Date of publication DEC-31, 2024, date of current version NOV-05, 2024.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.23.4.3757

Gesture Recognition based on Deep Learning for Quadcopters Flight Control

VOLODYMYR SAMOTYY[1,3], NIKITA KISELOV[2], ULIANA DZELENDZYAK[3], OKSANA SHPAK[3]

[1]Department of Automatic Control and Information Technology, Faculty of Electrical and Computer Engineering, Cracow University of Technology, 31155 Cracow, Poland
[2]Paris-Saclay Faculty of Sciences, Université Paris-Saclay, Orsay 91400, France
[3]Department of Computerised Automatic Systems, Institute of Computer Technologies, Automation, and Metrology, Lviv Polytechnic National University, Lviv 79013, Ukraine

Corresponding author: Volodymyr Samotyy (e-mail: vsamotyy@pk.edu.pl).

ABSTRACT This article presents a system for controlling quadcopters with gestures, which are recognized by a model based on neural networks. A method based on a combined deep learning model is proposed that provides real-time recognition with minimal consumption of computing power. An implementation is presented that offers the possibility of controlling the quadcopter in two ways, via gestures or the keyboard. A functionality is also provided for adding new gestures for recognition using interactive code via the Jupyter Lab web application. A special mode is implemented that allows us to create a data set for a new test directly from the quadcopter camera to simplify data collection. The operation of the control and recognition module is demonstrated using an example in which a DJI Tello Edu drone is controlled. The results of tests under real conditions are presented. The developed software allows one to speed up the process of gesture recognition and facilitates the process of controlling the quadcopters. Several areas of improvement of the developed system and their possible technical implementation are proposed.

KEYWORDS Type gesture control; deep neural networks; computer vision; convolution neural network; artificial intelligence; quadcopter; TensorFlow; Tensorboard.

I. INTRODUCTION

THE use of Gesture control has always been a popular research topic, but following the advent of neural networks (NNs) in the domain of computer vision systems, the implementation of these systems in various devices has begun to expand rapidly to include not only specialized devices for people with hearing impairment but also smartphones and other devices with built-in cameras. The control of UAVs (unmanned aerial vehicles) using hand gestures is one of such examples that has gained popularity in the consumer sector and has also attracted interest from the military sector. This article presents a system based on a self-created NN and models from the MediaPipe platform for controlling a quadcopter via gestures. The developed system combines such tasks as gesture recognition, gesture classification, processing, and execution of commands transmitted to the quadcopter. A prototype model for the visual gesture control of quadcopters is built using modern developments in the field of artificial intelligence and the MediaPipe NN platform from Google.

Gesture recognition is an area of computer and language technologies and involves interpreting human gestures using mathematical algorithms [1, 2]. It is sometimes considered a sub-discipline of computer vision. Currently, users can use simple gestures to control and interact with devices without physically touching them, and most existing approaches are implemented using cameras and computer vision algorithms. Gesture control is essentially a natural interaction that does not rely on mechanical devices. An intelligent approach to human control of UAVs in real time was described in [3]. This approach uses a multi-mode command structure. Gesture recognition is implemented on the basis of machine learning. The main application areas are currently the automotive sector, consumer electronics, gaming, the military [3, 4], home automation (Internet of Things), and automated sign language translation.

The ability to track human hand movements and identify which gestures they are performing can be achieved using various tools. Here, we consider computer vision-based models, which are based on the visual perception of an image by a computer and its subsequent interpretation. The most popular method of classical computer vision for gesture and motion recognition is segmentation. A method of gesture recognition based on NNs exploits the architecture of a convolutional NN and different sets of systems for its use [5, 6,

7]. A hand-machine interface device [8] providing information in real-time was described in detail. One of the works [9] proposed a method of hand tracking and gesture recognition for visual interfaces. Gesture recognition is performed by pre-evaluating several defined gestures. A modern approach was described in [10]. The network gaming interface is designed around a digital assistant that uses language, gestures, and touch. The basis of a new type of interface for mobile computing systems was proposed in [11]. The concept of kinetic interfaces, in which movement was considered as the primary mode of input was highlighted. The system for recognition of gestures providing a way of non-verbal communication was described in [12]. The algorithm does not depend on user characteristics and, therefore does not require training of sample data. Instead, [13] proposed the method that provides effective drone control without lengthy training. Using the LeapMotion sensor, drone control via gestures was implemented in [14]. In work [15], a Natural User Interface (NUI) was developed for controlling drones using speech and hand gestures. Two fundamental solutions for positional tracking, marker, and non-marker methods were described in [16]. System indoor tracking with the base station and rotating laser and photodiode sensors on object tracking was presented in [17]. In [18], a method for estimating position was proposed using low-cost GPS and optical streams from UAV camera. Successful use and application of the supplement of reality was described in [19, 20], which includes combining several sensors, object tracking, and registration of real and virtual worlds. [21] developed methods for detecting and tracking 3D objects for various computer vision applications, including various fields such as robotics, driving, space, and the military. [22] described the training of an artificial neural network for the pose of an object using only synthetic single-channel edge-enhanced images. The influence of localization accuracy [23] on the visual effect of overlaying augmented reality, optimizes the implementation of demonstration response in a virtual geographic environment. Providing navigation on a quadcopter by a person and using a set of glasses was considered in [24]. The proposed approach can be used in many situations, and also for people with disabilities. In work [25], UAVs support of users in their actions was described. In this context, the exchange of spatial information between the user and the UAV is facilitated by the three-dimensional localization of the UAV assistant. A data synthesis pipeline was developed to create a realistic multimodal dataset that includes both an exocentric view of the user and an egocentric submission of UAVs.

