UAVs Control Using 3D Hand Keypoint Gestures

Van Khoi Nguyen and R. Alba-Flores
Department of Electrical and Computer Engineering
Georgia Southern University
Statesboro, GA
vn01118@georgiasouthern.edu, ralba@georgiasouthern.edu

SoutheastCon 2022 | 978-1-6654-0652-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/SoutheastCon48659.2022.9764030

***Abstract*— Over past few years, unmanned aircraft vehicles (UAVs) have been becoming more and more popular for various purposes such as surveillance, automated industry, robotics, vehicle guidance, traffic monitoring and control system. It is very important to have multiple methods of UAVs controlling to fit in UAVs usages. The goal of this work was to develop a new technique to control an UAV by using different hand gestures. To achieve this, a hand keypoint detection algorithm was used to detect 21 keypoints in the hand. Then this keypoints were used as the input to an intelligent system based on Convolutional Neural Networks (CNN) that was able to classify the hand gestures. To capture the hand gestures, the video camera of the UAV was used. A database containing 2400 hand images was created and used to train the CNN. The database contained 8 different hand gestures that were selected to send specific motion commands to the UAV. The accuracy of the CNN to classify the hand gestures was 93%. To test the capabilities of our intelligent control system, a small UAV, the DJI Ryze Tello drone, was used. The experimental results demonstrated that the DJI Tello drone was able to be successfully controlled by hand gestures in real time.**

Keywords*— *Hand Key-points, UAV, convolutional neural networks, classification

There are multiple ways to control an unmanned aircraft vehicles (UAV), being the most popular the use of a manual remote control (RC) through a smartphone, a computer, or a joystick. However, with the emergence of a variety of UAV's applications, and its easy accessibility for personal and businesses utilization, there is a need to find different methods for UAV's control. Recent advances in natural user interfaces have been intended to use human innate features, such as speech, gestures, and vision to interact with technology in the way that humans do with each other. [1]. In this works, we propose to utilize hand gestures as the main mechanism to send control commands to an UAV. By using more natural inputs, such as hand gestures, as an optional communication technique to control UAVs will allow non-expert users, who have little knowledge about the system, to interact and operate UAVs in more natural ways..

Natural user interfaces have been researched since the 1980s [2]. Two natural user interfaces that can be used to implement reliable human-UAV interaction include voice commands and body gestures. Since body gestures represent a direct expression of mental concepts, and it does not change drastically from one person to another, as voice does, a large amount of research in human-robot interaction have been focused on the use of body gestures [3]. In this work the main objective is to send control commands to an UAV through hand gestures. For example, by opening the hand the user can indicate the UAV to fly forward, a thumb signaling to the left can indicate the UAV to fly to the left, etc.

There are different ways to recognize hand gestures, for example, using biosensor signals that can be used to detect hand motions or positions. Typical biosensors used to detect hand motions include Electromyography (EMG) and Electroencephalography (EEG) sensors [4, 5]. Another way is to use video frames and apply image processing techniques to determine the hand gestures [6, 7]. In this work video frames obtained from the UAV's camera were used to acquire the hand gestures, then machine learning algorithms were used to detect and classify the hand gestures. To achieve this, different methods based on visual gesture recognition have been studied, and in particular, methods based on the detection of hand keypoints were selected for this work [7, 8]. A general block diagram that illustrates the steps followed in this work is shown in fig. 1. For this work the UAV that has been chosen is a DJI Ryze Tello drone, because its small size that is very suitable for indoor applications.

This paper is organized as follows: Section II provides details of the methods used in this work, Section III provides the results obtained in this work, and in Section IV the conclusions and future work are provided.

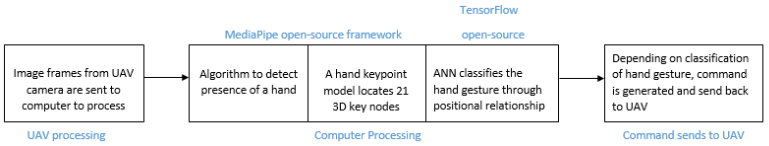


Figure 1. Block diagram of the developed system

I. METHODS

The main steps that were performed to achieve the goal are illustrated in fig. 1, and are as follows. i) First, frames from the UAV camera are captured, and the frames are sent to a computer wirelessly for the next process; ii) the computer processes the frames, and use an algorithm to detect 21 hand keypoint coordinates; iii) then, the meaning of the gesture in the input frame is determined through the positional relationships between the key nodes; iv) the hand gesture is classified using artificial neural networks, the result of the hand gesture classification is then used to generate control signals for the UAV. In order to command an UAVs to move to a specific direction, or to carry out a specific task, control commands are sent from a computer to the UAV. To be able to control the UAV with hand gestures, those hand gestures must be detected

This work was supported by the National Science Foundation, Research Experiences for Undergraduates (NSF-REU) , Award # 1950207.

