2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)
Bari, Italy. October 6-9, 2019

Smart Glove and Hand Gesture-based Control Interface For Multi-rotor Aerial Vehicles

Kianoush Haratiannejadi[1], Neshat Elhami Fard[2], and Rastko R. Selmic[3], *Senior Member, IEEE*

***Abstract*— This paper introduces an adaptable human-robot interface that uses two types of human-computer interactions: an image processing technique for a right-hand gesture recognition and a smart glove for left-hand commands. A fixed number of gestures is used for specific commands to the vehicle (take-off, land, hover, etc.), while the smart glove is used for the vehicle motors control. A single shot multi-box detector (SSD) model is used for a hand detection. After removing the cluttered background, the region of interest (RoI) is fed to a convolutional neural network (CNN) for right-hand gesture recognition. We propose three concurrent validation layers including a human-based validation. The validation layers allow the system to adapt to various users including different skin colors and hand shapes. Four flex sensors and a motion processing unit (MPU) are used in the smart glove to measure the bending ratio of each finger and the roll angle of the left hand. These signals are used for a left-hand gesture recognition as well as generation of continuous control signals such as throttle and angle commands of the vehicle. Extensive experimental results are presented that validate the proposed control methods.**

I. Introduction

Various techniques using wearable gadgets and gesture recognition have been used to advance humane-robot interaction methods including a wearable posture recognition glove to control robotic devices in disaster management [1] or wearable motion sensors to drive a virtual car [2]. In this paper we use gestures to develop a new control interface with self-validation for multi-rotor aerial vehicles.

We use a right-hand gesture recognition where a hand in each image frame is detected using a single shot multi-box detector (SSD) followed by a convolutional neural network (CNN) model that performs classification. The simultaneous left-hand control uses a smart glove with four flex sensors and a motion processing unit (MPU) that feeds data into a support vector machine (SVM) for left-hand gesture recognition. An advantage of this combined interface is that wearable devices, i.e. smart gloves, do not depend on environmental conditions such as light conditions or user physical features. The gesture recognition interface needs to be robust and adaptable enough to be able to handle various hand shapes and sizes.

A. Contribution

We propose an adaptable closed-loop system that can recognize and classify right-hand and left-hand gestures simultaneously. With its flex sensors and MPU, the left hand is also used to produce continuous-time signals that are useful for motors control (throttle and angles). We demonstrate how our method classifies gestures and how it deals with different undesirable situations such as showing gestures out of predefined shapes.

The proposed approach has the following novelties and advantages over existing methods: 1) The smart glove that produces 16 different gestures (commands) including two control signals which are used in the real-time control; 2) A hand recognition module before gesture classification improves the classifier performance; 3) Online learning which makes the system more adaptable to new users; 4) A self-validation of the classifier for improved reliability; and 5) A system validation with human-in-the-loop for the overall system reliability.

II. Background and Related Work

An effective human-robot interaction requires a gesture-vocabulary [3], [4]. While Bodiroža's implementation [3] used only right-hand gestures to interact with and command the waiters based on six gestures, our proposed method uses an image processing and self-validation technique with four gestures and a smart glove to classify 16 distinct gestures. The advantages of [5] and our approach in comparison to [6] and [7] are that the user does not have to stand while performing the gesture and suitability for real-time implementation.

We have combined two interaction approaches [4] and [8]: using a smart glove and an image processing classification method. The system recognizes the glove gestures by using a SVM which uses data from flex sensors and MPU, and simultaneously classifies right-hand gestures in each frame with three validation modules to validate and improves the hand gesture classification module performance. We use the SSD model for hand detection due to its simple training, suitable speed and accuracy, while the CNN model is used for gesture recognition.

In [9], Haar feature-based AdaBoost classifier is used to categorize five different gestures to control a drone. While the accuracy of their framework is highest once the operator poses within a specific distance which is 3 ft, our proposed method works at its highest accuracy until the operator poses the gesture within the camera sensing range which is 8.5 ft. Because, our method has a validation module which validates the classifier performance and if the classification accuracy is not higher than predefined value, validation module activates the re-trainer module that retrains the classifier and improves the accuracy. We have generated the CNN structure to

The authors are with Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, H3G 1M8, Canada. [1]k_harati@encs.concordia.ca, [2]n_elham@encs.concordia.ca, [3]rastko.selmic@concordia.ca

