Hand Gesture Recognition Based on Keypoint Vector

Heru Arwoko
Departement of Electrical Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
7022211028@mhs.its.ac.id
Departement of Informatic Engineering
Universitas Surabaya
Surabaya, Indonesia
heru_a@staff.ubaya.ac.id

Eko Mulyanto Yuniarno
Departement of Electrical Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
eko_mulyanto@ee.its.ac.id

Mauridhi Hery Purnomo
Departemen of Electrical Engeering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
hery@ee.its.ac.id

2022 International Electronics Symposium (IES) | 978-1-6654-8971-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/IES55876.2022.9888333

***Abstract*—Human-computer interaction (HCI) is usually associated with using popular input devices such as a mouse or keyboard. In other cases hand gestures can actually be useful for human-computer interaction when hand gestures are needed to make the game controls more interesting. There are three basic controls as input mouse: move, click, and drag. Hand gestures and hand shape are different for each person. This becomes a problem during automatic recognition. Recent research has proven the success of the Deep Neural Network (DNN) for representation and high accuracy in hand gesture recognition. DNN algorithms can study complex and non-linear relationships between features by applying multiple layers. This paper proposes hand feature based on the normalized keypoint vector using DNN. The model was trained on 2250 hand datasets which were divided into 3 classes to identify the mouse movement. The network design uses multilayer with neuron sizes (13, 12, 15, 14) with 500 epochs and achieves the best accuracy of 98.5% for normalized features. The important work in this research is the use of keypoint vector from hand gestures as features to be fed to the DNN to achieve good accuracy.**

Keywords—Hand Gesture Recognition, Keypoint, Normalized Vector, Deep Neural Network.

I. Introduction

Human and computer interaction is usually uses mouse and keyboards [5]. Hand gestures can be used as a means of the interaction to make it more enjoyable, especially the playing of computers games. For kid games where hand movements make the game more interesting. Communication between devices can help in building a comfortable user interface for many applications [2]. Various automatic devices have used hand gestures as controls. The physical movement of the human hand produces movement and the recognition of hand gestures has led to advances in automated vehicle movement systems [3]. Along with the advancement of device control, the implementation of hand gesture recognition is increasingly being used as a basic human-computer interaction. However, considering environmental constraints, light sources, enclosed areas, and other factors, the diversity and complexity of gestures has a major impact on gesture recognition [4]. In many applications, the hand occupies only about 10% of the image, Spatial localization of the hand in such a scenario can be a challenging task [6]. Hand gestures vary in finger orientation and hand shape in different people. The problem that arises is the size of the hand and the shape of the hand that varies from people of different ages. So that is one of the characteristics of hand gestures that must be solved.

This method extracts various types of hand trajectory features and their various combinations to represent hand gestures. A common problem encountered in hand gestures is how to make computers understand and recognize hand gestures accurately.

In this paper, we propose the keypoint of hand gestures is used as an important feature in the recognition process. We choose a center point and five points on the fingertip as key points. Then a vector is formed that connects the center point to the point at the fingertip. We take the center of the hand as the center point of the formation of the fingers. So there are five vector orientations that represent hand gestures. The five feature vectors are normalized with the consideration that the features are robust against changes in scale and angle of inclination of various hand movements. After finding the features, the next step is to classify. The main problem is how to use this feature to get high accuracy.

In this paper, we use MediaPipe framework [16] to find keypoint hand gesture and deep neural networks are used to classify features. Then we compared the accuracy of the normalized versus non-normalized hand shape orientation features. It can be shown that the normalized feature vector has a higher accuracy. This research emphasizes on static motion recognition for automatic recognition of hand movements that can be applied to various fields such as robotics, virtual reality, sign language, and as game control.