978-1-6654-0652-9/22/$31.00 ©2022 IEEE

first, then classified, and finally related to specific UAV control signals that includes controlling the pitch, yaw, roll, and throttle of the UAV.

In this project we used hand pose estimation algorithms to find key points features of hands from images. These key points were used as landmarks to classify hand gestures. Hand keypoint detection is the process of finding the joints on the finger as well as the finger-tips in a given image or frame in a real-time video [9]. In recent years researchers have developed different frameworks that estimate hand keypoints from single camera or video frames that allow users to track a hand. Some of these frameworks include Handmap [10], hand pose, and Hand 3d [6].

To detect the hand pose we based our project on the MediaPipe framework to make the process of Hand Keypoint detection faster and more precise. MediaPipe [11] is a graph-based framework for building multimodal (video, audio, and sensors) applied machine learning pipelines. The platform facilitates the use of machine learning techniques to a wide range of applications. The Hand keypoint detection approach used in this work follows the one developed in [7]. This approach consists in using convolutional neural networks to obtain rough estimates of hand keypoints. This detection method is used for single images and it runs in real time on RGB images, and it has an accuracy comparable to methods that uses depth sensors. There are five main parts to successfully control an UAV using hand gestures: data acquisition, data preprocessing, model training, classification, and generation of command signals for the UAV.

A. Data Acquisition

Data from 8 different hand gestures were collected, and preprocessed. Fig. 2 lists the 8 hand gestures used in this work. The database consisted of a total of 2400 raw data retrieving from five volunteers, 300 samples of each hand gestures. There are 21 points detected on each hand gesture. The MediaPipe framework uses a single-shot palm detection model and performs precise key point localization of 21 3D palm coordinates in the detected hand region [9]. The MediaPipe pipeline utilizes multiple models like, a palm detection model that returns an oriented hand bounding box from the full image. The cropped image region is fed to a hand landmark model defined by the palm detector and returns high-fidelity 3D hand key points.

The MediaPipe framework provided the coordinates of 21 keypoints on a hand. These keypoints were used as the input of a Convolutional Neural Network (CNN) that was used to train the system to classify the hand gestures. The detail of the 21 keypoints on the hand is shown in fig. 3.

B. Data Preprocessing

Using the created hand gestures database (2400 hand gestures), the MediaPipe framework was used to detected the 21 hand keypoints in each image. Each of the 21 hand landmarks is composed of the coordinates x, y and z, with original values of these coordinates in the range 0 to 255. The x and y values represent the 2D coordinates of each hand keypoint on the image, and the z value represents the landmark depth with the depth at the wrist being the origin, and the smaller the value the closer the landmark is to the wrist. Before training the dataset, the landmark coordinates need to be normalized in order to make the training process faster and more accurate. The coordinates x and y were normalized to [-1 to 1] by the image width and height respectively. To normalize the x and y keypoints, the length and width of the input image were used. The normalization equations are given in equation. (1) and (2).

$$xnorm = \frac{x}{width} \quad (1)$$

$$ynorm = \frac{y}{height} \quad (2)$$

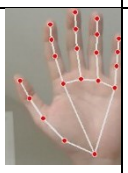
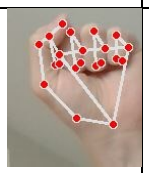
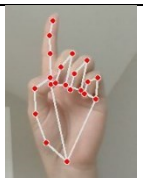
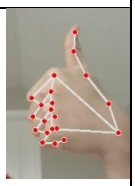
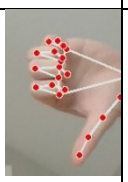
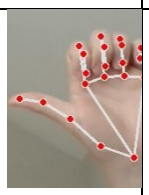
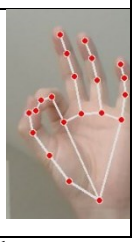
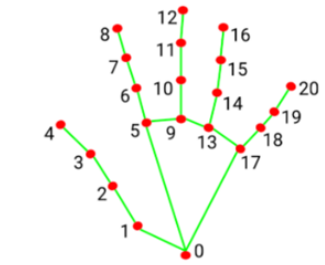
Class	0	1	2	3
Hand Motion	Hand Open	Hand Closed	Pointer	Thumb Up
UAV Motion	Fly Forward	Fly Backward	No Motion	Fly up
Example				
Class	4	5	6	7
Hand Motion	Thumb Down	Thumb Left	Thumb Right	OK
UAV Motion	Fly Down	Fly Left	Fly Right	Land
Example				

Figure 2. The 8 hand gestures used in this work.

