**ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika**

ISSN(p): 2338-8323 | ISSN(e): 2459-9638 | Vol. 8 | No. 3 | Halaman 683 - 692
DOI : http://dx.doi.org/10.26760/elkomika.v8i3.683 September 2020

# The Viability of Leap Motion Implementation in Controlling Drone using K-Nearest Neighbor Algorithm

**LISA KRISTIANA, HAFIDZ DAYU ADITYA**

Department of Informatics Institut Teknologi Nasional Bandung, Indonesia
Email: lisa@itenas.ac.id

*Received* 14 Juni 2020 **|** *Revised* 11 Juli 2020 **|** *Accepted* 16 Agustus 2020

**ABSTRAK**

*Pengendalian drone secara konvensional menggunakan joystik mengurangi fleksibilitas pergerakannya. Metoda pengendalian akan menjadi lebih bebas dan fleksibel dengan menggunakan pergerakan tangan. Metode pengendalian dengan pergerakan tangan ini menghasilkan data set dalam jumlah yang besar yang mengendalikan arah drone. Dengan alasan tersebut, Leap Motion Controller dibutuhkan untuk merekam dan mengenali contoh-contoh pose tangan dan mengekstrak data set. Metode pendekatan yang di lakukan adalah menggunakan algoritma K-Nearest Neighbor (KNN) untuk mengklasifikasikan nilai x, y, z, Pitch, Roll dan Yaw yang berdasarkan pergerakan pesawat konvensional. Riset ini fokus pada nilai akurasi dalam menerapkan peralatan Leap Motion dalam mengontrol arah drone dengan menggunakan algoritma KNN. Hasil eksperimen menunjukkan bahwa nilai k =3 menghasilkan tingkat akurasi sebesar 72.8%.*

***Kata kunci**: Drone Controller, Hand Gesture, K-Nearest Neighbor Algorithm, Leap Motion, K-value*

**ABSTRACT**

*Controlling a drone can be more entertaining and flexible by using a hand gesture compare to the conventional mode by using a joystick. However, a drone controlling using the hand gestures produce a large number of data sets that drive the drone's movements in particular. For this reason, a Leap Motion Controller is required to record and recognize the hand pose samples and extract the data sets. Our approach is to use the K-Nearest Neighbor (KNN) algorithm as our method in order to classify the x, y, z, Pitch, Roll and Yaw values which are based on the conventional aircraft motions. This research focuses on the accuracy value of implementing the Leap Motion device to control a drone with the KNN algorithm. The result shows that the k-values from 3 obtain 72.8% of accuracy*

***Keywords**: Drone Controller, Hand Gesture, K-Nearest Neighbor Algorithm, Leap Motion, K-value*

## 1. INTRODUCTION

A drone technology and its implementation have been developed during the last decade. **(Kim, 2017)**. There are several conventional methods to drive a drone such as a joystick and a mobile phone as controllers **(Rechy-Ramirez, 2018)**. In addition, a drone can also be controlled using a leap motion device **(Hadi, 2016)**. The leap motion device offers the flexibility of drone's movement since it relies on hand's gesture **(Sarkar, 2016)**. However, the motions that are produced by the hand's gesture lead to a complexity in terms of data samples and accuracy **(Weichert, 2013)**. Thus, this research focuses on implementing the leap motion in controlling a drone and addressing the complexity of hand's gesture recognition. The existing hand gesture recognition such as Support Vector Machines (SVM) and the Hidden Markov Models (HMM) only gains 12% accuracy **(Ren, Y., 2009)**. Based on that, our approach is to evaluate the K-Nearest Neighbor (KNN) method in order to determine the drone's movement. This paper consists of hand gesture recognition system which is provided in Section II. The KNN algorithm and its relevance to the drone movement are discussed in section III. The evaluation of our approach is presented in Section IV and followed by the conclusion which is provided in Section V.

## 2. LEAP MOTION-TO-DRONE CONTROLLING SYSTEM

The leap motion devices require inputs that are produced by the hand's gesture recognition system **(Hsiao, 2016)**. These inputs are the gesture-tracking of Pitch, Roll, and Yaw values. Concurrently, these values are classified by KNN algorithm that contains 2 processes *i.e.*, the training data and recognition process as shown in Figure 1.