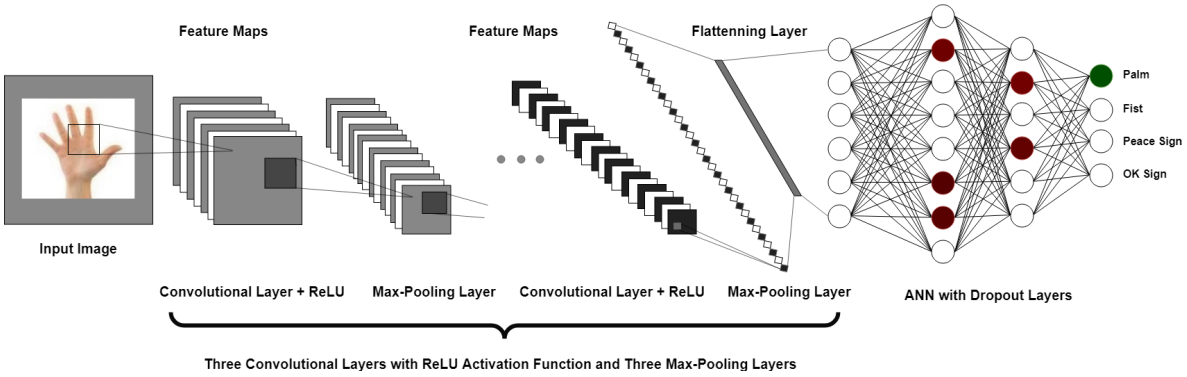


Fig. 1: Gesture recognition model architecture.

TABLE I: The number of layers used to design the CNN model in this paper.

NO.	*Layers*	*Number of Layers*
1	Convolutional Layer	3
2	Max-pooling Layer	3
3	Flattening Layer	1
4	Fully Connected Layer	2
5	Rectifier Activation Function	5
6	Dropout Layer	2
7	Sigmoid Activation Function	1

classify created datasets into four distinct classes for control of a multi-rotor aerial vehicle similar to [10].

A. Single Shot Multi-Box Detector Model

SSD is an object detection model with a two-component structure [11]. The model takes a picture as an input and categorizes its various objects into different classes. For image classification, it applies a base network including a standard architecture while for object detection it uses the additional feature layered structure.

In this paper, we use the SSD object detection model for hand tracking. The model has a layer in the base network section as well as four layers in additional feature layers with different scales. The layers in auxiliary component are sorted by size in descending order. Subsequently, a non-maximum suppression (NMS) algorithm [12] is used to merge all the detected regions related to the similar hand gestures. The NMS output is the input of the hand gesture classification module.

B. Convolutional Neural Network Model

CNN model consists of multiple layers of artificial neural networks (ANNs) stacked in a cascading form [13]. This feedforward network is comprised of convolution, max-pooling, flattening, and fully connected layers.

In the convolution layer, CNN extracts the feature maps from the input image using feature detectors. When feature maps are attached together, the convolutional layer is formed. Since the convolution operation is linear, the output of the convolution is passed through an activation function to make the output non-linear and a rectified linear unit (ReLU) is used as an activation function to enhance the non-linearity of the feature maps.

By using a max-pooling technique, unnecessary information is removed from the generated feature maps in the convolutional layer, thus reducing their dimensions. Finally, the fully connected layer is created by connecting each neuron to all neurons in the previous layer in order to classify the extracted features. CNNs also use dropout technique and a sigmoid activation function to prevent overfitting [14]. The number of layers in the CNN model that we used in this paper is shown in Table I, and the CNN model structure is illustrated in Fig. 1.

C. Smart Glove

Wearable gadgets for measuring body movements are a rapidly growing field. Collected data from hand movements and gestures can be converted to voices or commands. For instance, people with speech impairment have a problem communicating as most people do not understand the sign language. Our glove converts the hand gestures to a set of pre-defined commands similar to Bhaskaran *et al.* [15], which maps the sign language to a speech output by recognizing the hand gesture.

There are various devices that provide a human-machine interaction, namely joysticks, keypads, gloves, and more. A significant advantage of using wearable gloves is compatibility with natural human movement. Using fingers curvature and hand angles is an elegant solution for controlling robot movements. In [16], the authors made a connection between wearable gadgets and non-wearable ones, i.e., joystick, mouse position using random applicants. The results show that the above method is more consistent with natural hand movements, has better interaction, and is closer to what a user might expect from the robot behaviour.

