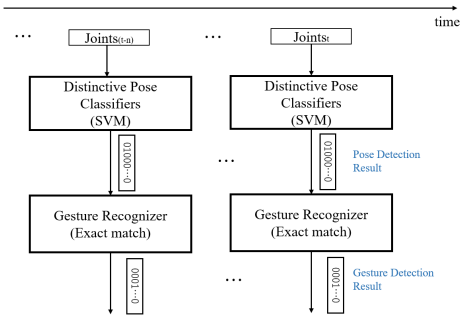

Figure 8. Diagram of Version1

We developed our Gesture Recognition System Version 1 [2] according to the following policy. Figure 8 shows its overall view.

- Skeleton data is pre-processed and expressed in 3-stage relative relation.
- Non-Flexible selection of Distinctive Poses.
- Creating classifier for each Distinctive Pose using SVM.
- using Exact Match to detect the time-series appearance of Distinctive Poses for a gesture.

See [2] for details of the experiment and its results. The version 1 has the following problems:

- 3-stage of relative positional relationships is too extremely quantized.
- Non-Flexible selection of Distinctive Pose
- Non-Flexible Gesture Detector using Exact Match

8. Our System: Version 2

We have developed Version 2 [3] to improve the drawbacks of Version 1. Figure 9 shows its overall view. The design policy of Version 2 is as follows.

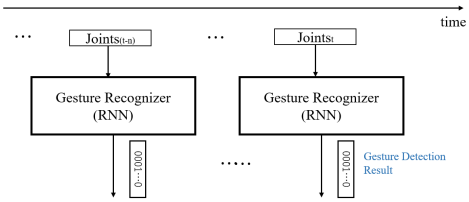

Figure 9. Diagram of Version2

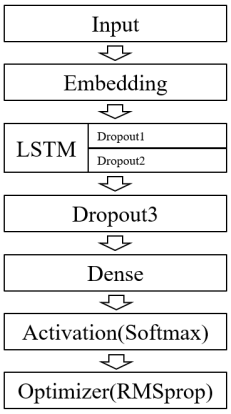

Figure 10. RNN Model of Version 2

- The Quantization of pre-processing is not performed on the skeleton data.
- The skeleton data is rotated around the y-axis, normalized, converted to polar coordinates, and input into the RNN.
- Input all time-series skeleton data directly into the RNN without selecting distinctive poses.
- The RNN is created using TensorFlow [18] and Keras [19]. The model of the RNN is shown in Figure 10.

To check the gesture recognition rate of Version 2 system, we conducted some experiments 2a and 2b. In experiment 2a, the "wave" gesture data from the Florence 3D Actions Dataset [20] is used. In experiment 2b, we prepared many of the same types of "wave" gesture data as in 2a.

Dropout1 is the LSTM layer's dropout rate to input data. Dropout2 is the LSTM layer's dropout rate on every retry. Dropout3 is the Dropout layer's dropout rate. We measured the recognition rate while varying these 3 dropout parameters. The number of hidden layers in the LSTM is 128, epoch is 100, and batch size is 64.

Among the result of experiment 2a, the overall recognition rate is high when Dropout3=0.1. The experimental results at that time are shown in Table 2. Training Accuracy is over 90%, but Test Accuracy is

Table 2. Experiment Result 2a for Version 2 [3]

Dropout1	Dropout2	Training Acc.	Test Acc.
0.2	0.2	0.9898	0.7754
0.2	0.3	0.9886	0.7373
0.2	0.4	0.9822	0.7792
0.2	0.5	0.9819	0.7665
0.3	0.2	0.9879	0.7386
0.3	0.3	0.9809	0.8135
0.3	0.4	0.9765	0.7931
0.3	0.5	0.9800	0.7906
0.4	0.2	0.9841	0.7297
0.4	0.3	0.9794	0.7830
0.4	0.4	0.9682	0.7652
0.4	0.5	0.9657	0.7614
0.5	0.2	0.9775	0.7322
0.5	0.3	0.9676	0.7919
0.5	0.4	0.9505	0.7830
0.5	0.5	0.9508	0.7982

Table 3. Experiment Result 2b for Version 2 [3]

Learning Rate	Dropout2	Training Acc.	Validation Acc.	Test Acc.
0.001	0.2	0.8476	0.7147	0.7147
0.005	0.2	0.7301	0.7054	0.6491
0.0001	0.2	0.6926	0.7209	0.5977
0.0005	0.2	0.8379	0.7902	0.7230
0.001	0.3	0.8428	0.8065	0.7201
0.005	0.3	0.4694	0.5483	0.5121
0.0001	0.3	0.6662	0.7229	0.5965
0.0005	0.3	0.8258	0.7773	0.7084
0.001	0.4	0.8252	0.8237	0.7218
0.005	0.4	0.5554	0.6649	0.584
0.0001	0.4	0.5967	0.6745	0.5297
0.0005	0.4	0.8085	0.8211	0.7180
0.001	0.5	0.8169	0.7961	0.6988
0.005	0.5	0.4487	0.4559	0.4119
0.0001	0.5	0.4174	0.4283	0.3830
0.0005	0.5	0.7947	0.7906	0.7005

from 70% to 80%, so it can be seen that overfitting is occurring. We assume that the number of training data is insufficient.