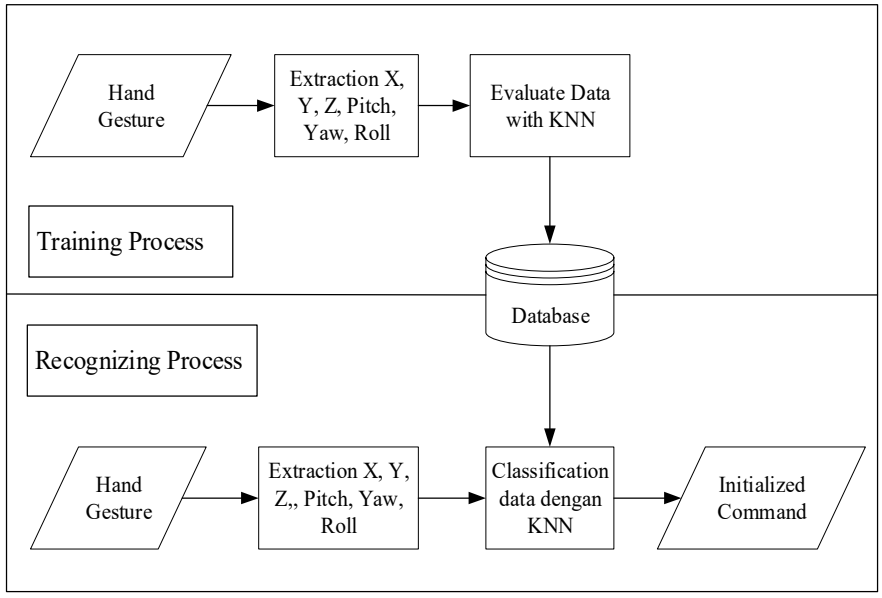



**Figure 1. System Diagram**

A training process is the process of collecting hand's gesture samples. These hand gesture samples, also known as Training data, are captured by the Leap Motion Controller **(Dzulkarnain , 2016)**. As the result, the Training data are extracted therefore the Pitch, Roll and Yaw values are obtained. These values are further evaluated using the KNN algorithm in

order to determine the data validity. The valid data is stored as a reference in recognizing process.

A recognizing process starts with a Test data sampling which has the similar algorithm with the Training data. As a result, the values which are evaluated by the KNN algorithm are classified into the same tone based on the stored training data. The outputs of this process are tones which have the highest similarity value to the training data.

## 2.1. The Hand's Gestures Classification

In this work, there are six hand's gestures which are used to classify the flying movement of the drone as illustrated in Figure 2.

