Hand Gesture-based Wearable Human-Drone Interface for Intuitive Movement Control

Sang-Yun Shin
Dept. of Computer Science
Sejong University
Seoul, Republic of Korea
Email: kimshin812@sju.ac.kr

Yong-Won Kang
Dept. of Computer Science
Sejong University
Seoul, Republic of Korea
Email: kyw2539@sju.ac.kr

Yong-Guk Kim*
Dept. of Computer Science
Sejong University
Seoul, Repubic of Korea
Email: *ykim@sejong.ac.kr

***Abstract*—Although Radio Control (RC) has been a dominant device for controlling a drone, it is known that a fair amount of training period is required to master it. One way to sidestep such an RC-based control scheme would be utilizing either Kinect or Leap Motion sensor by which the user interacts with a drone more naturally. In such cases, however, the pilot has to hang around the sensor since the operating distance of such sensors is rather short. In this study, we propose a new wearable human-drone interface embedded on a Raspberry Zero, by which even a novice can let the drone not only take-off, land and fly to the intended directions according his hand-pose gestures but also make diverse flying trajectories such as circle, square and spiral using a sequence of hand gestures. Results from Gazebo simulator and several field experiments combined with a personalized calibration program demonstrate the feasibility of its commercial applications.**

I. Introduction

As Unmanned Aerial Vehicle (UAV) or drone is available in our daily life and it has diverse applications, usability becomes the main issue simply because maneuvering a drone is not an easy job for a novice user. From the first, RC has been a major user interface between the user and the drone because of its historical reason, *i.e.* RC car. However, as it is well known, it requires to spend a substantial amount of time to become an expert drone pilot. Recently, for instance, racing drone competition attracts many audiences, where dexterity in controlling a drone using a RC is a valuable asset.

On the other hand, since many people have expected for a more intuitive way to interact with the drone, researchers from the Human-Computer Interaction (HCI) and computer vision area have proposed some natural interfaces by which one can interact with the drone in a human friendly fashion[1], [2], [3]. In such case, smart sensors often play an important role. Among many, Kinect[4] has been a favored sensor since it is relatively cheap and diverse open source software is available[5], [6], [7]. In addition, Leap Motion sensor is also useful in controlling the drone because of similar reasons[8]. However, this kind of setup has some limitation. Since such sensor requires a PC to recognize users hand gesture and the operating range between user and the sensor is relatively short, the drone pilot has to be around the ground station and cannot walk away from it while controlling the drone. Another possible way to interact with the drone would be adopting a wearable device. Indeed, there was an interesting sensor, so called Myo, with which the user can make different commands by combining his hand gestures for controlling the drone[9]. We have thought that if such wearable device were a handy hand-held type, it would be much convenient and natural way to communicate with the drone. And yet, one potential problem when one would like to adopt these commercial sensors such Kinect or Myo, it requires a stand-alone computer. The wearable device that we present in this study contains both the necessary sensors and the computing processor. Given that recently high-performing embedded processors such as Raspberry Pi modules are available, we have designed a hand-held type wearable device for controlling the drone as well as for generating different figural trajectories. Basically, our wearable device consists of sensors module and a computing processor as shown on the right of Figure 1. We found that it has enough processing power with which it can carry out a fairly complex task such as signal processing, multi-layer neural networks and generation of the figural trajectory.

Fig. 1. (Left) an illustration of the present system where a user controls a drone with his hand pose by choosing direction control mode among 2 modes. (Right) Top views of the wearable device. It consists of Raspberry Pi Zero W and 6 axis accelerometer and gyroscope sensors. The size of the device is 84mm long x 42mm wide x 15mm thick