In experiment 2b, we collected our own gesture data of the "wave" gesture, 100 positive data and $500 \times 2 = 1000$ negative data. Test Accuracy is relatively high when Dropout1=0.2 and Dropout3=0.2. The experimental result at that time are shown in Table 3. In general, neither Training Accuracy nor Test Accuracy is very high. We assume that the complexity of the neural network is insufficient to deal with this problem. A complicated neural network requires a large number of training data and long training time.

Since our goal is a user-customizable system, making the neural network more complicated is not appropriate for our policy.

9. Our System: Version 3

For Version 3 , we decided to make improvements based on Version 1. Figure 11 shows its overall view.

The design policy is as follows.

- As Version 1, it consists of two components. One is a component that recognizes distinctive

Table 4. Experiment Result of Version 3 [4]

	Training Acc.	Validation Acc.	Test Acc.
Florence 3D Actions Dataset	0.9892	0.9913	0.9838
home-made *wave* data	0.9627	0.9572	0.9622

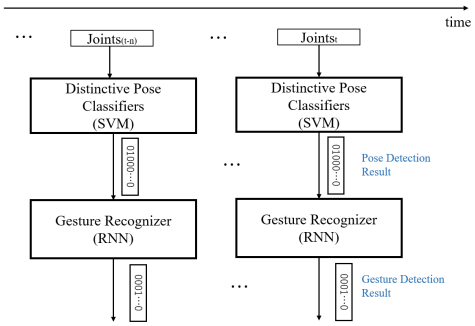

Figure 11. Diagram of Version3

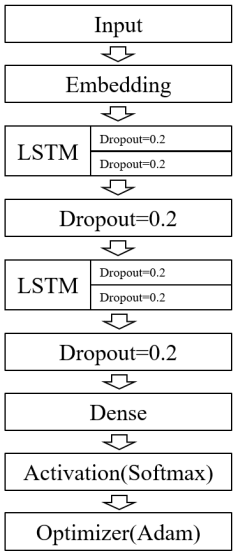

Figure 12. RNN Model of Version 3

poses. The other is a component that detects the appearance of a time-series of distinctive poses in each gesture.

- RNN is used for the component that detects the appearance of a time-series of distinctive poses. The model of RNN is shown in Figure 12.
- Skeleton data is preprocessed by normalization and quantization, and is represented by relative position relation (5-stage) of joints.
- Distinctive Poses are selected flexibly using k-means.

Using the experimental data used in Experiment 2a and 2b, we performed Experiment 3 to obtain the recognition rate of the system Version3. The result of Experiment 3 is shown in Table 4.

Test Accuracy and Validation Accuracy are both high, which means that the system can be trained properly.

However, it has been found that this system tends to misrecognize similar gestures. It is presumed that similar poses could not be well distinguished due to the influence of quantization (5-stage). Quantization (n-stage) enables high-speed training and high recognition rate of this system. The appropriate n value will depend on the set of gestures to be identified. It is necessary to prepare a set of gestures that are supposed to be used and determine the value n suitable for the set by further experiments.

10. Our System: Version 4

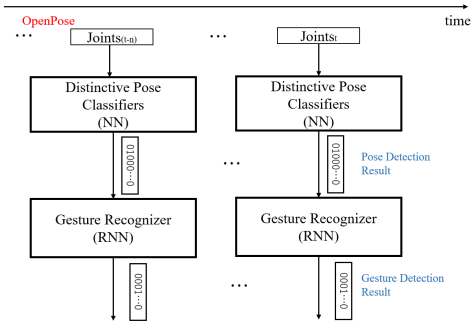

Figure 13. Diagram of Version4

Up to Version 3, we have developed our system using RGB-D camera, Kinect V2. However, from a practical point of view, it is preferable to recognize the skeleton using only the RGB camera. Therefore, the basic policy remains the same as Version 3, but we will change 3D skeleton recognition component to OpenPose [6] in Our system Version 4. Figure 13 shows its overall view. NN means Neural Networks.

Version 4 is currently under development.