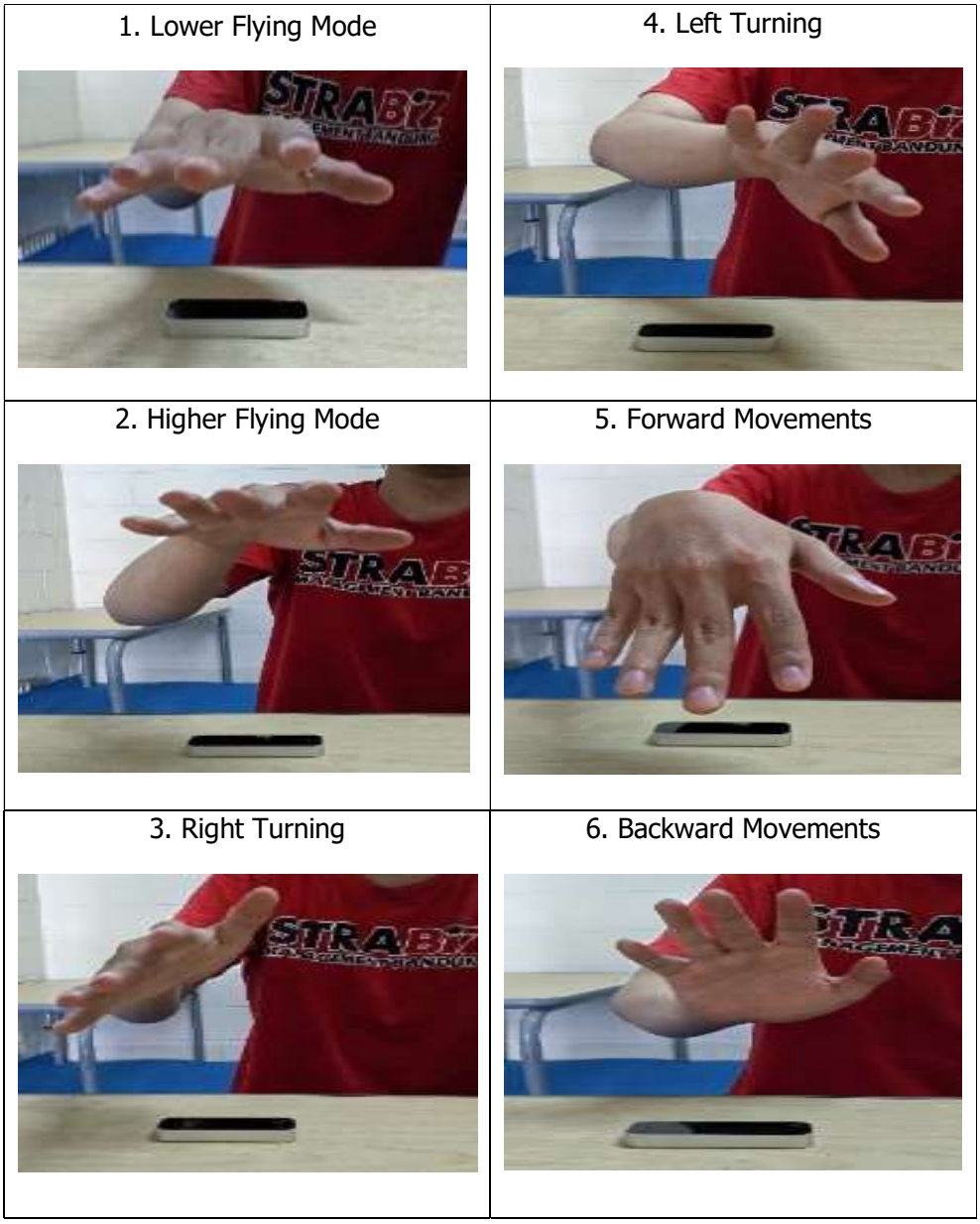
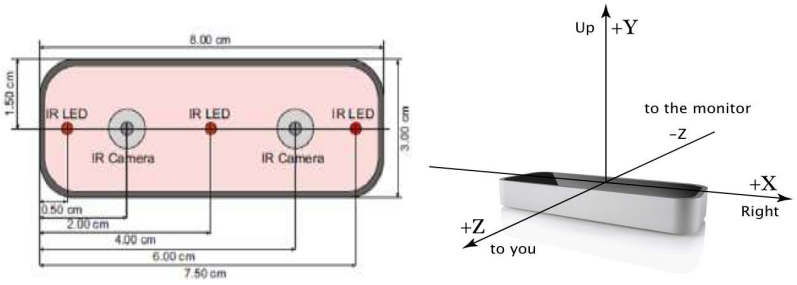