The space coordinate system was established with the 0 node at the wrist as show in fig. 3. The smaller the value of z is, the closer the node is to the camera. The normalization method is also used to determine the depth coordinates using eq. 3.

$$znorm = \frac{z - zo}{zcam - zo} \quad (3)$$



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 3. Hand Keypoint nodes [11].

Figure 4 illustrate a small portion of the normalized database. In this figure each row provides the keypoints of one image. The first column indicate to which class this image belongs (in this work we have 8 classes, each class corresponds to a particular hand gesture). Columns 2 and 3 provide the coordinates (x,y) of the writs (node 0), the next two columns (col. 4 & 5) provide the (x,y) coordinates of node 1, columns 6 and 7 are the coordinates of node 2, and so on.

Class	Node 0		Node 1		Node 2		Node 3		Node 4	
0	0	0	0.1893	0.102881	0.403292	0.127572	0.584362	0.135802	0.728395	0.164609
0	0	0	0.190283	0.101215	0.40081	0.137652	0.582996	0.149798	0.728745	0.182186
0	0	0	0.184739	0.104418	0.389558	0.140562	0.566265	0.164659	0.702811	0.204819
0	0	0	0.180723	0.108434	0.381526	0.156627	0.554217	0.176707	0.690763	0.212851
0	0	0	0.176	0.112	0.376	0.164	0.548	0.192	0.68	0.232

Figure 4: Segment of Normalized values of the Database

C. *Training and Classifications*

After the keypoints in each image in the database are normalized, the next step is to enter the normalized data to a convolutional neural network (CNN) to train it and classify the hand gestures. To accomplish this, Tensorflow library was used. The main steps used for the training and classification are as follow [12]:

- Load the preprocessed data
- Split in random way the data, into training and testing sets (60% for training)
- Train the model using the Rectified Linear Unit (ReLU) activation function [ref Olimov]
- Optimize the classifier and convert the output to a probability vector. The activation function used for this step was Softmax.
- Accuracy and score of model.

Fig. 5 illustrate the process of training a CNN [12]. In this work 60% of the database was used for training, and 40% was used for testing. For the training stage a ReLU activation function was used, and for the optimization stage the Softmax function was utilized. The final result of this stage was to obtain a trained model which is used as a guidance to run real-time frame captured from the UAV camera through the algorithm to correlate with the training data set.

D. *Command Generation Signal for the UAV.*

After archiving the trained model from previous step, a Python based programming is written is

In the last stage, the result of the classification system was used to send control signals to the DJI Ryze Tello Drone via Wi-Fi. The control signals were generated using a Python based code and sent to the UAV via Wi-Fi. The speed of the drone was set to 20 cm/s to be able to have a good control of the drone in an indoor environment.

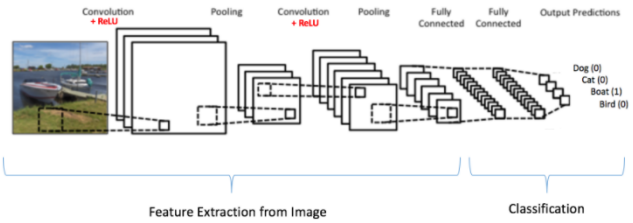


Figure 5: Predicting input after training model [12].

II. RESULTS

After completing the training of the data and building a model for the hand gesture classification, the next step was to evaluate the model's performance. The evaluation metrics that were used in this work were Precision, Recall, and the F1-score.

Precision Scores. The Precision score helps when the costs of false positives are high. When false positives are too high, those who monitor the results will learn to ignore them after being experienced many incorrect results. A model with low precise values means that the model finds a lot of the positives, but its selection method is noisy, this is, that it also wrongly detects many positives that are not actually positives. On the other side, a precise model means that maybe it does not find all the positives, but the ones that the model finds as positive are very likely to be correct. The formula to compute the precision score is given in eq. 4.

Recall Scores. Recall scores help when the cost of false negatives is high. A model with high recall score succeeds well in finding all the positive cases in the data, even though they may also wrongly identify some negative cases as positive cases. A model with low recall score is not able to find all (or a large part) of the positive cases in the data. The formula to compute the recall score is given in eq. 5.