The latest developments in this area include the so-called combination models, which are formed from combinations of different types of deep NNs. For example, a combination of neural model based on several convolutional deep NNs and a simple multi-layer neural model can be used for gesture recognition systems [26-29]. Convolutional NNs are responsible for finding the hand image in the input data and identifying key points, while a conventional NN classifies these as points in space that are specific to a particular gesture. This model can be easily adapted and retrained to recognize new gestures, and can be optimized to run on mobile devices, and is therefore used in the proposed system.

II. SYSTEM ARCHITECTURE

The development of the proposed system architecture is based on the following main principles: fault tolerance, flexibility of component replacement (i.e., the UAV can be replaced with another model, or new commands can be added to the NN), and image processing that does not take place on the UAV, which only executes commands (Fig. 1). Gestures are used to control an UAV, or rather a quadcopter, which is a type of multicopter. We used a DJI Tello Edu quadcopter, a special STEM version of a quadcopter from the Chinese company Ryze Robotics in cooperation with DJI, another Chinese company that is the market leader in drones. Tello Edu's image processing is supported by the Intel Movidius Myriad processor, which allows for the execution of instructions written in Python and also supports low-level image processing and recognition tools.

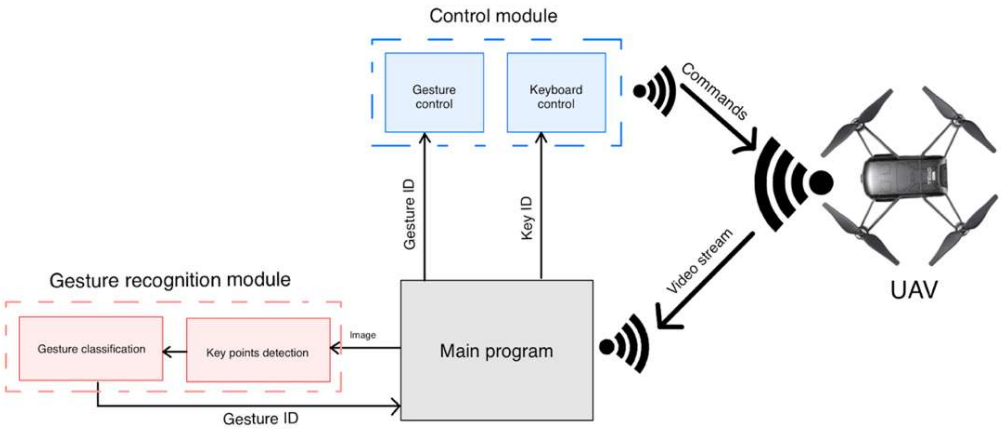


Figure 1. Schematic representation of the system architecture.

The most important component of the gesture management system is, of course, the gesture recognition module. It consists of two parts: a comprehensive key point recognition model called MediaPipe Hands, and a NN classifier that recognizes gestures based on the key points found. The output is the ordinal index of the recognized gesture. This index is transmitted to the quadcopter control module, where, depending on the parameter settings, the control command is transmitted to the quadcopter.

The quadcopter can also be controlled via a computer keyboard. This part of the control module is independent of the recognition module, and interacts only with the main program.

The main program initializes all of the modules, establishes communication with the quadcopter, and is responsible for data transfer between modules. For example, this module transmits the image from the quadcopter's video stream to the gesture recognition module, and sends the index of the recognized gesture to the quadcopter control module. The proposed architecture provides several options for controlling the quadcopter, which increases flight safety, while the modularity of the architecture allows the user to change and customize quadcopter control parameters (such as speed or command type) and add new gestures without changing the main execution program. This is enough to make the necessary changes to the solution modules.

III. IMPLEMENTATION

The following technologies are used to implement the quadcopter gesture control system:

- the Python SDK *djitellopy* library;
- a module for recognizing hand key points;
- the backend of the project, which connects and controls the quadcopter;
- code for training an NN written using the TensorFlow framework;
- code for optimization of the NN hyperparameters using the Tensorboard platform.

In addition, to enable graphical visualization of gesture recognition, code is written to display the image from the quadcopter's camera in a program window using the OpenCV Python library. This library is used to visualize the results from the NNs in real time, to process the image before feeding it to the NN input, and to display additional information on the video stream from the quadcopter. The battery status of the drone is displayed in the lower left corner, and the frame rate per second in the upper left. The key points of the hand are drawn on top of the image as a "white skeleton" (Fig. 2).

