Classifying Arm Movement with Embedded System and Machine Learning
Wonseok Choi, Jungsub Oh, Heesu Kim, and Jaehyun Kim
LIG Nex1, Yongin, Korea

Abstract
This paper presents a method to classify the arm movements into 8 categories for the sensor data acquired from a micro-electro-mechanical sensor. The method uses the attribute weighted KNN (AWKNN) which is a kind of machine learning algorithm. The measurement system consists of gyroscopes and an accelerometer, attached to a human arm to measure arm movement. The system has been implemented with field-programmable gate array. The sensor data are pre-processed and handed over the AWKNN-based machine learning classifier to classify the sensed arm movements into 8 classes. The training data sets have been collected from the group of men and women who had exercised the predefined arm movements. The developed arm movement recognition system has achieved the accuracy higher than 90% in the experiments.

Keywords: Machine learning, Embedded system, AWKNN, MEMS sensor

1. Introduction
Recent events like AlphaGo and IBM Waston have given a strong impression to the public about the future of technological development especially in machine learning. Such advances in deep learning have raised the expectation of machine learning techniques in various domains.

Human computer interaction (HCI) technology, for interaction between humans and machines, is also a prevalent issue. At present, HCI technology is in increasing demand, because it can remove the need to use a remote controller to control TVs, electric fans, lights, and so on. There are various types of HCI technology, and high expectations are being placed on such technology based on image. The Kinect sensor is an appropriate example to explain HCI technology. With Kinect, people are able to use their bodies to interact with games in a natural manner. The key enabling technology is human body language understanding: the computer must understand what a user is doing before it can respond [1].

In this study, we classify arm movements by applying attribute weighted KNN (AWKNN), which is a type of machine learning algorithm. Moreover, we determine the possibility of applying HCI technology using the classified information.

The human arm is equipped with low-cost micro-electro-mechanical systems (MEMS)-type sensors consisting of gyroscopes and an accelerometer. The information acquired by the sensors is transmitted to the machine learning calculation computer (MLCC) in real time by using a field-programmable gate array (FPGA). Transmitted information from the sensors is a “feature” of machine learning used in this study.

The remainder of this paper is organized as follows. In Section 2, we introduce the...
embedded system, which is composed of FPGA and MEMS sensors, for acquiring arm movement data. In Section 3, the test results from applying AWKNN are demonstrated, and finally, Section 4 draws conclusions.

2. Related Work

2.1 Learning Arm Movements with AWKNN

The KNN algorithm is commonly used in various fields in machine learning for classification and prediction, due to its simple structure and strong prediction abilities [2]. We note certain beneficial aspects of the nearest neighbor (NN) classifier: (1) Many other techniques (such as decision trees and linear discriminants) require explicit construction of a feature space, which is intractable for some distance functions. (2) The NN classifier can effortlessly deal with the hugely multi-class nature of visual object recognition. (3) From a theoretical point of view, it possesses the remarkable property that, under very mild conditions, the error rate of a KNN classifier tends to the Bayes optimal as the sample size tends to infinity [3].

The KNN algorithm can be divided into several cases, such as the simple KNN algorithm, back elimination, AWKNN, and instance weighted KNN. Back elimination is implemented by removing unnecessary features, while AWKNN assigns weights to features that affect classification, and instance weight KNN assigns weights to instances that affect classification [4].

It is challenging to classify classes that are made up of similar data, or different classes with similar features. In such cases with ambiguity and inaccurate information, the KNN algorithm cannot always produce quality solution due to its probabilistic nature [5]. The proposed method, hence employs the AWKNN algorithm to classify arm movements. Algorithm 1 shows how the classification works using the AWKNN algorithm. These steps

The proposed method, therefore, uses the AWKNN algorithm to classify arm movements to alleviate the limitation. The AWKNN algorithm is described in Algorithm 1, which is made of the steps for learning and testing.

3. Proposed Method

3.1 Embedded System Configuration

The system for acquiring human arm movement data should be small, so as not to interfere with the arm’s movement. For this reason, the system is composed of embedded hardware with FPGA, gyroscopes, and an accelerometer, as illustrated in Figure 1.