**Figure 2. The Six Classes of the Hand's Gestures**

## 2.2. Data Extraction in Leap Motion Controller

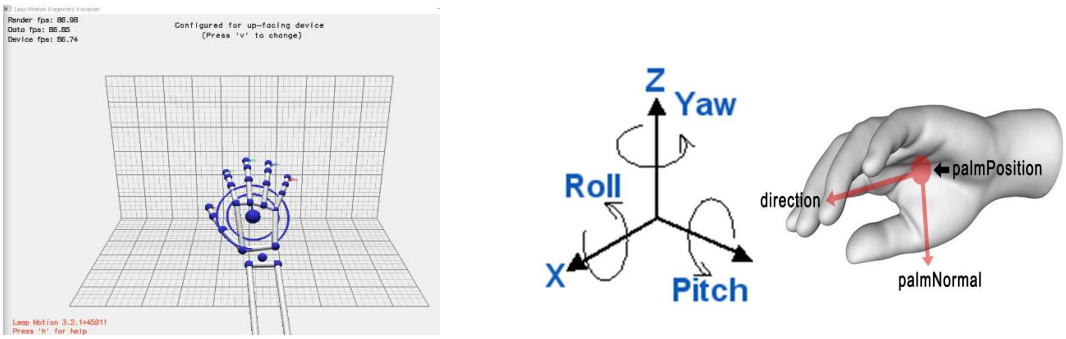
A Leap Motion Controller is a device that detects hand and fingers based on their current position. This device operates on 200 Frame per Second (fps) using 2 infrared cameras with high precision and 3 LEDs to capture the active palm range in a short distance about 1 meter. The Leap Motion uses the complex mathematics to extract the 3D position and comparing it to 2D frame which is generated by two cameras. The Leap motion controller is illustrated in Figure 3(a), shows the two infrared cameras along with three LEDs, which is implemented in this work.

The 3D coordinates as illustrated in Figure 3(b), are determined in Cartesian (x, y, z), thus the direction of "Up", "Right", "Left", "To monitor", and "To User" are defined. When the leap motion detects a hand and fingers, the detected images are tagged with unique IDs. The new ID will be applicable when the trace is disappeared. The hand detection illustration using the leap motion is shown in Figure 4(a). It indicates the right hand value with several blue points. In addition, the values of Pitch, Roll, and Yaw are obtained as follow: the Pitch value is the rolling movement in a y-axis, the Roll value is the rolling movement in a x-axis, and the Yaw value is the rolling movement in a z-axis (Figure 4(b)). Those values are represented in a palm position as illustrated in Figure 4(c).




**(a)** **(b)**

**Figure 3. (a) Leap Motion Controller Schematic, (b) Leap Motion Controller (Weichert, 2013)**




**(a)** **(b)**

**Figure 4. (a) Hand Detection on With Leap Motion Controller, (b) Pitch, Yaw, Roll and Palm Position (Weichert, 2013)**

### 2.3. K-Nearest Neighbor (KNN)

K-Nearest Neighbor (KNN) is a method to classify an object based on training data which are located closest to that particular object **(Croassacipto, 2019)**. This classification method considers the new object based on attribute and training data. Once a query point is determined, therefore, the K object and training points are learned based on the closest position to the determined query points. A prediction value is set based on the neighbour classification. With the obtained K value, thus the accuracy of training data and testing data are obtained.

The Euclidean distance is used to calculate the distance between neighbors and formulated in Equation (1).

$$D(a,b) = \sqrt{\sum_{k=1}^{d}(a_k - b_k)^2} \quad (1)$$

Where :
*D(a,b)* : Euclidean Distance Value (scalar)
*a* : training data
*b* : testing data (classification)
*k* : k (feature)
*d* : dimension (number of k)

Figure 5 shows the classification process in KNN algorithm. In KNN algorithm there are 2 processes, namely the process of preparing training data and test data (classification process) which are discussed as follow:

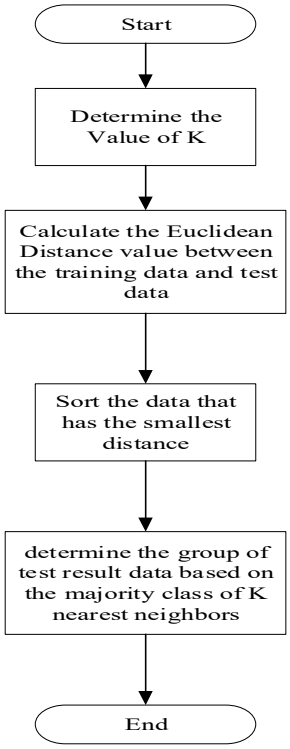



**Figure 5. Classification Process Flowchart With KNN Algorithm**

### 2.3.1. Training Data Preparation Process

The training data preparation process as shown in Figure 6(a) is used as a parameter to determine the navigation direction system according to the hand gestures. Where will be stored several sample data values Pitch, Roll, Yaw, *X, Y, Z* into the dataset.