II. Related Works

Various studies have been reported to the recognition of hand gestures using the shape context method by utilizing 3D information locally and globally from hand movements [1],[15]. This method extracts features based on the 3D shape context structure at various scales. The feature is noise-resistant and has a greater range of articulation of the hand shape. However, this method costs a lot of 3D extraction features using kinect sensors which require high equipment budget. Hand recognition system can also been developed using a neural network based on the shape fitting to resolve memory and power budgets [4][7]. Another approach for hand gesture recognition is used from finger angle estimation based on barometric pressure sensing [8] and also ultrasonic-based hand gesture recognition system. This works requires high cost equipment. Featuring a classification algorithm for temporal variations of movement was proposed [9]. Hand gesture recognition using both spatial and temporal features

for dynamic hand movement [10], [11], [12], [13], [14]. This work has been developed and used for hand recognition, due to its strong resistance to translation, rotation, and scaling. The most popular method for hand gesture recognition uses convolutional neural network (CNN) [1], [2], [3], [4]. This work requires analyzing a large amount of image data and analyzing the effects of data augmentation. Researchers have competed in terms of achieving accuracy and time efficiency in processing for hand gesture identification. We use the open source MediaPipe framework [16] as the base hand skeleton tracker. To improve robustness for translation, rotation, and scaling, we set up vector relations of key point positions as hand gesture features to feed the neural network.

III. Methodology

This section describes the process steps that were used. The step of methodology is shown on Figure 1.

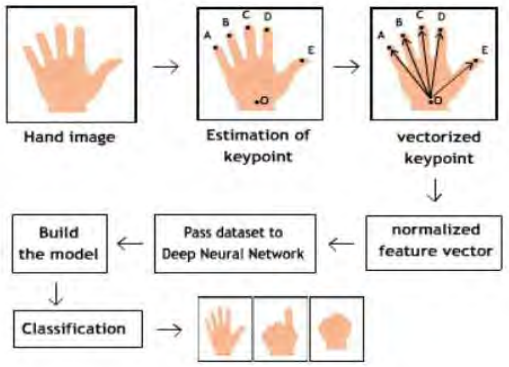


Fig.1. System Framework

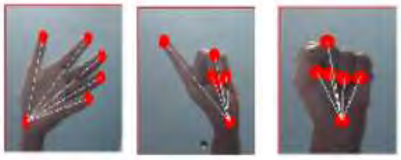
The experiment started from preparing the dataset, pre-processing to determine the hand area, determining the keypoint vector and building the DNN model. As shown in Figure 1, the image of the hand that was detected, 6 keypoint data were taken, the point O as the center of the hand formation and 5 other points (A, B, C, D, E) at the fingertips. Image of the detected hand, estimated hand center position and fingertip point. Then some vectors are formed connecting the center to each fingertip. Feature extraction is obtained from feature vector normalization. This is done so that this feature is robust on changing the scale and rotation of the hand gesture. The shape of the hand model can be represented from the predefined keypoint vector. The extraction features are fed to the neural network for classification of various hand gestures in the dataset. Next we perform various k-Fold Cross Validation, in this case we make k validation experiments from the results of the training data, which becomes the test data for each sample.

A. Input Data and Training Data

The first step in this research is to collect data directly from the camera. Data were collected in 2250 and divided into three classes: palm, index, and fist hand gestures. So for each gesture, 750 data were taken, which interprets as mouse input: move, click, and drag

B. Extraction Features from keypoint

Experimental data collection was arranged with various hand gestures, scales, hand tilt angles, and variations in distance from the camera. This is due to obtain very diverse training data, so it is expected to achieve high accuracy



(a) palm (b) index (c) fist

Fig. 2. Three hand gestures to identify the mouse state: mouse move (palm), click (index), and drag (fist)

As shown in Figure 2, various hand gestures are taken directly from the camera. Next, five keypoints are selected as feature vectors. Each vector is represented by two components of the x and y directions, so there are 10 vectors that represent the hand gesture feature in each image.