To implement the smart glove, we use a glove with four flex sensors, one MPU module, and an Arduino UNO microprocessor board. Flex sensors are used to measure the bending ratio of each finger. Also, the MPU module calculates the angles of the hand in three axes. The microprocessor gathers all the raw data to analyze them in separate units. For instance, in Fig. 3, when the user makes a victory sign posture the microprocessor gathers the bending ratio of each flex sensor and then sends them to the computation unit. Afterwards, a machine learning (ML) model classifies

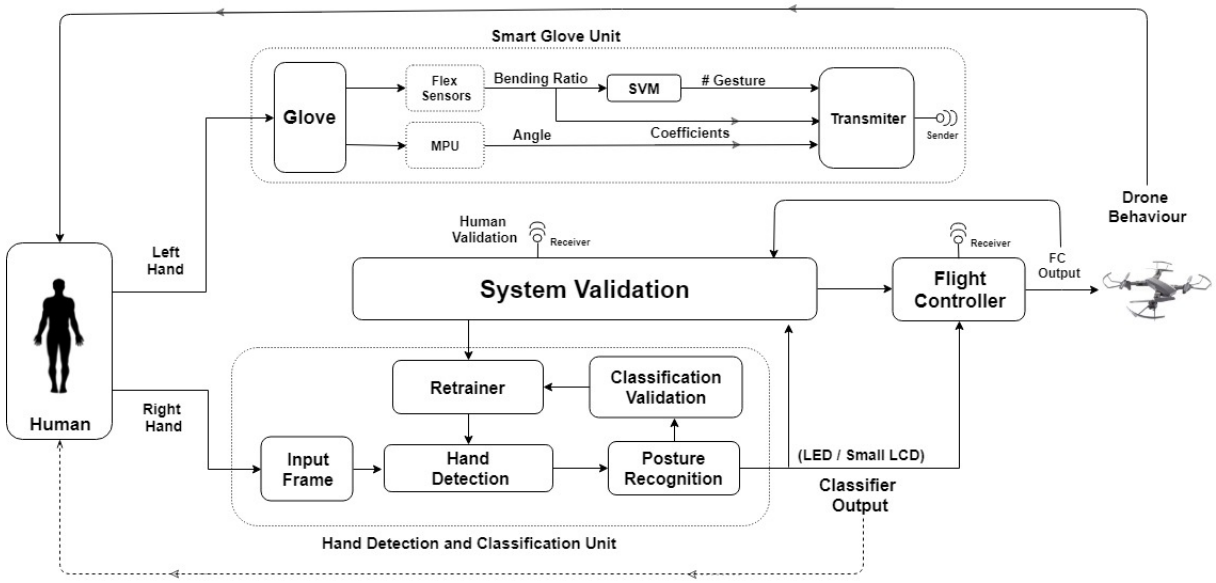


Fig. 2: System block diagram.

collected data into one of the 16 possible hand gestures.

III. Design Details

Here we describe a gesture-based, closed-loop control system with a classification validation. The user can also validate the output by observing the behaviour of the drone. If a misbehaviour has been detected, the validation module traces the source of a problem. As illustrated in the Fig. 2, the gesture-based, closed-loop control system contains four main units: a human-in-the-loop, a hand detection and gesture classification, the controller, and the system validation. Two types of gesture recognition are used to interact with an aerial vehicle: a smart glove-based and an image processing-based methods.

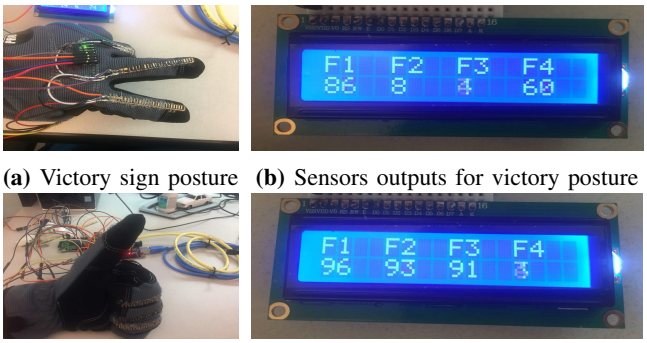
A. Gesture Recognition Module Using Glove

Flex sensors generate voltage signals as a function of applied strain. The first step is calibrating the flex sensors and the MPU. To calibrate the flex sensors, user must keep his/her fingers fully open and the algorithm assigns the flex sensors output values as zeros and after that the user must close his/her fingers and the algorithm assigns the outputs as 100%s. To calibrate the MPU, the user must keep the glove in a stable position and parallel to the ground.