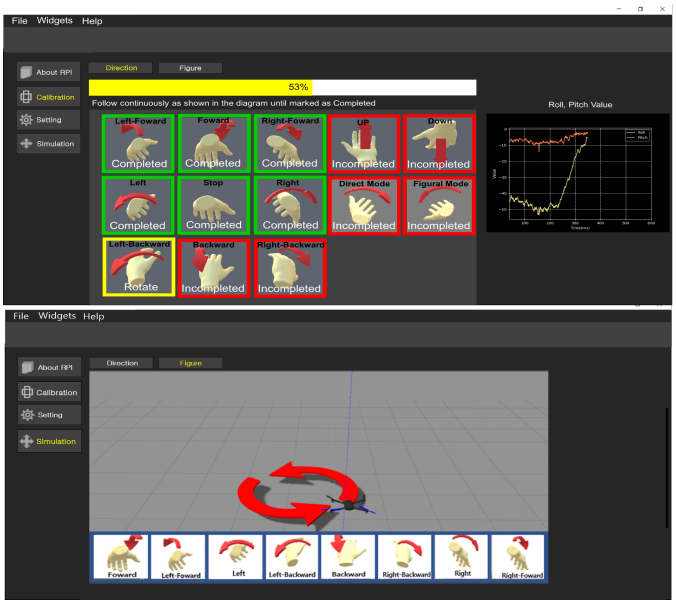

Fig. 2. There are two modes for controlling the drone: Direction mode, Figural mode. (Top) Gathering training data for the direction mode. With the guidance of the program, a userâĂŹs data is saved for training his or her classifier. There are 9 possible hand poses while palm down, and 4 possible hand poses while palm up; (Bottom) Practice for generating a circular trajectoriey in the simulation. All hand poses indicate the corresponding directions of UAV. Note that the user needs to make only one hand-pose for the directional control, whereas a sequence of hand pose gestures is necessary for making a figural trajectory.

In addition, our study differs from others because the user can make different figural flight trajectories using a sequence of hand poses as shown in the bottom of Figure 2. A typical way to make a flight trajectory with a drone would be designating a sequence of GPS coordinates on a PC or mobile device although it is not possible to do so in the GPS-denied area. As far as we know, there is no report where the drone can make diverse figural trajectories using one of the commercial sensors such as Kinect, Leap and Myo. With the present wearable device, the user can let the drone fly along one of several figural trajectories such as circle, triangle, square and spiral. In addition, the user is also able to adjust the scale of the figural trajectory. Research on trajectory generation for the UAV has been often related with obstacle avoidance or path finding. It is believed that many drone users would like to make such figural trajectories with their drones for fun. Then, the proposed system could be used for similar circumstance.

The proposed wearable device should be operating with any drones that support programmable control *i.e.* ROS (Robot Operating System). Therefore, Bebop 2 is chosen as our drone platform since it is compatible with ROS. In a similar vein, our wearable device can work with various commercial drones and Pixhawk based custom made drones, as almost all drones support programmable control.

II. THE WEARABLE DEVICE AND SOFTWARE

Our wearable device consists of a Raspberry Pi Zero W board, containing ARM 32 bits Single-core processor, 512MB RAM, 802.11 wireless LAN for Wifi communication, and MPU-6050 6 Axis sensor, having gyroscope and accelerometer. ROS Kinetic and Raspbian Jessi run for operating system, and Tensorflow is used for the neural network design. Bebop SDK and Mavros are installed for communicating with a FCU (Flight Control Unit) of Bebop 2, and Gazebo[10] simulator, repectively. Communication between the wearable device and drones uses Wifi. Python2.7, C++, C# are used for all programming.

III. HAND GESTURE RECOGNITION FOR HUMAN-DRONE INTERFACE

The close relation between gesture and mind (or human intention) has been suggested by many researchers including a leading figure like McNeil[11]. According to his categorization of gestures, there are two visual categories: iconic and metaphoric. Given that any human-drone interface is to faithfully convey the human intention into a movable artifact, *i.e.* UAV, hand gestures can be a very effective medium in carrying out the given task. In a recent survey on natural interfaces for the drone, human gestures are categorized into three groups: Imitative, Instrumented, and Intelligent[12]. The first two groups share several elementary gestures such as Up & Down, Left & Right, Rotate Left & Right, and Stop, whereas the third group contains high-level gestures like Find the object, Circle, Square, and Take-off & Land. Here, it is supposed that human utilizes either his hands or upper body for making gestures, suggesting that a visual pickup could be employed in capturing gestures.