C. Normalized Vector Feature

The feature (X) obtained from the position vector at the fingertip, can be written as :

$$X = \{ \overrightarrow{OA}, \overrightarrow{OB}, \overrightarrow{OC}, \overrightarrow{OD}, \overrightarrow{OE} \} \quad (1)$$

if normalized, it can be written as :

$$\hat{X} = \frac{\{ \overrightarrow{OA}, \overrightarrow{OB}, \overrightarrow{OC}, \overrightarrow{OD}, \overrightarrow{OE} \}}{\sqrt{oa^2+ob^2+oc^2+od^2+oe^2}} \quad (2)$$

The performance of the accuracy results using the features in (1) and (2) will be examined and the results will be compared.

IV. Experimental Result

The first step is to perform a feature vector calculation on each keypoint using MediaPipe framework. Each vector represents a direction from the center to the fingertips. So every vector has an x direction and a y direction. Table I shows the orientation of the feature vector in the x-axis direction.

TABLE I. Keypoint Hand Gesture

Label	OA_x	OB_x	OC_x	OD_x	OE_x
0	-41	32	87	112	137
0	-47	61	132	157	155
0	-55	69	138	169	170
0	-72	58	140	163	162
…	…	…	…	…	…
1	-50	-39	-14	11	38
1	-45	-35	-9	12	37
1	-46	-37	-12	10	35
1	-45	-35	-11	11	36
…	…	…	…	…	…
2	-9	-1	-11	6	38
2	-8	-1	-14	2	36
2	-6	1	-13	4	38
2	-8	0	-14	5	40
…	…	…	…	…	…

This OA_x means the orientation vector joining the center point O to point A in the x-direction component. In the same way, calculations can be performed for OB_x, OC_x, OD_x, and OE_x. The y-direction component will

return OA_y , OB_y, OC_y, OD_y, and OE_y, so we get 10 features for this extraction method.

The feature as normalized vectors are fed to a neural network for classification. In order to build the best network model, we design experiments to perform various network models and we choose the model that can produce the best accuracy. To complete the analysis of this experiment, we do the process of normalizing the input vector to examine how much influence it has on the accuracy results achieved. The next step in building a DNN model is to perform layer variations. In this research, multilayer of neural network are used. By varying the number of neurons in each layer, the measurement results are obtained as shown in Table II with 3 layers.

TABLE II. BUILDING MODEL - 3 LAYERS

No	Layer 1	Layer 2	Layer 3	Accuracy
1	13	12	12	0.960
2	13	12	13	0.970
3	13	12	14	0.966
4	13	12	15	0.948
5	13	13	10	0.967
6	13	13	11	0.960
7	13	13	12	0.978
8	13	13	13	0.972
9	13	13	14	0.984
10	13	13	15	0.966
11	13	14	10	0.970
12	13	14	11	0.959
13	13	14	12	0.976
14	13	14	13	0.963
15	13	14	14	0.976

Then the experiment is repeated for the model with 4 layers as shown in Table III.

TABLE III. BUILDING MODEL - 4 LAYERS

No	Layer 1	Layer 2	Layer 3	Layer 4	Accuracy
1	13	12	13	13	0.971
2	13	12	13	14	0.967
3	13	12	13	15	0.970
4	13	12	14	13	0.958
5	13	12	14	14	0.959
6	13	12	14	15	0.978
7	13	12	15	13	0.966
8	13	12	15	14	0.985
9	13	12	15	15	0.956
10	13	13	13	13	0.972
11	13	13	13	14	0.975
12	13	13	13	15	0.961
13	13	13	14	13	0.968
14	13	13	14	14	0.971
15	13	13	14	15	0.969

The optimal network design was finally obtained by testing and trialing. Hence we chose the neural network design 4 layers (13, 12, 15, 14) with 500 epochs for our experiment because it provides the best accuracy. The last step is to carry out a dataset training process consisting of 2250 data, in which there are 3 classes, each class consists of 750 data. We did the k-Fold Cross Validation test to get a thorough test performance on the accuracy results achieved. As shown in Table IV, through the k-Fold Cross Validation, where the value of k is varied from numbers 5 to 10. This is meant to produce the best average accuracy of two types of features, features without normalization and features with normalization. It can be seen that the normalized features have significantly better accuracy than the features without normalization. Likewise with processing time, it can be seen that the normalized features have a more stable processing time during data training. In this experiment, it is shown that the vector features of the normalized finger formation position are superior to use as feature extraction in hand gesture recognition problems as shown in Table IV.