After reading the data, the microprocessor broadcasts them via a transmitter to the flight controller and the system validation module, as shown in Fig. 2.

The smart glove operates between two different modes simultaneously. The first mode is considering each finger as an on/off switch and the second mode is using the exact bending ratio of each flex sensor.

1) Flex Sensors as On/Off Switches: The smart glove supports 16 different gestures. To analyze these gestures, the SVM model with a linear kernel is used. This model is trained using 7216 samples on four fingers in 16 different classes (commands). The output of the SVM model



(a) Victory sign posture **(b)** Sensors outputs for victory posture

(c) Thumb up posture **(d)** Sensors outputs for thumb up posture

Fig. 3: Posture with the smart glove.

determines that each flex sensor is on or off and what the command is then, the algorithm uses the exact bending ratio of each flex sensor in next mode. As shown in Fig. 2, these commands are separated into two class types to communicate with the system-validation module and the flight controller.

2) Bending Ratio: This mode determines the first coefficient for the chosen command. For instance, if gesture $\#12$ represents "turning left", fingers bending ratio determine the velocity of the rotation.

The system converts each gesture into a binary code. Each digit represents a flex sensor and the SVM output determines that each digit is on or off. For example, gesture $\#0$ (Thumb up, Fig. 3c) is coded as binary 0001. After determining each digit, the system considers the rate of changes $rs_j(t)$ for the corresponding flex sensors (those became ones) in each sampling time Δt as:

$$rs_j(t) = \frac{f_j(t) - f_j(t - \Delta t)}{\Delta t}, \tag{1}$$

where $f_j(t)$ is the corresponding bending ratios of the activated flex sensor j at time t. We determine the reference

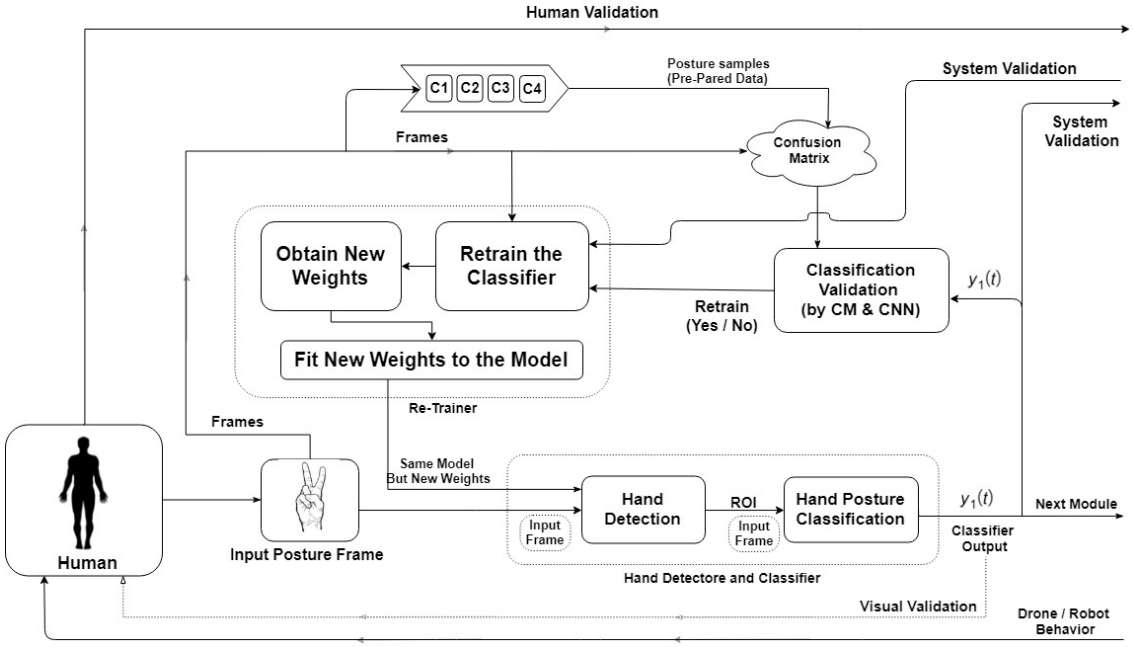


Fig. 4: Hand detection and gesture classification block diagram.

velocity of the drone, $v_G(t)$, as

$$v_G(t) = \frac{rs_j(t) - min\{rs_j\})}{max\{rs_j\} - min\{rs_j\}} + v_0, \qquad (2)$$

where v_0 is the velocity of the drone before each classification which we consider it as an initial velocity. In Section VI we show the effect of an activated flex sensor that corresponds to f_3.