### 2.3.2. Classification Process

Test data or classification process as shown in Figure 6.b is done by comparing the values of Pitch, Roll, Yaw, X, Y, Z received with the value of the previous training data stored in the dataset.

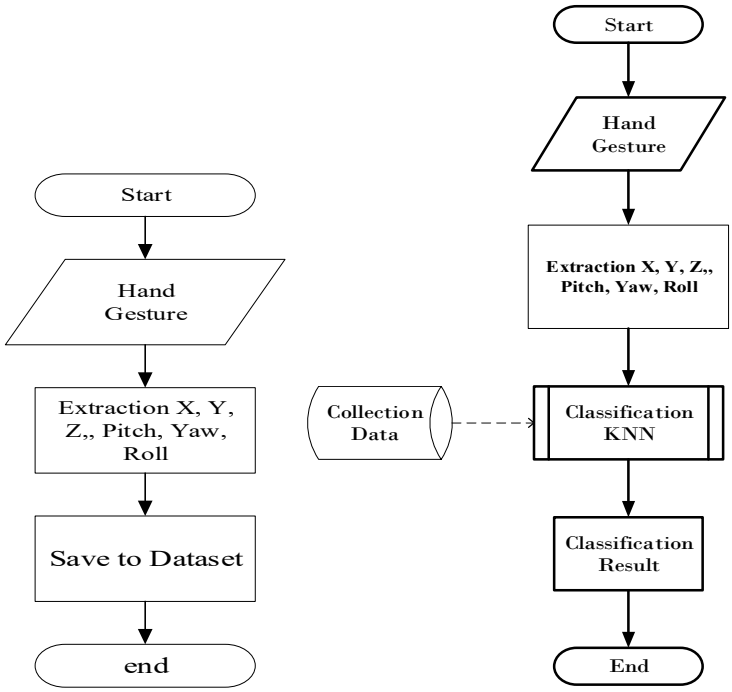



**Figure 6. (a) Training Data Preparation Process, (b) Classification Process**

## 3. TESTING AND EVALUATION

The real measurement of drone controlling using leap motion is applied with device's specification **(https://developer.leapmotion.com) (Birdayansyah, 2015)** as listed in Table 1.

As a result, Table 2 – Table 5 show the implementation of KNN algorithm with k-values as determined in range from k = 1 – 5. The numbers of data testing are obtained as shown in Table 3 until Table 6.

**Table 1. Device's Specifications**

| Description | Unit |
|---|---|
| Drone Flying Distance | 30 Meters |
| Battery Flying Time | 5 - 6 minutes. |
| Gyro | 6 axis |
| Quadcopter Size | 8.5x8.5x7.5cm |
| Arduino Uno | ATmega328 |
| LCD | 16x2 |
| Leap Motion Connector | LM01, USB |
| Leap Motion Performance | 200 reports per second |
| Leap Motion Accuracy | ± 0.00039 in |

**Table 2. Testing Result k=1**

| No. | Flying Direction | Numbers of Data Testing | Classification | | Accuracy | Error |
|---|---|---|---|---|---|---|
| | | | True | False | | |
| 1. | Throttle Up | 226 | 226 | 0 | 100% | 0% |
| 2. | Throttle Down | 284 | 284 | 0 | 100% | 0% |
| 3. | Throttle Left | 402 | 178 | 224 | 44.2% | 55.7% |
| 4. | Throttle Right | 278 | 46 | 232 | 16.5% | 83.4% |
| 5. | Throttle Forward | 304 | 304 | 0 | 100% | 0% |
| 6. | Throttle Backward | 216 | 216 | 0 | 100% | 0% |
| Average | | | | | 72.8% | 27.1% |

**Table 3. Testing Result k=2**

| No. | Flying Direction | Numbers of Data Testing | Classification | | Accuracy | Error |
|---|---|---|---|---|---|---|
| | | | True | False | | |
| 1. | Throttle Up | 226 | 226 | 0 | 100% | 0% |
| 2. | Throttle Down | 284 | 368 | 0 | 100% | 0% |
| 3. | Throttle Left | 402 | 176 | 226 | 43.8% | 56.2% |
| 4. | Throttle Right | 278 | 46 | 232 | 16.5% | 57.7% |
| 5. | Throttle Forward | 304 | 304 | 0 | 100% | 0% |
| 6. | Throttle Backward | 216 | 216 | 0 | 100% | 0% |
| Average | | | | | 72.2% | 27.3% |