However, our case differs from the above one since we adopt inertial sensors rather than visual one in recognizing hand gestures. And the size of gestures is relatively smaller than that for the visual sensor case. By considering other important factors while maneuvering the drone, such as stability during take-off, hovering and landing, and response time between a hand gesture and its reaction by the drone, we categorize our hand gestures into two groups. The first group is for orientation commands for controlling the drone directly, such as Up & Down, Right & Left, Forward & Backward, Forward-Right & Forward-Left, and Backward-Right & Backward-Left. The second group is for generating figural trajectories, such as Circle, Triangle, Square and Spiral as shown in the bottom of Figure 2 and Figure 4. Note that the Up command is also used for take-off and the Down command is used for landing, respectively. In addition, there are two extra commands with which the user can select a mode between two groups.

The present human-drone interface uses two MEMS sensors, *i.e.* a gravity sensor (or accelerometer) and a gyroscope. X, Y, and Z signals from the gravity sensor are input to a neural network 1(NN1). Figure 3 illustrates a flowchart where the raw signals from two sensors are processed through two neural networks and how both directional commands and figural trajectories are classified and generated, respectively.

First, the signals from MPU-6050 is used to measure acceleration of gravity with X, Y, Z coordinates. However, it is known that acceleration of gravity varies significantly when

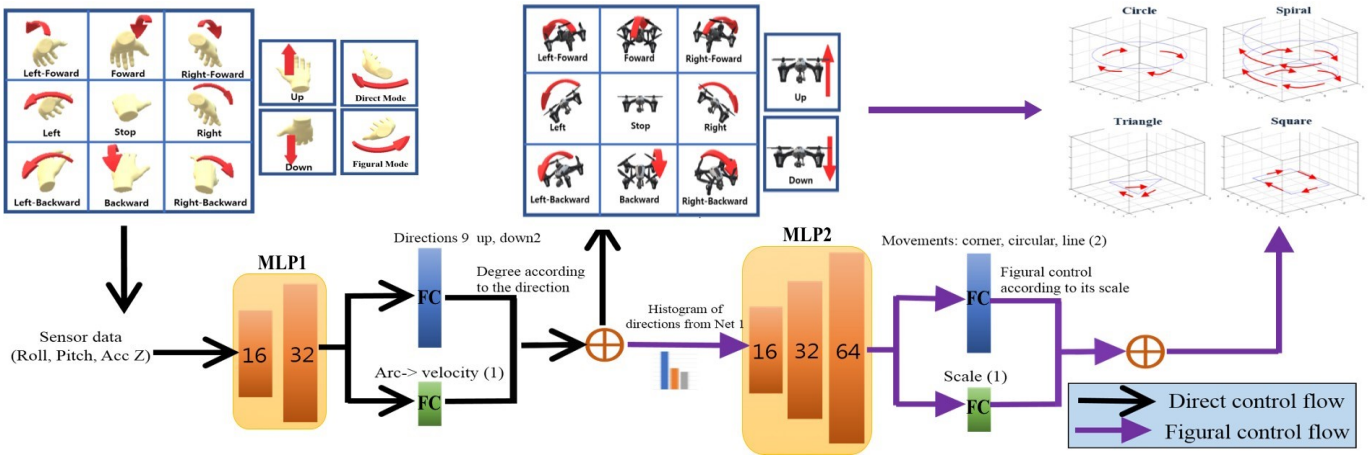

Fig. 3. Flowchart for generating orientation commands (or direct control) and figural trajectories with a wearable device: The raw data from sensor module MPU-6050 are processed through two serial neural networks: the first Multi Layer Perceptron(MLP1) followed by 2 Fully Connected layers(FC) classifies the hand pose given by the user, whereas MLP2 followed by FCs classifies figural trajectory when the hand poses are given as a sequence. Note that the user can make 9 hand poses during Palm Down and 4 hand poses during Palm Up, totaling 13 hand poses. Among them, 9 hand poses are used for directing the orientations of the drone. The hand pose Stop is used whenever any command for the figural trajectory is made in the start as well as in the end, whereas the hand pose Direct Mode, and Figural Mode are used in selecting the modes. ReLU is utilized for activations between layers in MLP1 and MLP2 blocks, whereas Sigmoid function is used for the last two separated layers(FCs).