The MPU module, with its gyroscope sensor, divides the space into three sections: 0° to 60°, 60° to 120°, and 120° to 180°. By holding the hand in different areas, the x-axis, y-axis and z-axis have different values and these values are used as the second coefficients for 16 hand gestures or separate coefficients for a certain parameter. For instance, we can define the pitch axis (z-axis) of the hand as the coefficient of the drone altitude. This does not change unless a command is given to change the state of the drone. If the user holds the glove pitch angle in the third area ($120^\circ - 180^\circ$), the altitude is rising. Also, if the user holds the glove pitch angle in the second area ($60^\circ - 120^\circ$), the altitude does not change. If the pitch angle in the z-axis drops to the range ($0^\circ - 60^\circ$), the altitude is decreasing.

B. Hand Detection and Gesture Recognition Module Using Image Processing

Right-hand gesture recognition consists of hand detection and gesture classification. The algorithm detects the hand in each frame and then classifies the detected gesture. We considered hand detection and gesture recognition by using the AUSDOM AW335 full HD 1080p Webcam. At first, the images are converted from RGB to grayscale thus reducing the image dimensionality and reducing processing time. These grayscale images are fed into the next section to find and grab the hand region if there is one in each frame.

We chose the SSD model for the hand detection because of its simple training, speed, and accuracy [11]. We trained the SSD model using N images with a size of $n \times m$ pixels from different users who were in front of a white background showing C pre-assigned gestures (in our implementation $N = 4600$, $n \times m = 640 \times 480$ and $C = 4$). After the hand has been tracked, the SSD model selects a rectangular region of interest (RoI) which includes the traced hand. The RoI and the corresponding frame are flattened and fed into another pre-trained model for classification. The classifier classifies various hand gestures in each frame. It only checks the position in the main frame which is related to the extracted RoI.

IV. Hand Gesture Classification

We use the CNN model to classify the output of the hand detection module into one of the classes illustrated in Fig. 6. To train the classifier, we used the Keras library. By using the Keras module and CNN model, each detected hand is classified into one of the four available classes as shown in Fig. 6.

Hand gesture classification unit has three main modules: a hand detector and classifier module, a classification validation module, and a re-trainer module. The primary input to this unit is a gesture frame, and the main output is the classified gesture as a class. The classifier has an internal validation module which checks and improves the classifier performance.

A. Hand Classifier

Upon receiving the input image, the first step is detecting a hand in the input frame, Fig. 4. The algorithm uses SSD model to detect the hand and CNN model to classify the detected hand into one of the predefined classes, respectively.

The classifier output maps to one of the pre-determined commands such as hover, land, turn right or turn left. Then, the output command is sent to the flight controller which is used to execute the command, the system validation module, and the classification validation module.

B. Re-trainer Module

As illustrated in the Fig. 4, re-trainer module is activated in two cases: (1) when classification validation module detects low confidence in the confusion matrix; and (2) when the system validation module sends the re-training penalty. After the re-trainer module is enabled, the re-trainer starts getting S new images from the user as its input (in our implementation and experimental results $S = 5$). Afterwards, re-trainer re-trains the running CNN model with the newly gathered data in order to obtain new weights. Then, re-trainer fits the new weights to the CNN model, and sends the CNN model to the classifier.

V. Validations

There are three validation modules in this system. The user is the first to validate the system output by observing the robot's behavior. Second, for the outer layer of the system, there is a validation module which finds the source of the problem that can be either with the classifier or the flight controller. Finally, there is a validation module for classifier output inside the classification module. These three validation modules work based on reward and penalty principles to improve the system performance.

A. Human Validation

The user is responsible for two tasks, as illustrated in the Fig. 2: (1) to feed the system with the hand gestures; and (2) to validate the drone's performance. Through his/her observation of the drone's behaviour the user decides if the system output is correct or not. In case when the user decides that the output is incorrect, the user sends a signal to the system validation module using the glove. If the problem is with the classifier, then the system asks the user to feed S new gestures to the camera.