**Table 4. Testing Result k=3**

| No. | Flying Direction | Numbers of Data Testing | Classification | | Accuracy | Error |
|---|---|---|---|---|---|---|
| | | | True | False | | |
| 1. | Throttle Up | 226 | 226 | 0 | 100% | 0% |
| 2. | Throttle Down | 284 | 284 | 0 | 100% | 0% |
| 3. | Throttle Left | 402 | 176 | 226 | 43.8% | 56.2% |
| 4. | Throttle Right | 278 | 47 | 231 | 16.9% | 83.1% |
| 5. | Throttle Forward | 304 | 304 | 0 | 100% | 0% |
| 6. | Throttle Backward | 216 | 216 | 0 | 100% | 0% |
| Average | | | | | 72.8% | 27.2% |

**Table 5. Testing Result k=4**

| No. | Flying Direction | Numbers of Data Testing | Classification | | Accuracy | Error |
|---|---|---|---|---|---|---|
| | | | True | False | | |
| 1. | Throttle Up | 226 | 226 | 0 | 100% | 0% |
| 2. | Throttle Down | 284 | 284 | 0 | 100% | 0% |
| 3. | Throttle Left | 402 | 167 | 235 | 41.6% | 58.4% |
| 4. | Throttle Right | 278 | 47 | 231 | 16.9% | 83.4% |
| 5. | Throttle Forward | 304 | 204 | 0 | 100% | 0% |
| 6. | Throttle Backward | 216 | 216 | 0 | 100% | 0% |
| Average | | | | | 72.3% | 27.7% |

**Table 6. Test Result k=5**

| No. | Flying Direction | Numbers of Data Testing | Classification | | Accuracy | Error |
|---|---|---|---|---|---|---|
| | | | True | False | | |
| 1. | Throttle Up | 226 | 226 | 0 | 100% | 0% |
| 2. | Throttle Down | 284 | 284 | 0 | 100% | 0% |
| 3. | Throttle Left | 402 | 167 | 235 | 41.5% | 58.5% |
| 4. | Throttle Right | 278 | 47 | 231 | 16.9% | 83.4% |
| 5. | Throttle Forward | 304 | 204 | 0 | 100% | 0% |
| 6. | Throttle Backward | 216 | 216 | 0 | 100% | 0% |
| Average | | | | | 72.3% | 27.7% |

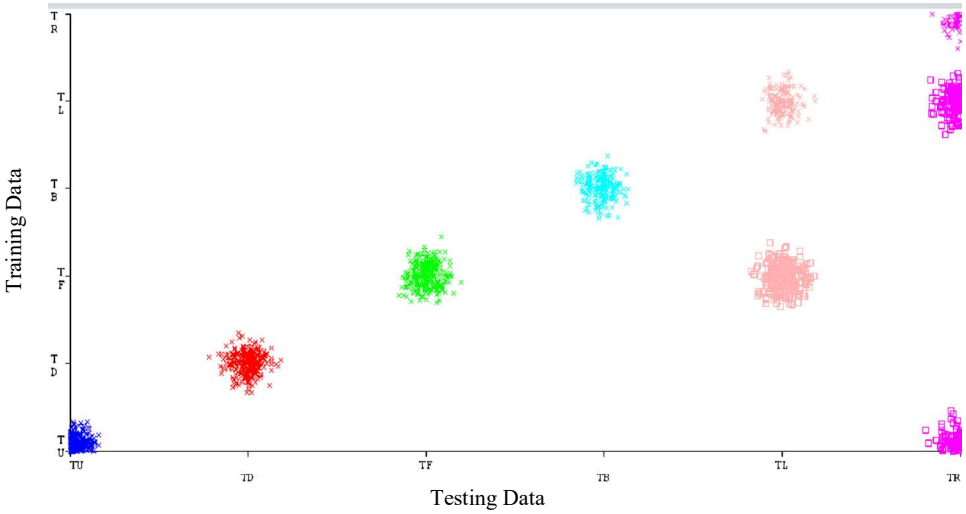



**Figure 7. Classification Diagram Result**

Based on the test results as shown in Table 2, Table 3, Table 4, Table 5 and Table 6, with the predetermined k, it was found that the optimal k value is k = 3 with an accuracy of 73.8% as visualized in Figure 7 for each flight direction. As indicated in Figure 7 the throttle up (TU) movement has 226 data that are appropriate visualized in dark blue color. For throttle down (TD) movement, it has 284 suitable data that are visualized in red color. For throttle forward (TF) data it has 304 corresponding data that are visualized in green color, for throttle back