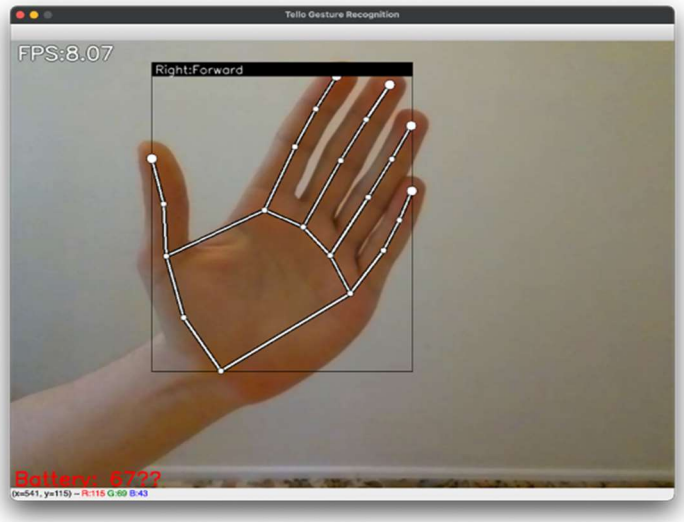


Figure 2. Graphical visualization of interconnected key points used for gesture recognition. The name of the classified gesture is displayed in the corner of the black frame that surrounds the hand in the image.

The code can be divided into three parts: the hand recognition module, the gesture classifier module and the quadcopter control module. However, the components of the system have a slightly more complex organizational structure, and detailed information can be found in the Codebase that is available on the GitHub repository (https://github.com/kinivi/tello-gesture-control).

A. HAND RECOGNITION MODULE

As already mentioned, the code consists of three parts, and the gesture recognition module was developed first. A combined architecture is used for gesture recognition, in which the MediaPipe Hands model is used to recognize key points of the hand and a self-developed NN is applied to classify gestures based on these points. Depending on the classified gesture, a certain command is transmitted to the quadcopter.

The solution is implemented using MediaPipe, a framework for building cross-platform machine learning solutions. The proposed model and architecture yield real-time inference speed on mobile GPUs, with high prediction quality. A single-pass deep learning-based detector model optimized for a real-time mobile application similar to BlazeFace, which is also available in MediaPipe, is used to detect the initial hand positions. Hand detection is an extremely challenging task, as the model must work on different hand sizes with a large zoom range (~20x) and needs to be able to detect even intertwined hands. The use of a palm detector solves these problems, and the hand landmark model is activated when the palm has been successfully detected by the detector in the image. After running palm detection over the entire image, a hand landmark model subsequently performs precise localization of 21 keypoint coordinates in 3D, within the detected hand regions, using regression. The model learns a constant representation of the internal pose of the hand, and is robust even to partially visible hands and self-occlusion. The model has three outputs:

- 21 hand marks, (x, y) coordinates and relative depths;
- a hand flag indicating the probability of the presence of a hand in the input image;
- binary classification of the hand (left or right).

To enable recovery from a tracking failure, there is another model result, which is analogous to calculating the probability of an event, that detects whether a hand is actually present in

the frame. If the score is below a given threshold, the detector is triggered and resets the tracking. Since the MediaPipe Hands model is ready to use, the process of connecting it using a Python script is quite simple: it is sufficient to import the hands class from the MediaPipe library and transfer the image as a vector. The output is the 3D coordinates of the key points of the hand, with (x, y, z) coordinates for each of 21 points. Two coordinates for points (x, y) in the 2D plane are used for gesture recognition. In order to process the results with an NN classifier, it is necessary to transform the data into a vector and normalize the data, which allows the coordinates to be one range, thereby speeding up the training of the model and increasing accuracy.

The step-by-step process of converting and processing the results into a vector is as follows:

- Point coordinates are converted from absolute to relative. Instead of indicating the coordinates within the entire image, they will determine the position of the hand relative to the hand. The reference point is the base point with index 0 (coordinates (0,0,0)).
- The list of arrays of points is combined into a single consecutive array with a length of 42 elements.
- This concatenated array is normalized based on the maximum number, according to formula (1), where x – input vector to the layer, [] – operation of taking the absolute value and z – output vector

$$z = \frac{x - \min(x)}{[\max(x) - \min(x)]}. \quad (1)$$

The steps of the process are visualized in Fig.3 with a mock-up data. The output is a 1x42 array that is ready for use by a classifier module.

(Landmark coordinates)

ID：0	ID：1	ID：2	ID：3	……	ID：17	ID：18	ID：19	ID：20
[551, 465]	[485, 428]	[439, 362]	[408, 307]	……	[633, 315]	[668, 261]	[687, 225]	[702, 188]

(Convert to relative coordinates from ID:0)

ID：0	ID：1	ID：2	ID：3	……	ID：17	ID：18	ID：19	ID：20
[0, 0]	[-66, -37]	[-112, -103]	[-143, -158]	……	[82, -150]	[117, -204]	[136, -240]	[151, -277]

(Flatten to a one-dimensional array)

ID：0		ID：1		ID：2		ID：3		……	ID：17		ID：18		ID：19		ID：20	
0	0	-66	-37	-112	-103	-143	-158	……	82	-150	117	-204	136	-240	151	-277

(Normalized to the maximum value(absolute value))

ID：0		ID：1		ID：2		ID：3		……	ID：17		ID：18		ID：19		ID：20	
0	0	-0.24	-0.13	-0.4	-0.37	-0.52	-0.57	……	0.296	-0.54	0.422	-0.74	0.491	-0.87	0.545	-1

Figure 3. Step-by-step processing based on an example of real data.