B. Classification Validation

Classification validation module is responsible to check the classifier accuracy and, if necessary, to send the re-training command to the re-trainer. Classification validation module has access to the CNN model. To test the reliability of the classifier output, one needs a trustworthy source. We use d images that have been taken from the user during an initialization phase and the the system assigns their corresponding classes (In our implementation and experimental results $d = 10$). The validation module assumes that such data is correctly classified. Then the classification validation module uses the running CNN model to create the confusion matrix (CM). The classification validation module analyzes the CM to calculate running CNN classification errors.

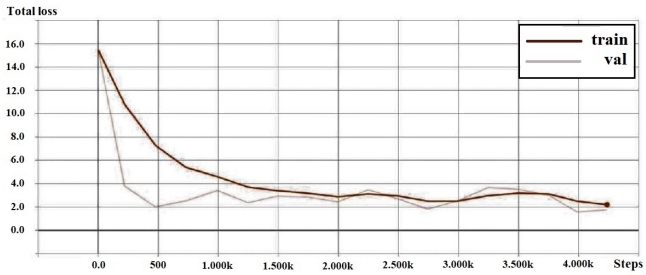


Fig. 5: Total loss of the training process with respect to the steps.

Let us define a classifier output reliability R_{C_i} as

$$R_{C_i} = \frac{x_{i,i}}{\sum_{j=1}^{C} x_{i,j}}, \tag{3}$$

where $x_{i,j}$ are elements of the CM (in our case $C = 4$). If the classifier output reliability is less than equal a predefined threshold, the re-trainer module re-trains the CNN model with new images and updates the CNN weights.

C. System Validation

Triggered by a human validation, the system validation module checks performance of the flight controller and the classifier output. The system validation first assumption is that classifier output is correct and the problem is with the flight controller. If flight controller check is passed, the initial assumption that the classifier classified correctly was wrong so the system validation module triggers the classification re-trainer module. After the system validation module detects which output is incorrect, it transmits a command to the related part to improve the performance.

To check the first assumption, the system validation module compares the controller output parameters with their previous values for each command. For instance, if the classifier output command was landing, the system validation module has to receive a reduction in altitude parameter of the flight controller output otherwise it is a mismatch. If there is a mismatch, the system validation module transmits a penalty command to the controller. Otherwise, the system validation module sends the penalty command to the classification module to activate the re-trainer and update the CNN model.

VI. Experimental Results

In this section, we demonstrate results for hand detection and gesture recognition using a AUSDOM AW335 Full HD 1080p webcam. Also, the validation system performance is examined in different scenarios. The webcam takes two-dimensional RGB images where all pixels have the same depth. A computer with 16 GB of RAM and an NVIDIA Geforce 1060 Ti for GPU are used to train and validate the modules.

(a) Palm **(b)** Fist **(c)** Victory sign **(d)** Ok sign

Fig. 6: Specified gestures for the webcam.

A. Hand Detection Using the SSD Model

We obtained 4600 images from 46 different subjects (100 images per subject with various distances from the camera) that were separated into a training and validation sets. The training and validation sets consist of 85% and 15% of the total dataset, respectively. The loss function for the SSD model [11] is a sum of classification and localization losses where the localization loss is calculated only for boxes with the same class label as the ground truth. During training, the total loss between the training set and the validation set is expected to converge to the minimum. Once the total loss converges to the minimum, an inference graph was generated as shown in Fig. 5. This graph displays the SSD total loss vs. the training steps. The tracking process was tested on a set of 635 new JPEG color pictures. The average accuracy is 94.28% which was obtained in 1.43 seconds.

B. Hand Gesture Recognition Using the Webcam

A dataset of images is gathered by capturing 125 color images with a size of 640×480 pixels from each subject for each distance from the camera. A total of 5000 images were collected from 40 subjects standing at various distances from the camera, i.e., at 1m, 1.5m, 2m, 2.5m, and 3m. Subjects stood in front of a white background with their right hand showing four pre-determined static gestures as shown in Fig. 6. After manually filtering the blurry and noisy images, a dataset of 4600 JPEG color pictures was obtained.

To train the model, we divided the database into two training and testing datasets. We allocated 70% and 30% of all refined images to the training and testing sets, respectively. Data augmentation was used on the training set to prevent an overfitting.

A sign of an overfitted model is when the training set has a higher accuracy than the validation set. From Fig. 7, it can be observed that overfitting was avoided at each epoch. The model was trained for 15 minutes for 15 epochs with the accuracy of 97.33%. After training, the system accuracy on 120 images from each class (480 pictures that system has never seen before) was 98.12%. Training and test graphs during this process are shown in Fig. 7.