one wants to use it for the hand pose estimation purpose[13], [14]. One potential way to deal with such problem, a combined module of a gyroscope and a magnetic sensor can be used[15], and yet one cannot escape from the noise problem. In this study, to handle such problem, we define 13 hand pose categories illustrated in Figure 3 and then a neural network is employed to classify the given hand pose gesture. Along with the classification of 13 zones, the arc value within the classified zone is also trained for the flexible control by measuring the degree of hand bending. For this training, we use the dueling architecture[16] originally proposed for deep reinforcement learning. In dueling architecture, fully connected layers in a neural network are separated to learn action values and state values, respectively, and then concatenated at the end to have one value representing the state in accordance with the action. It takes advantage of state value when predicting the action value. In our case, actions can be interpreted as directions or movements and the arc or scale can be as a state value. Apart from it's high performance by utilizing 2 separated layers at the end, it also has an advantage that a direction and the corresponding arc can be predicted using one neural network module. By defining the layer that measures the direction as $FC_{direction}$ and the layer that measures the arc as FC_{arc}, the output of Network1 is defined as follows:

$$Net1(X, zone; \theta) = FC_{arc}(X; \theta) + (FC_{direction}(X; \theta) - \frac{1}{|direction|} \sum_{zone} FC_{direction}(X, zone; \theta) \quad (1)$$

where θ is the weights of Net1, X is input sensor data that consist of roll,pitch and z, and $|directions|$ is the number of directions. Therefore, the zone according to the current sensor value is defined as $argmaxNet1(X; \theta)$ and the corresponding arc is $Net1(y = argmaxNet1(X; \theta)|X; \theta)$. To gather the training data for arc, we collect 2,500 data with labels ranging from 0 to 1.0 for every 0.1 unit. Note that the total number of the data is 32,500 for a person since each zone has 2,500. The target value for the training of direct control mode is:

$$Target = \begin{cases} arc & if Net1(y = currentzone|X; \theta) \\ 0 & if Net1(y \neq currentzone|X; \theta) \end{cases} \quad (2)$$

Then loss function is defined as follow:

$$Loss \frac{1}{n} \sum_{i}^{n} (Net1(X_i; \theta) - Target_i)^2 \quad (3)$$

Back-propagation and gradient descent optimization were performed using the loss value. When the current hand direction and the corresponding arc value come out as a classified zone with its value through Net1, the linear velocity and angular velocity change according to these outputs so that one can maneuver a drone in the corresponding direction (See section 4). After extensive training session, we measured classification accuracy through $argmaxNet1(X; \theta)$, and measured the arc value through Root Mean Squared Error(RMSE). Classification accuracy for 10 fold cross validation using 12 subjects' data was 98% and RMSE score for predicting arc value was 0.05. 100 gradient descent steps with Adam optimizer($\alpha = 0.01$) were performed using Nvidia Titan X GPU. The training time took about 1 minute in average. The average execution time for NN1 was 0.054 second. When a user controls a drone using the 9 directions and Up & Down, the control signal is transferred to the drone immediately. However, in case of mode change, the program is set to check if Net1 outputs the corresponding zone consecutively for 2 seconds to prevent unintended changes.

Secondly, we describe how the figural trajectories are generated using the hand pose sequences. 9 outputs(directions) from NN1 are used as input to neural network 2 (NN2), which

supposes to classify the figural trajectory(corner,circular). Here, the hand pose sequences are processed into frequency distribution corresponding to their directions from NN1. Thus, the processed data has the length of 9 where each of which has its frequency value from the hand pose sequences. This dictionary mapping approach allows to create a fixed-size trainable data, while making use of the feature within the trajectory made by oneâĂŹs hand. By distinguishing corner movement and circular movement of the hand, the path for the drone is generated differently. That is, the path is generated considering Yaw when classified as circular movement and not when it is corner. The path is made up of the stacked-output from NN1 when the user makes the trajectory by his or her hand. On the figural mode, the program counts stop signal from NN1 to confirm the beginning and end of the trajectory from the user. In other words, if the stop signal continues for more than a period, it recognizes that the gesture has begun to be drawn and starts recording the sensor data. If stop signal continues for more than 1 second again, recording the data stops and sequential output of NN1 is used for the path generation.