F1 scores. Precision and Recall are the two building blocks of the F1 score. The goal of the F1 score is to combine the precision and recall metrics into a single metric. In particular, the F1 score works very well on imbalanced data. The formula to compute the F1 score is given in eq. 6.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (4)$$

$$Recall = \frac{turepositives}{trupositives + falsenegatives} \quad (5)$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (6)$$

In this work a CNN was implemented to classify 8 different hand gestures: hand open, hand close, pointer, thumb up, thumb down, thumb left, thumb right, and the OK sign as shown in fig. 2. Each hand gesture has been associated with a specific control command for the UAV. The metrics used to evaluate the accuracy of the classification model were Precision, Recall, and the F1-score. The results of these metrics for each of the classes are given in Table I. The data set consisted of 300 images of each hand gesture. Analysis of the F1-score shows that the accuracy of the classification system was 90% or higher.

Experimental results demonstrated that the developed intelligent system was able to control the UAV in real time, based on the hand gestures inputs from the user. Figure 6 shows a real-time picture from the UAV's camera and how the hand keypoints are detected. To have a good control of the UAV in indoor environments, for this experimental the speed of the UAV was set to 20cm/sec.

Table I. Summary of Model Accuracy after Training

Classification Report				
Class	Precision	Recall	F1-score	Support
0	0.95	0.95	0.95	300
1	0.93	0.91	0.92	300
2	0.93	0.99	0.96	300
3	0.98	0.84	0.90	300
4	0.87	1.00	0.93	300
5	0.93	0.98	0.95	300
6	0.91	0.88	0.90	300
7	0.90	0.83	0.90	300

Figure 6: Real-time picture from UAV camera

III. CONCLUSION

We have presented an application of computer vision used to control an UAV. The accuracy of the hand gesture classification was above 90%, so that the user was able to easily control an UAV with hand gestures. However, the training process was applied only on the right hand, so the left hand does not work on this project. This issue can easily solve by re-training the model with both hands, and then the UAV can easily be controlled using any of the hands. Furthermore, the UAV camera can be easily replaced by any fixed camera so that users can have more flexibility while controlling the UAV. The only limitation of the CNN system is that the classification is only recognized by the trained models. If a different hand gesture is captured, the CNN system will map it to the closest hand gesture such as two fingers pointer is classified as one finger pointer if only one finger pointer gesture is trained. This issue can be improved by having more training data so that the final accuracy is closer to 1.

This work does not stop from here, there are still many advances that can be made to this work so that the UAV has more accuracy and functions.

REFERENCES

[1] R. A. Suarez Fernandez, J. L. Sanchez-Lopez, C. Sampedro, H. Bavle, M. Molina, and P. Campoy, "Natural user interfaces for human-drone multi-modal interaction, 2016 International Conference on Unmanned Aircraft Systems (ICUAS), June 2016, doi:10.1109/ICUAS.2016.7502665

[2] Richard A. Bolt. Put-that-There: Voice and gesture at the graphics interface. SIGGRAPH Comput. Graph., 14(3):262–270, July 1980.

[3] .I. Pavlovic, R. Sharma, and T.S. Huang. Visual interpretation of hand gestures for human-computer interaction: a review. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 19(7):677–695, Jul 1997.

[4] A. Singandhupe, H. Manh, and D. Feil-Seifer, "Reliable Security Algorithm for Drones Using Individual Characteristics from and EEG signal", IEEE Open Access Journal, April 2018, doi: 10.1109/ACCESS.2018.2827362

[5] J. Vigliotta, J. Cipleu, A. Mikell, and R. Alba-Flores, “EMG Controlled Electric Wheelchair”, Intelligent Systems Conference, Amsterdam, The Netherlands, September 2-3, 2021

[6] M. Oberweger, G. Riegler, P. Wohlhart, and V. Lepetit, “Efficiently creating 3d training data for fine hand pose estimation,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4957–4965, 2016.

[7] Tomas Simon, H. Joo, I. Matthews, and Y. Sheikh, “Hand Keypoint Detection in Single Images using Multiview Bootstrapping”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1145-1153.

[8] X. Chen, L. Yu, X. Meng, and Y. Hua , “Visual gesture recognition base on hand key points”, Proceedings of the 2nd Conference on Computer Vision and Data Mining (ICCVDM 2021), doi: 10.1088/1742-6596/2021/1/012037

[9] Hand Keypoint Detection using Deep Learning and OpenCV, https://learnopencv.com/hand-keypoint-detection-using-deep-learning-and-opencv/

[10] Xiaokun Wu, Daniel Finnegan, Eamonn O’Neill and Yong-Liang Yang, “HandMap: Robust Hand Pose Estimation via Intermediate Dense Guidance Map Supervision”, European Conference on Computer Vision (ECCV), Sep. 2018, Munich, Germany

[11] MediaPipe Hands,Google Github, https://google.github.io/mediapipe/solutions/hands.html

[12] KDnuggets, “An Intuitive Explanation of Convolutional Neural Networks”, https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/3