(TB) data it has 216 data that is visualized in light blue color. For throttle left (TL) data from 402 recorded data, there were 176 data matches while 226 data were not suitable (235 data detected throttle forward) which were visualized in pink color. For throttle right (TR) data from 278 data recorded 47 data is true while 231 data is not suitable (150 data is detected throttle left and 81 data is detected throttle up) visualized in purple color. In overall, the drone flight in aforementioned directions i.e., TU, TD, TF, TB, TL and TD with 73.8% accuracy by implementing the leap motion.

## 4. CONCLUSION AND FUTURE WORK

This work managed to control the drone movements using the leap motion. The KNN algorithm achieved to classify hand poses based on the *x, y, z*, Pitch, Roll and Yaw values, specifically with k-value optimal spanned from k = 1 to k = 5. The accuracy value reached 73.8% under the position change states. However, the system experienced delays in command processing. In addition, the accumulated commands caused the halting operation in drone movement. In future, the higher speed processor and different data set platform will be integrated to overcome the delays and the accumulated commands.

## REFERENCES

Birdayansyah, R., Soedjarwanto, N., & Zebua, O. (2015). Pengendalian Kecepatan Motor DC Menggunakan Perintah Suara Berbasis Mikrokontroler Arduino. *Electrician*, *9*(2), 97-108.

Croassacipto, M., Ichwan, M., & Utami, D. B. (2019). Klasifikasi Nada Sesuai Kodàly Handsign Dengan Metode K-Nearest Neighbor Pada Leap Motion Controller. *Indonesia Journal on Computing (Indo-JC)*, *4*(1), 75-84.

Hadi, S. W., Setyawan, G. E. & Maulana, R., (2017). Sistem Kendali Terbang *Ar.Drone Quadcopter* Dengan Prinsip *Natural User Interface* Menggunakan *Microsoft Kinect.* Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK), 380-386.

Hsiao, D. Y., Sun, M., Ballweber, C., Cooper, S., & Popović, Z. (2016, May). Proactive sensing for improving hand pose estimation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems,* (pp. 2348-2352).

Dzulkarnain, I., Sumpeno, S., & Christyowidiasmoro. (2016). Pengenalan Isyarat Tangan Menggunakan *Leap Motion Controller* untuk Pertunjukan Boneka Tangan *Virtual. Jurnal Teknik ITS*, *5*(2).

Kim, J., Park, C., Ahn, J., Ko, Y., Park, J., & Gallagher, J. C. (2017, March). Real-time UAV sound detection and analysis system. In *2017 IEEE Sensors Applications Symposium (SAS),* (pp. 1-5).

Python SDK Leap Motion Documentation, https://developer.leapmotion.com/documentation. Last visited : Oktober 2019.

Rechy-Ramirez, E. J., Marin-Hernandez, A., & Rios-Figueroa, H. V. (2018). Impact of commercial sensors in human computer interaction: a review. *Journal of Ambient Intelligence and Humanized Computing*, *9*(5), 1479-1496.

Ren, Y., & Zhang, F. (2009). Hand gesture recognition based on MEB-SVM. In *2009 International Conference on Embedded Software and Systems,* (pp. 344-349).

Sarkar, A., Patel, K. A., Ram, R. G., & Capoor, G. K. (2016, March). Gesture control of drone using a motion controller. In *2016 International Conference on Industrial Informatics and Computer Systems (CIICS)* (pp. 1-5). IEEE.

Weichert, F., Bachmann, D., Rudak, B., & Fisseler, D. (2013). Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, *13*(5), 6380-6393.