If one identical zone appears consecutively in NN1's output, a buffer stacks the ratio $r_{current} = \frac{frequency_{currentzone}}{frequency_{totallength}}$ sequentially as well. All ratios stacked in the buffer are multiplied by the scale factor to determine how many seconds to move in the corresponding direction. That is, if one zone in a certain range comes out more consecutively than others, the drone will move longer in the direction corresponding to the zone and shorter if the zone appears less. When the input of NN2 is classified as corner movement, data in the buffer is interpreted as an extension of direct control and the program controls a drone through roll and pitch. In case of Circular, it controls Roll, Pitch, Yaw of the drone according to whether the sequence is clockwise or counter clockwise. For example, it is counter clockwise when it comes out in the order of Forward $\rightarrow$ Left forward and clockwise when it comes out in the order of Forward $\rightarrow$ Right forward. When NN2 determines which figural trajectory should be generated, the corresponding flight parameters are generated in real-time. For more detail on this, see section 4 drone control. The scale S of the trajectory is made in a similar way to the arc of NN1. The data is labeled with 5 scales for each movement (Circular, Corner) from 0 to 1, drawn gradually larger with the guidance of the program as shown in Figure 2. Therefore, similar to NN1, if the output of NN2 is defined as follows:

$$Net2(X, move; \theta) = FC_{scale}(X; \theta) + (FC_{move}(X; \theta)) - \frac{1}{|move|} \sum_{move} FC_{move}(X, scale; \theta)) \quad (4)$$

then the target of NN2 is :

$$Target = \begin{cases} scale & if Net2(y = currentzone|X; \theta) \\ 0 & if Net2(y \neq currentzone|X; \theta) \end{cases} \quad (5)$$

Loss function, learning parameter, and evaluation method are all same as the training of NN1(Direct control) mentioned above. The classification-accuracy measured using $argmaxNet2$ was 92%, and the RMSE score for predicting scale using $Net2(y = argmaxNet2(X; \theta)|X; \theta))$was 0.064. With Raspberry Pi Zero, the time taken for both prediction and path generation was approximately 0.95 seconds.

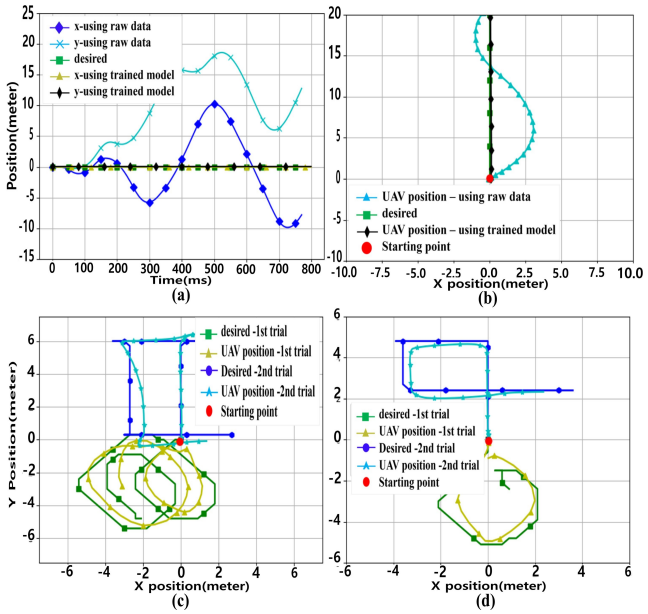

Fig. 4. Illustration of the desired trajectories and their corresponding drone trajectories simulated in Gazebo simulator. (a) and (b) show how the presence of NN1 stabilizes the movement of the drone using the identical input data. For the non-NN1 cases, the normalized raw roll and pitch data are used for the control the Stop (hover) (a) and the Forward direction movement (b), respectively. In (c), the user makes two desired movements: a rectangular and a spiral hand gesture. The corresponding drone trajectories are shown. Similarly, the user make a circular and rectangular movement, respectively, in (d) and their corresponding drone trajectories are shown.