In the above system, the FPGA receives data from two gyroscopes and an accelerometer by means of real-time serial peripheral interface bus (SPI) communication protocol, and

| Algorithm 1. The AWKNN-based training and testing. |
|---------------------------------------------------|
| 1. Acquire the training data from each pre-defined class. |
| 2. Set the K-value, where K is the number of training data for comparison, used to determine the testing data class [6]. |
| 3. Set the learning rate, which can be thought of as a moving speed, to determine the optimal figure weight: |
| coefficient = coefficient − α * delta, |
| α = learning rate. |
| 4. Initialize all features’ weightings randomly in the training data. |
| 5. Classify the data into training and testing data for cross-verification. |
| 6. Select one set of data from the testing data. |
| 7. Determine all features of the data selected in step 6, and all the features of all training data by using the Euclidean distance: |
| \[ d(x, y) = \sqrt{\sum_{j=1}^{d} w_j^2 (x_j - y_j)^2}, \] |
| where \( d \) is the number of features, \( w_j \) is the weighting of features, \( x_j \) is the \( j \)th feature value in the testing data, and \( y_j \) is the \( j \)th feature value in the training data. |
| 8. Add all features’ distances in all training data and store the value. |
| 9. Select K training data with the shortest distance according to the K-value set in step 2, and classify the largest number of selected data classes as prediction classes. |
| 10. Control the feature weighting when the prediction and actual classes differ. |
| 11. Select another one set of data in the testing data, then repeat steps 6 to 11. |
| 12. Test all testing data. |
transmits sensor data to the MLCC via serial communication protocol.

The FPGA used in this study is a Xilinx Artix-7 model, manufactured in a 16 nm process and 15 mm in both width and height. In addition to providing a soft-core processor MicroBlaze in this tiny chip, it is also possible to design user logic. The power is supplied to the FPGA, gyroscope, and accelerometer was also designed using a switching regulator.

The MLCC plays the following roles:

- Receive three-axis angular velocity from the gyroscopes.
- Receive three-axis acceleration from the accelerometer.
- Learn training data, which is composed of 12 features (consisting of sensor data).
- Classify and predict arm movements by using testing data.

The roles of the FPGA are as follows:

- Receive sensor data in real time.
- Conduct sensor offset correction and temperature compensation.
- Transmit sensor data to the MLCC.

Figure 2 shows the actual shapes of the arm movement data acquisition system. Figure 3 shows the arm movement data acquisition system is on a man’s arm. FPGAs and MEMS-type sensors are utilized to the fullest, in order to make the system ultra compact and lightweight. With the development of semiconductor manufacturing technology, FPGAs are becoming increasingly smaller. In this study, an FPGA with a size of 10 mm × 10 mm was applied.

3.2 Displaying of Real-Time Movement Information

Figure 4 shows a graphic user interface (GUI) program for displaying and receiving sensor data in real time.

The GUI displays three-axis acceleration gyro information in real time. Through this GUI, a user can determine whether correct movement takes place every time the arm is moved. The sensor data received in real time is used to obtain the average and maximum values, and finally, the AWKNN input value is determined.

3.3 Classification of Movement Information and Recognition Method

Arm movement data can be described using data from the MEMS-type gyroscopes and accelerometer, which are com-
Table 1. The 12 features for expressing arm motion information

| Number | Feature                                      |
|--------|----------------------------------------------|
| 0      | Average X value of 3-axis accelerometer      |
| 1      | Average Y value of 3-axis accelerometer      |
| 2      | Average X value of 3-axis accelerometer      |
| 3      | Average pitch value of 3-axis gyroscope      |
| 4      | Average yaw value of 3-axis gyroscope        |
| 5      | Average roll value of 3-axis gyroscope       |
| 6      | Maximum absolute X value of 3-axis accelerometer |
| 7      | Maximum absolute Y value of 3-axis accelerometer |
| 8      | Maximum absolute X value of 3-axis accelerometer |
| 9      | Maximum absolute pitch value of 3-axis gyroscope |
| 10     | Maximum absolute yaw value of 3-axis gyroscope |
| 11     | Maximum absolute roll value of 3-axis gyroscope |

ponents of the embedded system. This data can also be used to make up a class’s features showing the arm’s movement information. Three-axis angular velocity from the gyroscopes and three-axis acceleration from the accelerometer are used for the 12 features. Table 1 displays the 12 features expressing arm motion information.

This information is transmitted to the MLCC in real time. The movement information starting point is not defined; only the ending point. The embedded system monitors the three-axis data from the gyroscopes. When all of the three-axis data is less than $\pm 5$ deg/sec, information of six features is transmitted to the MLCC. Figure 5 provides the data acquisition flow chart.

The arm movements are categorized into eight classes, as illustrated in Figure 6. The training set contains 125 data sets for each class.

4. Test and Results

For the sake of objectivity, the test group was divided into groups of men and women, and learning was performed with eight classes, with 125 instances of data for each class. Table 2 shows the accuracy for men and women when $K$ is 3 and the running rate is 5.

Figure 7 shows the feature weight of the women’s training data in our test. Feature 4 (average yaw value of three-axis gyroscope) has the maximum weight value, while feature 5 (average roll value of three-axis gyroscope) has the minimum weight value. From the results, we can conclude that the machine learning process is influenced to a great extent by the average yaw value of the three-axis gyroscope; however, the average roll value of the three-axis gyroscope does not have much of a significant effect.