TABLE IV. CLASSIFICATION RESULT

k-Fold	Without Normalized		With Normalized	
	Accuracy	Time	Accuracy	Time
5	0.9515	19.72	0.9577	21.57
6	0.9488	22.49	0.9684	25.60
7	0.9604	29.37	0.9715	32.48
8	0.9617	34.16	0.9732	32.85
9	0.9537	35.19	0.9693	36.54
10	0.9622	40.31	0.9720	36.38

Next, the experiment was repeated using 50-50 split training-testing. It was observed that the accuracy for 50-50 splitting was 97.56% for normalized features, and 95.42% for non-normalized features. These results demonstrate the ability of the model to adapt to a larger test set.

The confusion matrices of the two models are provided in Figure 3 and Figure 4. This shows in the matrix that the values on the diagonal are very high. This means the recognition performance is good. The normalized feature model performs better in all three classes (palm, fist, and index) compared to the non-normalized model. Better performance for all classes on normalized features because it contains the right features.

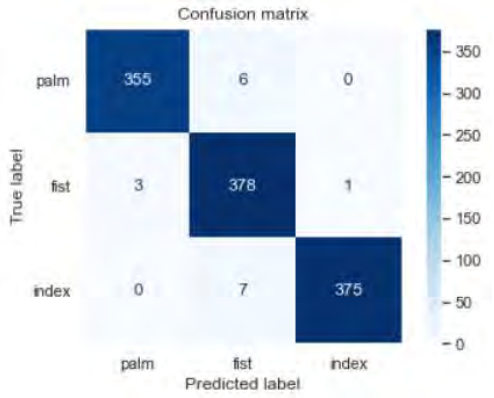


Fig. 3. Confusion Matrix Hand Gesture Recognition Database without normalized features

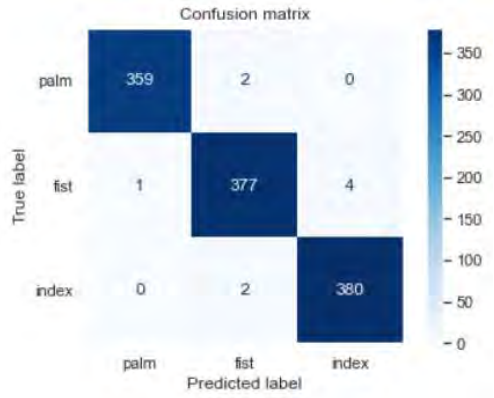


Fig. 4. Confusion Matrix Hand Gesture Recognition Database with normalized features

In general, it shows the confusion matrix of the normalized model has lower error results than the non-normalized model.

TABLE V. PERFORMANCE RESULT

Performance	Without Normalized	With Normalized
Precision	0.95	0.97
Recall	0.97	0.99
F-Measure	0.97	0.99
Accuracy	0.97	0.99

The performance measures shown in Table V prove this statement. It can be known that the normalized features have higher accuracy performance than the non-normalized features.

V. CONCLUSION

This study shows that keypoint vector can be used successfully as a feature to recognize hand gestures to be applied to computer controls such as mouse pointers. This research proves that the normalized keypoint vector feature yields high accuracy and good performance. The problem in hand gesture recognition can be solved by analyzing from the orientation of feature vectors using DNN. This experiment proves the hand gesture keypoint feature vector is a vector-based methodology that has a great influence on recognition accuracy. Even though the system can recognize the hand gestures successfully, however for some hand orientations have a recognition error and this becomes a challenging work in the future.