IV. DRONE CONTROL

As mentioned above, there are two modes in controlling our drone: one is the direct control (or orientation commands) mode and the other is the figural trajectory generation mode. For both modes, it controls the linear acceleration and angular velocity of the drone[17]. First, for the direct control, Up and Down zones control the vertical velocity to make drone ascend or descend by setting V_{hand} to 1 or -1. Forward and Backward zones set $pitch_{hand}$ to +1 or -1. Left and Right zones set $roll_{hand}$ to -1 or +1, respectively. For the diagonal direction, a combination of these values are sent to the drone. Therefore, in the case of direct control, the commands sent to the drone after measuring the roll and pitch of the hand are as follows:

$$\begin{aligned} V_{drone} &= V_{hand} * max_{speed} \\ roll_{drone} &= roll_{hand} * arc * max_{angle} \\ pitch_{drone} &= pitch_{hand} * arc * max_{angle} \end{aligned} \quad (6)$$

Secondly, for the figural control, the Roll, Pitch, and Yaw commands sent to set the drone's attitude $pose_{drone}$ are:

$$\begin{aligned} roll_{drone} &= roll_{hand} * arc * max_{angle} \\ pitch_{drone} &= pitch_{hand} * arc * max_{angle} \\ yaw_{drone} &= \frac{rotationspeed}{S} \end{aligned} \quad (7)$$

In order to prevent the drone from moving too fast, $Max_{angle} and max_{speed}$ were set to 60 and 1, respectively. Each $pose_{drone}$ is sent to the Drone for $S * max_{second} * r_{current}$ time to reflect the scale. As mentioned above, the Yaw rotation value is only used in Circular control. In the experiment, we set the yaw rotation speed to +, - 0.5 according to clockwise and counterclockwise direction. However, if the rotation speed is too high, the drone may move within a small circle despite the large scale, and if the rotation speed is too low, it may not move within a circular shape. Therefore, the yaw value was adjusted according to the scale for the circular control. In Figural control, we did not use arc values for the better shape of the trajectory as the user intended. The above $roll_{drone}, pitch_{drone}, and yaw_{drone}$ are converted to Quaternions[18] and then sent to the drone to match the pose. ROS was used for transmission from the wearable device to the drone.

During the direct control case, given that all 9 possible directions and Up&Down commands are to move the drone into a specific direction, when the user makes a hand pose, it immediately applies to the drone without any waiting. We found, via user test, that this is very important simply because any user wants a fast feedback, *i.e.* visually recognizable movement of the drone, from his hand gesture.

For the figural trajectory case, the user needs to make a sequence of hand poses, such as Stop, Left forward, Left backward, Right, and Stop for a triangle trajectory, as shown in Figure 3. Note that every figural trajectory requires Stop signal from NN1 for the start as well as for the end, indicating that the NN2 has to wait until a pose Stop arrives in the end. And the network makes a decision which trajectory is given by incoming hand pose sequence. Then, process will generate the corresponding values of linear acceleration and angular velocity for making the flight. Figure 4(c),(d) show figural trajectories of the drone. Here, the desired path is calculated using the sequence of hand poses. For example, +1 or -1 is added to current y position in case of Forward & Backward direction, and -1 or +1 is added to current x values in the case of Left & Right directions, respectively. After the raw x,y trajectory is made, the scaling factors are multiplied to make the desired path. Figure 5 shows how our scaling algorithm works with rectangular and circular movement. Each figure contains three cases such as large, medium and small, respectively. The radius of a trajectory differs as scaling factor S from Net2 changes.

V. EXPERIMENTS AND RESULT

A. Experiements on Direct control

First, we have carried out a user test for the direct control mode with the present wearable device. The subjects consist of 3 different groups (n=15), such as expert-level drone pilots, students who know about drone, and naive subjects who do not have any experience of the drone before. The task was simple: each subject was asked to take-off, hover and make several directional movements such as left, right, forward and backward and finally land on the floor after a demonstration

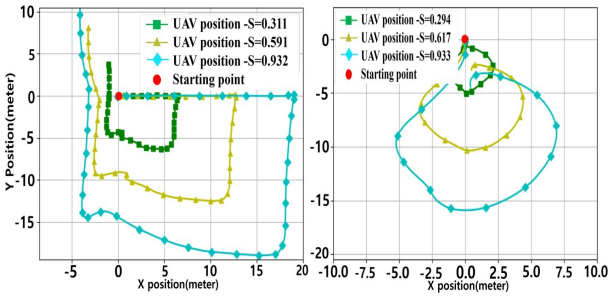

Fig. 5. Illustration of two figural trajectories with three different scaling factors performed using Gazebo simulator. $maxsecond$ is set to 5. (Unit: m)

by the experimenter. Result suggests that drone pilots (n=3) like the device because it is easy to control the drone and it has a quick response time. All students (n=9) participated in this test were able to complete the given task. Three naive subjects, who are middle age man and women, were also able to carry out the task. Also, we experimented on Gazebo simulator to see how the use of NN1 affects the movement of drones. Experimental results show that it is more stable to control the drone using the divided direction zone than using raw roll and pitch of the hand, as shown in Figure 4 (a),(b).