Figure 8 shows the feature weights according to gender. As shown in Figure 8, although there are certain differences between men and women, they are generally similar.

The learning rate is simply a number, which can be divided into three categories, namely big (overshooting), small (undershooting), and equal, as in programming. The overshooting phenomenon occurs when the learning rate is too high. If the stride is too large, it may move to the opposite slope, past the lowest point, and in certain cases, it may rise instead of going down the slope. Overshooting refers to the phenomenon of climbing the slope and eventually escaping the rice bowl.

The undershooting phenomenon occurs when the learning rate is too low. Despite the fact that it has gone down for a long time, it remains far from the lowest point. Generally, moving a small amount is not a disadvantage, but this is a fatal drawback in the case of several tens or hundreds of thousands of repetitions, as with machine learning. In such cases, it may take more than a month to get to the bottom of bowl.

![Flow chart for data acquisition](http://dx.doi.org/10.5391/IJFIS.2017.17.2.91)
Figure 6. Arm behavior according to class. Class 0: no movement, Class 1: make a circle, Class 2: move up and then down, Class 3: move down and then up, Class 4: stretch and fold arm, Class 5: roll the arm right and left, Class 6: stab with rolling arm, and back, Class 7: make Mobius strip.

Adjusting the learning rate so that it is not too high is very challenging and requires a great deal of experience. In reality, overshooting is not a serious problem, because it can be noticed as soon as it occurs, while there may be a delay in noticing if the learning rate is low, as it appears to be going down normally. In both cases, it is preferable to develop a sense of appropriateness or inadequacy through multiple experiences [7].

If the class predicted during the learning stage differs from the actual class, the optimal feature weight is determined by adjusting the learning rate. Table 3 shows the effects of changing the learning rate on accuracy. Table 3 displays the result of cross-verification with only training data, while Table 4 provides the result of the actual test with testing data. Accuracy according to the learning rate is not significantly different when K = 3.

Table 3. Results of accuracy according to learning rate through cross-verification

| K | Learning rate | # of training examples | # of testing examples | # of attributes | # of classes | Accuracy   |
|---|---------------|------------------------|----------------------|----------------|-------------|------------|
| 3 | 0.001         | 890                    | 890                  | 12             | 8           | 98.6517    |
| 3 | 0.003         | 890                    | 890                  | 12             | 8           | 98.5393    |
| 3 | 0.005         | 890                    | 890                  | 12             | 8           | 98.5393    |
| 3 | 0.01          | 890                    | 890                  | 12             | 8           | 98.5393    |
| 3 | 0.1           | 890                    | 890                  | 12             | 8           | 98.5393    |
| 3 | 1.0           | 890                    | 890                  | 12             | 8           | 98.5393    |
Table 4. Results of accuracy according to learning rate for actual test

| K | Learning rate | # of training examples | # of testing examples | # of attributes | # of classes | Accuracy  |
|---|---------------|------------------------|-----------------------|-----------------|--------------|-----------|
| 3 | 0.001         | 890                    | 359                   | 12              | 8            | 93.8719   |
| 3 | 0.003         | 890                    | 359                   | 12              | 8            | 93.8719   |
| 3 | 0.005         | 890                    | 359                   | 12              | 8            | 93.8719   |
| 3 | 0.01          | 890                    | 359                   | 12              | 8            | 94.1504   |
| 3 | 0.1           | 890                    | 359                   | 12              | 8            | 94.1504   |

Table 5. Results of accuracy according to K for cross-verification

| K | Learning rate | # of training examples | # of testing examples | # of attributes | # of classes | Accuracy  |
|---|---------------|------------------------|-----------------------|-----------------|--------------|-----------|
| 3 | 0.005         | 890                    | 890                   | 12              | 8            | 98.5393   |
| 5 | 0.005         | 890                    | 890                   | 12              | 8            | 98.0899   |
| 7 | 0.005         | 890                    | 890                   | 12              | 8            | 97.6404   |
| 9 | 0.005         | 890                    | 890                   | 12              | 8            | 97.7528   |
| 13| 0.005         | 890                    | 890                   | 12              | 8            | 97.6404   |
| 20| 0.005         | 890                    | 890                   | 12              | 8            | 97.7528   |
| 30| 0.005         | 890                    | 890                   | 12              | 8            | 97.3034   |