B. GESTURE CLASSIFIER MODULE

The model of the gesture classifier is a multiperceptron with four fully connected layers, the input to which takes the form of a pre-processed vector of key points. A multilayer perceptron (MLP) is a type of artificial neural network (ANN), and consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. Three of the four hidden layers of our model have a rectified linear unit (ReLU) activation function, and the last one is Softmax. To implement this NN, the Tensorflow library was used together with the Keras application software tool. The schematic description of the model with the mathematical definition is given in Fig 4. Simplified structure visualization is given in Fig. 5.

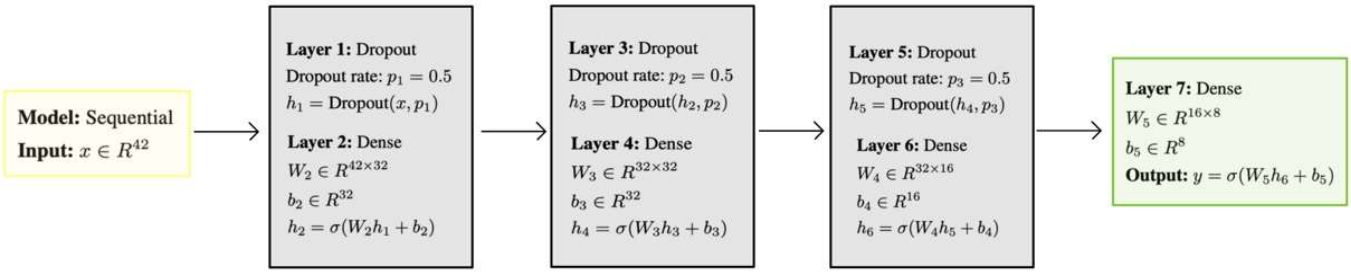


Figure 4. Schematic structure with mathematical definitions.

Here, σ denotes the ReLU activation function, which promotes faster training and mitigates the vanishing gradient problem due to its linear, nonlinear nature and computational simplicity. The ReLU activation function can be defined mathematically as in (2), where x denotes input signal from the previous layer.

$$\mathrm{Re\,LU}(x)=\begin{cases}x, & \text{if } x>0,\\ 0, & \text{otherwise.}\end{cases} \quad (2)$$

$$\mathrm{Dropout}(x,p)=\frac{1}{1-p}M(x)\odot x. \quad (3)$$

Dropout (3) is a regularization technique in which some elements of the input are randomly set to zero during training, with probability p_i. This is an effective technique that helps prevent overfitting and improves generalization in deep NNs by randomly dropping units during training, thereby encouraging more robust feature learning.

In this equation, x is the input vector, p is the dropout probability, $\odot$ denotes element-wise multiplication, and $M(x)$ is a binary mask vector generated by sampling each element independently from a Bernoulli distribution with probability p. A scaling factor of $1/1-p$ is applied to maintain the expected value of the input during training.

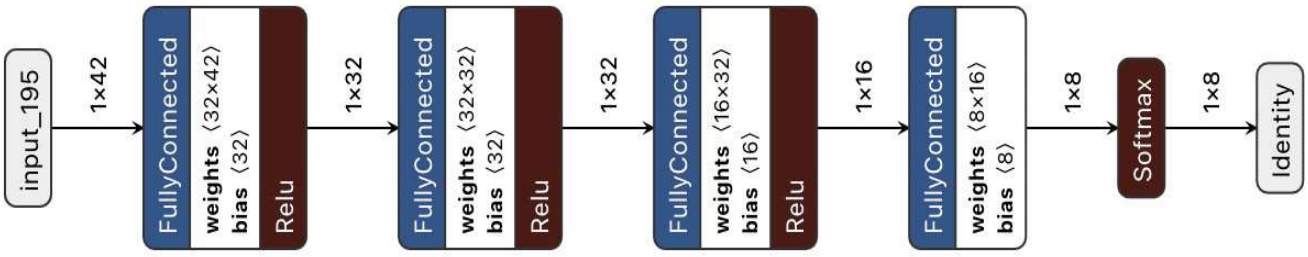


Figure 5. Simplified structure visualization of the classifier model

The Adam optimizer and a cross-entropy loss function are used in the developed model. More than 1,500 pairs of data for eight classes of gestures were collected to train the model, which was carried out using the Google Colab platform.

For greater accuracy, code was created using the Tensorboard data visualisation platform to select the best hyperparameters for model training. In Fig.6 the Tensorboard dashboard with the process of determining best possible set of parameters for our model is visualized. This helped in determining the optimal model parameters for the use in our case.