B. Experiments on Figural control

The second experiment is for the figural trajectory generation, and it is carried out to evaluate the performance of the proposed system in a indoor studio where the high definition motion capture equipment is installed. The subjects (n=2) are authors of this paper. The studio is measured by 14 x 12 x 4m, and 16 Eagle cameras, having 1230 x 1024 pixels resolution and operating with 500 fps, from Motion Analysis are installed for capturing the motion. Four visual markers are attached to the drone as well as the hand of the drone user, respectively. The position of the drone is automatically recorded as long as the markers are attached in x, y, z coordinates. Figure 6 shows how our drone makes different figural trajectories according to the given hand pose gestures such as square, triangle, circle and spiral, respectively. Here, each trajectory starts with a hovering operation as indicated with the dark color, and then follows a figural trajectory as illustrated as light green color. Note that each drawing of the trajectory appears to be very accurate probably because this experiment is performed within an indoor studio where wind disturbance is absent, and each trajectory is made by signal from the sensor and processor, not from any GPS signal at all. It is good to know how the hand gestures are made by the user. Since the markers are attached on his hand, the exact data of hand motions can be recorded by the same system. Note that each hand trajectory is very natural because the user makes an intended hand gesture in the air as shown in figure 6.

VI. DISCUSSION AND CONCLUSION

We present a new kind of wearable-based human-drone interface for controlling the drone. It has two modes: one

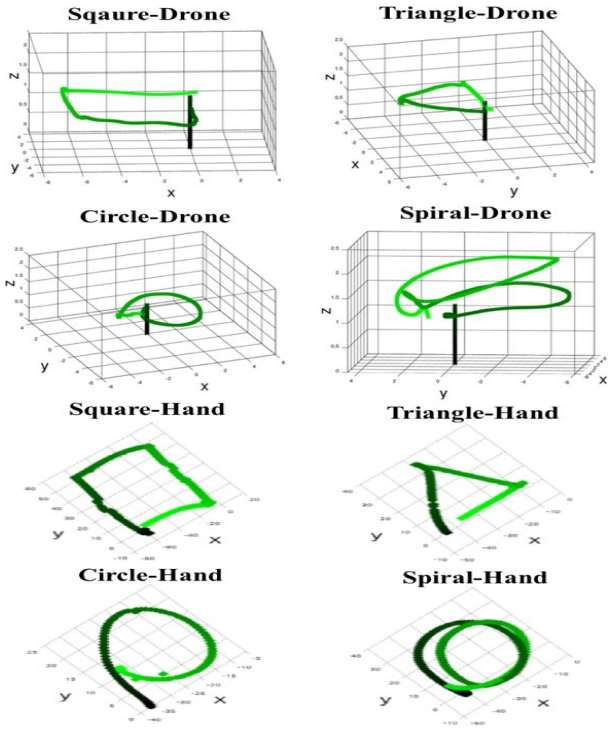

Fig. 6. (Top) Illustration of 4 different figural trajectories made by Bebop2 drone, and all data is acquired using the high-definition motion capture equipment. In each figure, the dark color indicates the starting point of each journey and the figural flight trajectory is shown as green color. maxsecond is set to 3 (Unit: m) (Bottom) 4 trajectories recording for the hand gestures where one can observe the natural trajectories by hand movement. (Unit: cm)

is to control the drone using the directional commands and the other is to generate diverse figural trajectories using a sequence of hand pose gestures. Contrast to previous studies on natural interface for controlling the drone, where they utilize Kinect, Leap or Myo, we use a handy wearable device attached on a palm to carry out the similar tasks. One of the advantages of the present device is that since it works well without any connection to a PC, the user can move freely while controlling the drone. In addition, since the hand pose gestures are designed in a very intuitive way, the user can easily learn, and move the drone according to his intention.