REFERENCES

[1] Z. Islam, M.S. Hossain, R.U. Islam, et al. "Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation", in Proceedings of International Conference on Imaging, IEEE, 2019, pp. 234-329.

[2] F. Zhan, "Hand Gesture Recognition with Convolution Neural Networks", in Proceedings of International Conference on Information Reuse and Integration for Data Science, IEEE, 2019, pp. 295-298.

[3] P. S. Neethu, R. Suguna, D. Sathish, "An efficient method for human hand gesture detection and recognition using deep learning convolutional neural networks", Springer, Soft Computing (2020), vol 24, pp. 15239-15248, 2020.

[4] Y. Luo,G. Cui, and D. Lia, "An Improved Gesture Segmentation Method for Gesture Recognition Based on CNN and YCbCr", Hindawi, Journal of Electrical and Computer Engineering, vol. 2021, pp.1-9, 2021

[5] C. Zhu, J. Yang, Z. Shao, at al. "Vision Based Hand Gesture Recognition Using 3D Shape Context", EEE/CAA, Journal of Automatica Sinica, Vol. 8, No. 9, pp.1600-1613, 2021

[6] P. Bao, A. I. Maqueda, Carlos, at al. "Tiny Hand Gesture Recognition without Localization via a Deep Convolutional Network", IEEE Transactions on Consumer Electronics, Vol. 63, No. 3, pp. 251-257, 2017.

[7] O. Köpük, A. Gunduz, N. Kose, at al. "Online Dynamic Hand Gesture Recognition Including Efficiency Analysis ", IEEE Transactions On Biometrics, Behavior, And Identity Science, Vol. 2, No. 2, pp. 85-97, 2020.

[8] P. B. Shull, S. Jiang, Y. Zhu, et al. "Hand Gesture Recognition and Finger Angle Estimation via Wrist-Worn Modified Barometric Pressure Sensing", IEEE Transactions on Neural Systems And Rehabilitation Engineering, Vol. 27, No. 4, pp. 724-732 , 2019.

[9] F. Zhou, X. Li, and Z. Wang , "Efficient High Cross-User Recognition Rate Ultrasonic Hand Gesture Recognition System", IEEE Sensors Journal, Vol. 20, No. 22, pp. 13501 - 13510, 2020.

[10] M.A Hammadi, G. Muhammad, W. Abdul, et al. "Hand Gesture Recognition for Sign Language Using 3DCNN", IEEE access Vol. 8, pp.79491 -79509 , 2020.

[11] K. S. Reddy, P. S. Latha, and M. R. Babu, "Hand Gesture Recognition Using Skeleton of Hand and Distance Based Metric", Springer, ACITY 2011, CCIS 198, pp. 346–354, 2011.

[12] S. S. Rautaray and A. Agrawal, "Adaptive Hand Gesture Recognition System for Multiple Applications" , Springer, IITM 2013, CCIS 276, pp. 53–65, 2013.

[13] B. Ionescu, D. Coquin, P. Lambert, et al. "Dynamic Hand Gesture Recognition Using the Skeleton of the Hand", Hindawi, Journal on Applied Signal Processing vol. 13, pp. 2101–2109:, 2005.

[14] Adam A. Q. Mohammed, Jiancheng Lv1, Md. Sajjatul Islam, "Multi-model ensemble gesture recognition network for high-accuracy dynamic hand gesture recognition", Springer, Journal of Ambient Intelligence and Humanized Computing, 2021

[15] C.H. Wu, W.L. Chen1, and C. H. Lin, "Depth-based hand gesture recognition", Springer, Multimed Tools Appl, vol. 75, pp. 7065–7086, 2016.

[16] MediaPipe Framework. https://mediapipe.dev, 2019.