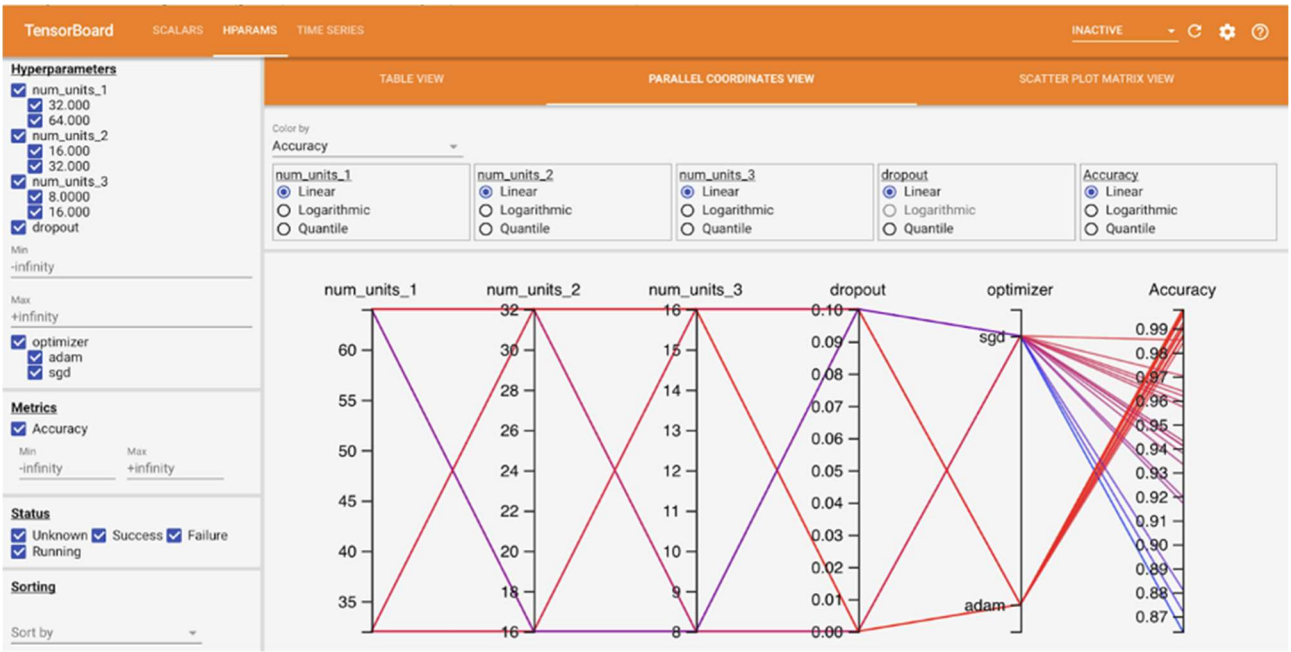
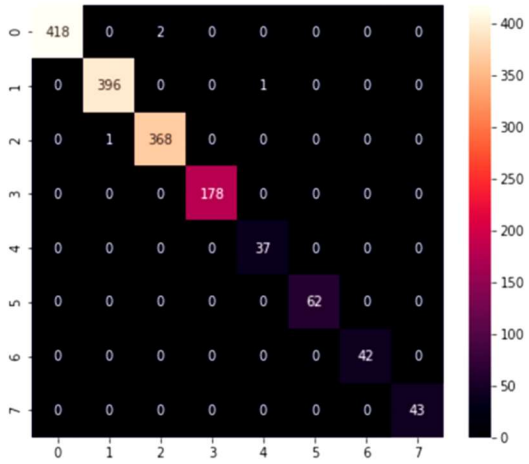


Figure 6. View of the Tensorboard interface and results for the optimal hyperparameters.

From Fig. 7, it can be seen that the accuracy of the model on the test dataset (30% of all data) was greater than 97% (precision must be > 97%) for each class. Due to the simple structure of the model, it is possible to obtain high accuracy with only a small number of examples of each class for training. There is no need to retrain the model for each gesture with different lighting, as MediaPipe takes care of all key point detection process.

C. QUADCOPTER CONTROL MODULE

When the gesture recognition stage is complete, the next step involves the implementation of the quadcopter control system based on the recognized gestures and receiving the image from the quadcopter's camera. We used the DJI Tello Edu quadcopter as a UAV with an open SDK for its programming and control, and the djitellopy library was used in view of its convenience and the speed of development.



Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	420
1	1.00	1.00	1.00	397
2	0.99	1.00	1.00	369
3	1.00	1.00	1.00	178
4	0.97	1.00	0.99	37
5	1.00	1.00	1.00	62
6	1.00	1.00	1.00	42
7	1.00	1.00	1.00	43
accuracy			1.00	1548
macro avg	1.00	1.00	1.00	1548
weighted avg	1.00	1.00	1.00	1548

Figure 7: Gestures and corresponding quadcopter control commands

This library is an ideal tool for our tasks, since the most difficult aspect of working with a quadcopter one is obtaining the image from the camera. A DJI Tello is controlled from a computer or phone via the WiFi protocol (IEEE 802) at a frequency of 2.4 GHz. The Tello SDK is connected to the quadcopter via a UDP WiFi port, allowing the user to control the quadcopter using text commands. Streaming applications often use UDP, as dropping packets is better than waiting for packets to be delayed due to retransmissions, which is not possible in a real-time system. For this reason, this protocol was chosen to control the quadcopter and transmit the image. Since communication with the quadcopter may not be stable over longer distances, and since the acquisition of real-time data is critical to the control of the quadcopter, the use of this protocol is reasonable.

The djitellopy library handles all the work with the protocol involving receiving and sending data. After the successful execution of the program in the output console, a status message is received to indicate that the connection is established and the data stream from the camera has been received. When the image has been processed by the gesture recognition module, the class of the gesture (or its absence) is transmitted in the form of an index. For each gesture and corresponding index, a certain command is executed by the drone. The commands and the corresponding gestures are shown in Fig. 8.