Result from the user test also suggests that most of subjects including experts and novices liked to use this device and to control the drone in a totally different way. For the figural trajectory case, because the user needs to understand the figural category and require a certain training period, we plan to investigate its usability in the future. As far as we know, this is the first attempt where one can draw an intended trajectory with the drone using a wearable device, including scaling factor in it.

All hardware using in this study is off the self, and all operating systems and libraries are open source based. Though we present the data using our present platforms Bebop 2 and the simulator, this wearable device can operate with other commercial drones and Pixhawk based drones.

ACKNOWLEDGEMENT

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018- 2016-0-00312) supervised by the IITP (Institute for information and communications Technology Promotion).

REFERENCES

[1] H. Beck, J. Lesueur, G. Charland-Arcand, O. Akhrif, S. Gagné, F. Gagnon, and D. Couillard, "Autonomous takeoff and landing of a quadcopter," in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*. IEEE, 2016, pp. 475–484.

[2] R. A. S. Fernández, J. L. Sanchez-Lopez, C. Sampedro, H. Bavle, M. Molina, and P. Campoy, "Natural user interfaces for human-drone multi-modal interaction," in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*. IEEE, 2016, pp. 1013–1022.

[3] J. A. Landay, J. R. Cauchard *et al.*, "Drone & wo: Cultural influences on human-drone interaction techniques," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2017, pp. 6794–6799.

[4] A. Sanna, F. Lamberti, G. Paravati, and F. Manuri, "A kinect-based natural interface for quadrotor control," *Entertainment Computing*, vol. 4, no. 3, pp. 179–186, 2013.

[5] P.-J. Bristeau, F. Callou, D. Vissiere, N. Petit *et al.*, "The navigation and control technology inside the ar. drone micro uav," in *18th IFAC world congress*, vol. 18, no. 1. Milano Italy, 2011, pp. 1477–1484.

[6] M. Chandarana, A. Trujillo, K. Shimada, and B. D. Allen, "A natural interaction interface for uavs using intuitive gesture recognition," in *Advances in Human Factors in Robots and Unmanned Systems*. Springer, 2017, pp. 387–398.

[7] L. V. Santana, A. S. Brandao, and M. Sarcinelli-Filho, "Outdoor waypoint navigation with the ar. drone quadrotor," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*. IEEE, 2015, pp. 303–311.

[8] G. Marin, F. Dominio, and P. Zanuttigh, "Hand gesture recognition with jointly calibrated leap motion and depth sensor," *Multimedia Tools and Applications*, vol. 75, no. 22, pp. 14 991–15 015, 2016.

[9] D. Jeremy, "Neuromuscular control system for small-unmanned aircraft systems," 2017.

[10] N. P. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator." in *IROS*, vol. 4. Citeseer, 2004, pp. 2149–2154.

[11] D. McNeill, *Hand and mind: What gestures reveal about thought*. University of Chicago press, 1992.

[12] E. Peshkova, M. Hitz, and B. Kaufmann, "Natural interaction techniques for an unmanned aerial vehicle system," *IEEE Pervasive Computing*, no. 1, pp. 34–42, 2017.

[13] Y. Thong, M. Woolfson, J. Crowe, B. Hayes-Gill, and R. Challis, "Dependence of inertial measurements of distance on accelerometer noise," *Measurement Science and Technology*, vol. 13, no. 8, p. 1163, 2002.

[14] A. Moschetti, L. Fiorini, D. Esposito, P. Dario, and F. Cavallo, "Recognition of daily gestures with wearable inertial rings and bracelets," *Sensors*, vol. 16, no. 8, p. 1341, 2016.

[15] X. Yuan, S. Yu, S. Zhang, G. Wang, and S. Liu, "Quaternion-based unscented kalman filter for accurate indoor heading estimation using wearable multi-sensor system," *Sensors*, vol. 15, no. 5, pp. 10 872–10 890, 2015.

[16] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[17] D. W. Mellinger, "Trajectory generation and control for quadrotors," 2012.

[18] Wikipedia, "Conversion between quaternions and Euler angles," http://en.wikipedia.org/w/index.php?title=Conversion%20between%20quaternions%20and%20Euler%20anglesoldid=860395080.