Additionally, we tested three different scenarios to study the system behaviour . In all scenarios, the user shows $Gesture1$ (a hover command), but each time the user observes different response.

C. Case One: Validation System with No Penalty

In the scenario one, the drone is on the ground and the user shows $Gesture$ 1Hand detection module obtains the hand

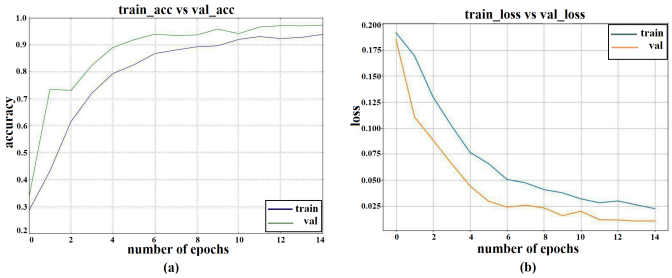


Fig. 7: Training set versus validation set metrics using a webcam. (a) Accuracy (b) Loss.

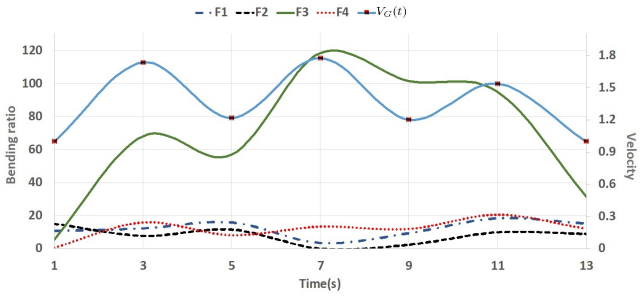


Fig. 8: Activating the third flex sensor

RoI, and then sends the hand RoI to the gesture classification module. Hence, the classifier detects class #1. Then the classifier output is sent to the controller and the controller generates a 'Hover' command. What the user observes is the drone is hovering. Therefore, the user confirms the output (drone behaviour). In this scenario the system operates correctly.

D. Case Two: Penalty on the Controller

In the scenario two, the drone is on the ground and the user shows $Gesture$ 1 . As in the previous scenario, classifier produces class #1 as its output. The classifier output is sent to the controller, and this time an error causes the controller to generate the 'Turn Left' command instead of the 'Hover'. Hence, the user disapproves the drone behaviour and sends a command to the system validation module through his/her glove. Once the system validation module receives class #1 from the classifier, expecting an increase in the flight controller output values, the system validation module then understands the mismatch and finds out that the problem is with the controller. The system validation module sends a penalty command to the controller causing future improvement in its performance.

E. Case Three: Penalty on the Classifier

In the third scenario, the user shows $Gesture$ 1. Hand detection picks the hand RoI and transfers the RoI to the classifier. This time the classifier provides class #3 instead of class #1, and then sends the classifier output to the controller. The controller presents 'Turn Left' command and forwards the command to the drone. What the user observes is that the drone is turning left, but he/she expects the drone to hover.

TABLE II: Penalty on the classifier: Classification validation decides to retrain the model on the Class #2 in three steps to raise the confidence from 16.67% to 100%. Also, meanwhile Class #3 has changed during this retraining.

Penalty on the Classifier: Class #2												
Steps	Step #1				Step #2				Step #3			
Class	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
C1	5	0	0	1	5	1	0	0	5	1	0	0
C2	0	1	5	0	0	11	0	0	0	11	0	0
C3	0	0	7	0	0	0	7	0	0	0	7	0
C4	0	0	1	7	0	2	1	5	0	0	0	11
Conf %	83.33	16.67	100	87.50	83.33	100	100	62.50	83.33	100	100	100

TABLE III: Effect of the activated flex sensor on the drone velocity.

t	f_1	f_2	f_3	f_4	***SVM output***	$rs_j(t)$	$V_G(t)$
1	10.67	15.21	5.78	0.37	P#0 : 0000	0	$V_0 = 1$
3	12.19	7.82	68.41	15.72	P#2 : 0010	31.32	1.74
5	15.82	11.65	56.98	7.98	P#2 : 0010	-37.03	1.22
7	3.49	0.03	118.71	13.19	P#2 : 0010	36.58	1.78
9	9.34	2.42	101.82	11.73	P#2 : 0010	-39.31	1.20
11	18.31	9.98	95.36	20.41	P#2 : 0010	5.22	1.54
13	15.19	9.06	31.93	11.88	P#0 : 0000	0	$V_0 = 1$

Therefore, the user disapproves the drone behaviour using the glove. In the first step, system validation checks the states of the classifier output and the controller output. The system validation module finds out that the controller operation is acceptable, so the problem is with the classifier. Next, the system validation module sends the penalty command to the classification module. The re-trainer then trains the classifier and obtains new weights. As indicated in Table II, system validation sends the penalty to the classifier, and classifier re-trains on three steps.