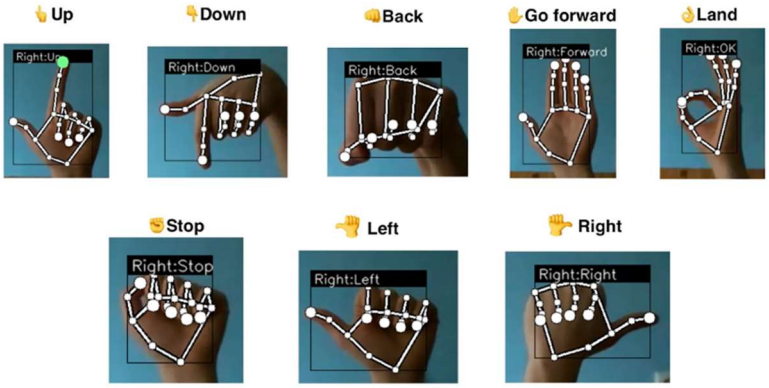


Figure 8. Gestures and corresponding quadcopter control commands.

Since there may be noise and other problems when transmitting the image, the commands are recorded in a special buffer. If the buffer is filled with commands of the same type, the speed of the quadcopter is set in the given direction; this allows for increased fault tolerance, as well as making the movements of the quadcopter smoother. Due to the fact that gesture recognition occurs in real time, the use of a buffer does not create problems with latency, as it fills up very quickly.

D. ADDING NEW GESTURES

The last stage of the implementation was the addition of a functionality to enable the recording of new gestures. Since the solution itself is modular, to add a new gesture, the user only needs to retrain the NN (the gesture classifier). To simplify data collection, a special mode is implemented that allows a dataset for a new gesture to be created directly from the quadcopter's camera.

In this mode, when a number key from "0" to "9" is pressed, the key points that have been recognized by the MediaPipe Hands model are recorded in a tabular data file from the indexes according to the pressed key. In Fig.9 an example of the created file with the data can be seen. The coordinates of

the points have already been recorded in the form of a pre-processed and normalized vector. In this mode, additional data can also be collected to improve the recognition of already insinuating gesture. An interactive code in Jupyter format was created to retrain the NN classifier on new data.

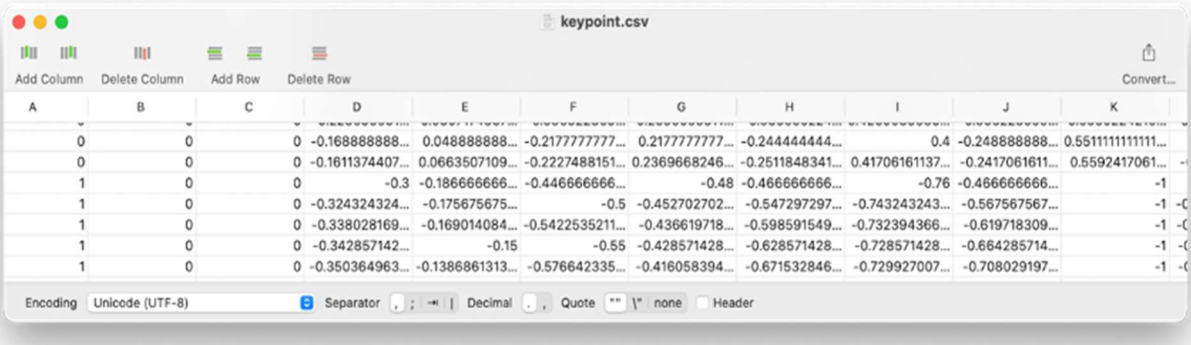


Figure 9. File with key points in the form of a normalized vector.

In this mode, when a number key from ”0” to ”9” is pressed, the key points that have been recognized by the MediaPipe Hands model are recorded in a tabular data file from the indexes according to the pressed key. On the Fig.9 can be seen an example of the created file with the data. The coordinates of the points have already been recorded in the form of a pre-processed and normalised vector. In this mode, additional data can also be collected to improve the recognition of already insinuating gesture. An interactive code in Jupyter format was created to retrain the NN classifier on new data. More information provided in the Codebase is available on the GitHub repository (https://github.com/kinivi/tello-gesture-control).

IV. RESULTS AND ANALYSIS

When the implementation of all components was complete, the system was tested and its full functionality in terms of quadcopter flight control was demonstrated. The recognition module performed gesture classification with ultra-high accuracy, both for the initially programmed gestures and after training on new ones. The key point recognition model based on the MediaPipe platform showed fast performance, even on a low-specification laptop with a dual-core Intel i5 processor with integrated graphics. It is worth noting here that until recently, such models could only be run on a multi-core PC with discrete graphics; that is, the performance of the recognition module fully justifies the chosen architecture. The classifier NN was quickly retrained to add new gestures. On average, it was sufficient to collect 30–70 examples to get accurate results in terms of recognizing a new gesture.

The control module operated correctly, and the quadcopter smoothly executed commands with stable flight. We also tested indoor emergencies (such as a collision with a wall or loss of visual contact). In this case, the quadcopter quickly recognized the “STOP” command or rapidly switched to the keyboard control mode, which allowed it to avoid emergency situations. Fig. 10 demonstrates the screenshot of the visualization interface during the quadcopter operation.