11. Conclusion

In this paper, we discussed the gesture recognition system based on a non-contact sensor based skeleton recognition system. We have implemented several versions of gesture recognition systems by combining

methods that we think to be effective, and we have measured the recognition rate of those systems.

The Version 1 has the problem that the detection of time-series distinctive poses is not flexible because of the Exact Match algorithm.

The Version 2 system tried to perform gesture recognition in one RNN, which caused the problems with low recognition rates, requiring a lot of data and time for training, and being difficult to customize.

The Version 3 showed a fairly high recognition rate. However, it may work well due to the small n in n-stage quantization. If n is too small, it will be difficult to separate multiple types of similar gestures. So we need to find an appropriate n for the expected set of gestures.

The experimental results showed that a combination of several methods is useful. Useful methods are summarized as follows.

- Gesture Recognition System may be better separated into two components: One is a component that detects distinctive poses in Gestures, and the other is a component that detects the time-series appearance of them.
- Distinctive poses should be selected flexibly using k-means.
- RNN is suitable for recognition of time-series distinctive poses.
- Pre-processing of skeleton data is important: Rotation, Normalization, Polar Coordinates, Relative positional relations, Quantization (n-stage).

References

[1] Ministry of Health, Labour and Welfare, JAPAN, "About Coronavirus Disease 2019 (COVID-19)." https://www.mhlw.go.jp/stf/seisakunitsuite/bunya/newpage_00032.html, (2020/07/11 access).

[2] H. Ichikawa, S. Iijima, and Y. Nitta, "Natural User Interface using Gesture on VR Space," *the 178-th conference SIG Human-Computer Interaction*, 2018 *(in Japanese)*.

[3] H. Ichikawa and Y. Nitta, "A study of gesture recognition with RNN to operate VR space," *the 183-th conference SIG Human-Computer Interaction*, 2019 *(in Japanese)*.

[4] H. Ichikawa, "Research of Marker-less Gesture Recognition for Natural User Interface," *Master's thesis, Computer Science, Graduate School of Tsuda University*, 2020 *(in Japanese)*.

[5] Microsoft, "Meet Kinect for Windows." https://developer.microsoft.com/en-us/windows/kinect, (2018/06/08 access).

[6] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields." https://arxiv.org/abs/1611.08050, (2019/02/10 access).

[7] Y. Nitta, "NtKinect: Kinect V2 C++ Programming with OpenCV on Windows10." http://nw.tsuda.ac.jp/lec/kinect2/index-en.html, (2018/06/08 access).

[8] Y. Nitta, "NtKinectDLL - DLL and Wrappers (Unity C#, Python) for NtKinect." http://nw.tsuda.ac.jp/lec/NtKinectDLL/index-en.html, (2018/03/31 access).

[9] Intel, "Intel RealSense Technology." https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html, (2019/03/10 access).

[10] 3DiVi Inc., "Nuitrack." https://nuitrack.com/, (2020/06/10 access).

[11] Microsoft, "Visual Gesture Builder." https://msdn.microsoft.com/en-us/library/dn785304.aspx, (2018/06/08 access).

[12] D. Avola, A. Petracca, G. Placidi, M. Spezialetti, L. Cinque, and S. Levialdi, "Markerless Hand Gesture Interface Based on LEAP Motion Controller," *Proc. of the 20th International Conference on Distributed Multimedia Systems: Research papers on distributed multimedia systems, distance education technologies and visual languages and computing*, pp. 27–29, 2014.

[13] E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante, "A human activity recognition system using skeleton data from rgbd sensors," *Computational intelligence and neuroscience*, vol. 2016, p. 4351435, 2016.

[14] A. Taha, H. Zayed, M. Khalifa, and E.-S. El-Horbarty, "Human activity recognition for surveillance applications," 05 2015.

[15] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," *CoRR*, vol. abs/1801.07455, 2018.

[16] M. Omran, C. Lassner, G. Pons-Moll, P. V. Gehler, and B. Schiele, "Neural body fitting: Unifying deep learning and model-based human pose and shape estimation," *CoRR*, vol. abs/1808.05942, 2018.

[17] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition," *CoRR*, vol. abs/1904.12659, 2019.

[18] Google Brain Team, "TensorFlow, Google Brain Teadm." https://tensorflow.org/, (2020/02/02 access).

[19] F. Chollet, "Keras Documentation." https://keras.io/ja/, (2020/02/02 access).

[20] MICC Media Integration and Communication Center, University of Firenze, Italy, "Florence 3D actions dataset." https://www.micc.unifi.it/resources/datasets/florence-3d-actions-dataset/, (2020/07/11 access).