VII. Conclusion

We presented a gesture-based control system for multi-rotor aerial vehicles. The proposed system contains three separate validation modules: (1) validation of the gesture classifier that can retrain the model when required based on new data from a new user; and (2) flight control validation that causes adjustments to the flight controller. We also include human-in-the-loop as the third validation. Moreover, the system operates with two different human-robot interaction systems: a glove that produces control commands and an image processing method that produces discrete commands using hand tracking and gesture recognition.

The smart glove produces commands with more specific details in comparison to the image processing approach such as changing the angle in 'turn right' and 'turn left' commands by changing the bending ratio of the fingers or increasing and decreasing the altitude of the drone by changing the angle of the glove. The image processing approach produces commands without any specific detail such as 'hover' or 'land' commands.

To keep the validation modules separated, the system validation module and the classifier validation module were placed on the drone, and the human validation module was located on the glove. Three possible scenarios were studied on the proposed system to check the system behaviour.

As future work, we will extend the proposed system to recognize and to classify the main user's right-hand gesture in a crowded environment.

References

[1] A. Asokan, A. J. Pothen, and R. K. Vijayaraj, "Armatrona wearable gesture recognition glove: For control of robotic devices in disaster management and human rehabilitation," in *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*. IEEE, 2016, pp. 1–5.

[2] D. Bannach, O. Amft, K. S. Kunze, E. A. Heinz, G. Troster, and P. Lukowicz, "Waving real hand gestures recorded by wearable motion sensors to a virtual car and driver in a mixed-reality parking game," in *2007 IEEE Symposium on Computational Intelligence and Games*. IEEE, 2007, pp. 32–39.

[3] S. Bodiroža, H. I. Stern, and Y. Edan, "Dynamic gesture vocabulary design for intuitive human-robot dialog," in *Proceedings of the seventh annual ACM/IEEE International Conference on Human-Robot Interaction*, Mar. 2012, pp. 111–112.

[4] S. Bodiroža, G. Doisy, and V. V. Hafner, "Position-invariant, real-time gesture recognition based on dynamic time warping," in *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Tnteraction*, Mar. 2013, pp. 87–88.

[5] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1297–1304.

[6] G. A. Ten Holt, M. J. Reinders, and E. Hendriks, "Multi-dimensional dynamic time warping for gesture recognition," in *Proceedings of the Thirteenth Annual Conference of the Advanced School for Computing and Imaging*, 2007, pp. 23–32.

[7] A. Corradini, "Dynamic time warping for off-line recognition of a small gesture vocabulary," in *Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, Jul. 2001, pp. 82–89.

[8] G. Doisy, A. Jevtić, and S. Bodiroža, "Spatially unconstrained, gesture-based human-robot interaction," in *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction*, Mar. 2013, pp. 117–118.

[9] K. Natarajan, T.-H. D. Nguyen, and M. Mete, "Hand gesture controlled drones: An open source library," in *Proceedings of the 1st International IEEE Conference on Data Intelligence and Security (ICDIS)*, Apr. 2018, pp. 168–175.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems 25*, 2012, pp. 1097–1105.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 21–37.

[12] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017, pp. 4507–4515.

[13] P. Zhu, J. Isaacs, B. Fu, and S. Ferrari, "Deep learning feature extraction for target recognition and classification in underwater sonar images," in *Proceedings of the IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec. 2017, pp. 2724–2731.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jun. 2014.

[15] K. A. Bhaskaran, A. G. Nair, K. D. Ram, K. Ananthanarayanan, and H. N. Vardhan, "Smart gloves for hand gesture recognition: Sign language to speech conversion system," in *Robotics and Automation for Humanitarian Applications (RAHA), 2016 International Conference on*. IEEE, 2016, pp. 1–6.

[16] J. Bae, A. Larson, R. M. Voyles, R. Godzdanker, and J. Pearce, "Development and user testing of the gestural joystick for gloves-on hazardous environments," in *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*. IEEE, 2007, pp. 1096–1101.