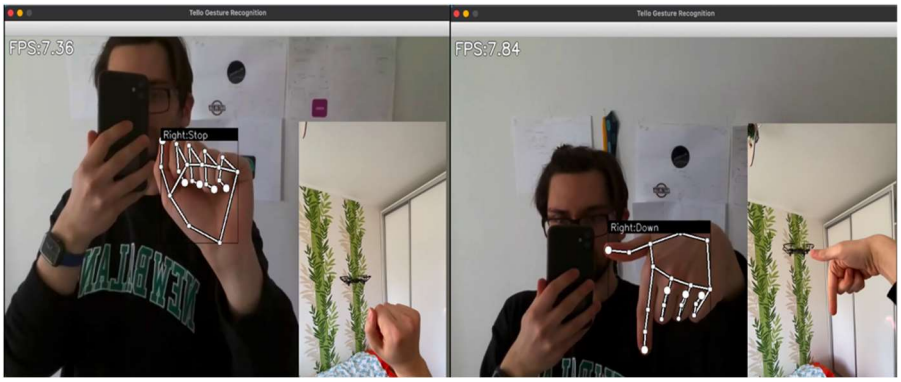


Figure 10. Demonstration of UAV gesture control. In the lower right corner, we can see video images from the smartphone camera used to record the quadcopter’s flight.

The main drawbacks that could be improved are that the recognition system did not work satisfactorily in very low lighting or at a large distance from the hand. The first problem can be solved by using infrared cameras instead of conventional ones (as infrared cameras can work based on the light emitted by the heat of a person’s palms). Since the silhouette of the hand does not change, the key point detection model can be retrained on the new data without changing the architecture.

The problem of long distances is less acute, but can be

solved by using a holistic model [30-34]. This model allows the system to recognize first the human body (key points of the skeleton), and then the key points of the hand from enlarged images of the area where the hands are located on the skeleton. This approach would theoretically allow for the recognition of gestures over long distances at which the image of the hand does not occupy most of the frame.

The addition of a functionality for the recognition of moving gestures may also be a direction for future development, for example, by writing a letter in the air with a finger or other moving hand gestures. Since movements are an integral part of human cognitive perception, the ability to recognize such gestures would add intuitiveness and convenience to the control system. For this purpose, it can be possible to use a NN classifier based on the long short-term memory architecture, where the input is the stored history of key points for the last *n* frames, and the output is the gesture class. Since the buffer stores only a vector of point coordinates, the memory consumption will be negligible. Another disadvantage is that the system recognizes gestures of only one hand. Since the hand tracking model can work when multiple hands are in the frame and can recognize each one, multi-gestures based on two hands can be used in the future. This will give more control over potentially difficult situations in flight. Codebase is available on the GitHub repository (https://github.com/kinivi/tello-gesture-control).

V. CONCLUSIONS

A gesture recognition system for quadcopter control was developed using the MediaPipe NN platform. Software for gesture recognition with high accuracy was developed using artificial intelligence, and the classified gesture commands were then used to control quadcopter. This software is based on platforms and tools from Google and DJI, and has the following technical characteristics: real-time gesture recognition and control of quadcopter, modularity of the solution for easy modification of recognition and control modules, the ability to add new gestures for recognition, the ability to control the quadcopter in two ways (via gestures or the keyboard), and a graphical interface for visualizing the gesture recognition process and displaying additional useful information. Since there is a strong trend toward the development of artificial intelligence in the field of computer vision and autonomous vehicles, the software developed here can serve as a basis for future gesture control programs for quadcopters both in the user sector and in the field of industrial UAVs. The potential of this program for use in the military sector is also considerable.

References

[1] J. Kobylarz, J. J. Bird, D. R. Faria, E. Parente Ribeiro, A., Ekárt, "Thumbs up, thumbs down: Non-verbal human-robot interaction through real-time EMG classification via inductive and supervised transductive transfer learning," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 6021-6031, 2020. https://doi.org/10.1007/s12652-020-01852-z.

[2] G. Kiss, "External manipulation of autonomous vehicles," *Proceedings of the IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation,* Leicester, UK, 19–23 August 2019, pp. 248-252. https://doi.org/10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00085.

[3] T. Müezzinoğlu, M. Karaköse, "An intelligent human – Unmanned aerial vehicle interaction approach in real time based on machine learning using wearable gloves," *Sensors*, vol. 21, issue 5, 1766, 2021. https://doi.org/10.3390/s21051766.

[4] PatSeer. Patent Landscape Report Hand Gesture Recognition PatSeer Pro. Available online: (accessed 2 November 2017).

[5] V. I. Pavlovic, R. Sharma, T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, issue 7, pp. 677–695, 1997. https://doi.org/10.1109/34.598226.

[6] R. P. Sharma, G. K. Verma, "Human computer interaction using hand gesture," *Procedia Computer Science*, vol. 54, pp. 721-727, 2015. https://doi.org/10.1016/j.procs.2015.06.085.

[7] M. Z. Islam, M. S. Hossain, R. Ul Islam, K. Andersson, "Static hand gesture recognition using convolutional neural network with data augmentation," *Proceedings of the 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, Spokane, WA, USA, 30 May – 02 June 2019. https://doi.org/10.1109/ICIEV.2019.8858563.