Table 6. Results of accuracy according to K for actual test

| K | Learning rate | # of training examples | # of testing examples | # of attributes | # of classes | Accuracy  |
|---|---------------|------------------------|-----------------------|-----------------|--------------|-----------|
| 3 | 0.005         | 890                    | 359                   | 12              | 8            | 93.8719   |
| 5 | 0.005         | 890                    | 359                   | 12              | 8            | 94.429    |
| 7 | 0.005         | 890                    | 359                   | 12              | 8            | 91.3649   |
| 9 | 0.005         | 890                    | 359                   | 12              | 8            | 89.9721   |
| 13| 0.005         | 890                    | 359                   | 12              | 8            | 90.5292   |
| 20| 0.005         | 890                    | 359                   | 12              | 8            | 90.5292   |

depending on the value of K. Table 5 displays the results of cross-verification with only training data, while Table 6 lists the results of the actual test with testing data. As shown in Table 6, when K is 5, accuracy is high. This demonstrates that setting an appropriate K-value is important in AWKNN, as with providing weightings for features that affect class classification [8].

5. Conclusion

This paper presented a method to classify arm movements with three-axis angular velocity and acceleration data. Although there are some variance in the accuracy and weightings for gender and individuals, the proposed method has successfully classified arm movement of accuracy more than 90%.

Quality choice of K-value can help increase in accuracy. With a proper K-value, the proposed method has achieved the accuracy higher than 95%. Furthermore, we found that learning rate is an important factor in determining accuracy. In conclusion, the exercise information classification of the arm using the AWKNN algorithm was successful, despite the limitation of testing only eight classes. Therefore, we can classify arm movements by using the arm movement data acquisition system, which is composed of low-cost MEMS-type sensors. In the future, we plan to classify exercise information using deep learning, which is one of the machine learning methods. The concept of deep learning is to discover multiple levels of representation, with the hope that higher-level features can represent the more abstract data semantics. Such abstract representations learned from a deep network are expected to provide greater robustness to intra-class variability [9].

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

Acknowledgements

This work was supported by the PGM Tech R&D Lab at LIG Nex1.

References

[1] Z. Zhang, “Microsoft kinect sensor and its effect,” IEEE Multimedia, vol. 19, no. 2, pp. 4-10, 2012. https://doi.org/10.1109/MMUL.2012.24

[2] M. H. Kim, S. Lee, and D. H. Shin, “Predictability test of K-nearest neighbors (K-NN) algorithm: application to the KOSPI 200 futures,” Korea Journal of Business Administration, vol. 28, no. 10, pp. 2613-2633, 2015. https://doi.org/10.18032/kaaba.2015.28.10.2613

[3] T. Cover, “Estimation by the nearest neighbor rule,” IEEE Transactions on Information Theory, vol. 14, no. 1, pp. 50-55, 1968. https://doi.org/10.1109/TIT.1968.1054098

[4] S. Deokar, “Weighted K nearest neighbor,” Available http://www.d.umn.edu/~deoka001/downloads/K_Nearest_Neighbor_Algorithm.pdf
[5] L. Jiao, Q. Pan, X. Feng, and F. Yang, “An evidential k-nearest neighbor classification method with weighted attributes,” in Proceedings of 16th International Conference on Information Fusion, Istanbul, Turkey, 2013, pp. 145-150.

[6] C. M. Bishop, Pattern Recognition and Machine Learning, New York, NY: Springer, 2006.

[7] S. Kim, “Learning rate, data preprocessing, overfitting,” Available [http://pythonkim.tistory.com/23](http://pythonkim.tistory.com/23)

[8] K. Zheng, H. Wang, H. Li, W. Xiang, L. Lei, J. Qiao, and X. S. Shen, “Energy-efficient localization and tracking of mobile devices in wireless sensor networks,” IEEE Transactions on Vehicular Technology, vol. 66, no. 3, pp. 2714-2726, 2017. [https://doi.org/10.1109/TVT.2016.2584104](https://doi.org/10.1109/TVT.2016.2584104)

[9] T. H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, “PCANet: a simple deep learning baseline for image classification?,” IEEE Transactions on Image Processing, vol. 24, no. 12, pp. 5017-5032, 2015. [https://doi.org/10.1109/TIP.2015.2475625](https://doi.org/10.1109/TIP.2015.2475625)

Wonseok Choi is currently a senior research engineer at LIG Nex1. His research interests include embedded systems with hardware and software.
E-mail: choiwonseok@lignex1.com

JungSub Oh is currently a senior research engineer at LIG Nex1. His research interests include embedded systems with real-time operating systems.
E-mail: jungsboh@lignex1.com

Heesu Kim is currently a senior research engineer at LIG Nex1. His research interests are embedded systems with hardware and software.
E-mail: heesu.kim@lignex1.com

Jaehyun Kim is currently a research engineer at LIG Nex1. Her research interests are embedded systems and real-time operating systems.
E-mail: jaehyun.kim@lignex1.com