[8] T. G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, Y. Harvill, "A Hand Gesture Interface Device," *ACM SIGCHI Bulletin*, vol. 18, no. 4, pp. 189-192, 1986. https://doi.org/10.1145/1165387.275628.

[9] Y. Liu, Y. Jia, "A robust hand tracking and gesture recognition method for wearable visual interfaces and its applications," *Proceedings of the Third International Conference on Image and Graphics (ICIG'04)*, Hong Kong, China, 18-20 December 2004, pp. 472-475. https://doi.org/10.1109/ICIG.2004.24.

[10] K.-B. Lee, J.-H. Kim, K.-S. Hong, "An implementation of multi-modal game interface based on PDAS," *Proceedings of the 5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007),* Busan, Korea, 20-22 August 2007, pp. 759-768. https://doi.org/10.1109/SERA.2007.48.

[11] V. Pallotta, P. Bruegger, B. Hirsbrunner, "Kinetic user interfaces: Physical embodied interaction with mobile pervasive computing systems," *Advances in Ubiquitous Computing: Future Paradigms and Directions*, IGI Publishing, 2008, 28 p. doi: 10.4018/978-1-59904-840-6.ch008.

[12] M. Panwar, P. S. Mehra, "Hand gesture recognition for human computer interaction," *Proceedings of the International Conference on Image Information Processing*, Shimla, India, 03-05 November 2011. pp. 1-7, https://doi.org/10.1109/ICIIP.2011.6108940.

[13] D. Tezza, M. Andujar, "The state-of-the-art of human–drone interaction: A survey," *IEEE Access*, vol. 7, pp. 167438–167454, 2019. https://doi.org/10.1109/ACCESS.2019.2953900.

[14] R. A. Suárez Fernández, J. L. Sanchez-Lopez, C. Sampedro, H. Bavle, M. Molina and P. Campoy, "Natural user interfaces for human-drone multi-modal interaction," *Proceedings of the 2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, Arlington, VA, USA, 2016, pp. 1013-1022, https://doi.org/10.1109/ICUAS.2016.7502665.

[15] R. Herrmann, L. Schmidt, "Design and evaluation of a natural user interface for piloting an unmanned aerial vehicle," *i-com*, vol. 17, issue 1, 2018, pp. 15–24. https://doi.org/10.1515/icom-2018-0001.

[16] S. P. Kleinschmidt, C. S. Wieghardt, B. Wagner, "Tracking solutions for mobile robots: Evaluating positional tracking using dual-axis rotating laser sweeps," *Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2017*, Madrid, Spain, 26–28 July 2017, pp. 155–164. https://doi.org/10.5220/0006473201550164.

[17] S. Islam, B. Ionescu, C. Gadea, D. Ionescu, "Indoor positional tracking using dual-axis rotating laser sweeps," *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference*, Taipei, Taiwan, 23–26 May 2016, pp. 1–6. https://doi.org/10.1109/I2MTC.2016.7520559.

[18] L. Arreola, A. Montes de Oca, A. Flores, J. Sanchez, G. Flores, "Improvement in the UAV position estimation with low-cost GPS, INS and vision-based system: Application to a quadrotor UAV," *Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, Dallas, TX, USA, 12–15 June 2018, pp. 1248–1254. https://doi.org/10.1109/ICUAS.2018.8453349.

[19] W. A. Hoff, K. Nguyen, T. Lyon, "Computer-vision-based registration techniques for augmented reality," *Proceedings of the Intelligent Robots and Computer Vision XV: Algorithms, Techniques, Active Vision, and Materials Handling, Photonics East'96*, Boston, MA, United States, vol. 2904, pp. 538–548, 1996. https://doi.org/10.1117/12.256311.

[20] S. S. Deshmukh, C. M. Joshi, R. S. Patel, Y. B. Gurav, "3D object tracking and manipulation in augmented reality," *International Research Journal of Engineering and Technology*, vol. 5, issue 1, pp. 287–289, 2018.

[21] E. Shreyas, M. H. Sheth and Mohana, "3D object detection and tracking methods using deep learning for computer vision applications," *Proceedings of the 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, Bangalore, India, 2021, pp. 735-738, https://doi.org/10.1109/RTEICT52294.2021.9573964.

[22] J. Rambach, C. Deng, A. Pagani, D. Stricker, "Learning 6DoF object poses from synthetic single channel images," *Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, Munich, Germany, 16–20 October, 2018, pp. 164–169. https://doi.org/10.1109/ISMAR-Adjunct.2018.00058.

[23] J. Li, C. Wang, X. Kang, Q. Zhao, "Camera localization for augmented reality and indoor positioning: A vision-based 3D feature database approach," *International Journal of Digital Earth*, vol. 13, issue 6, pp. 727–741, 2020. https://doi.org/10.1080/17538947.2018.1564379.

[24] L. Yuan, C. Reardon, G. Warnell, G. Loianno, "Human gaze-driven spatial tasking of an autonomous MAV," *IEEE Robotics and Automation Letters*, vol. 4, issue 2, pp. 1343–1350, 2019. https://doi.org/10.1109/LRA.2019.2895419.

[25] G. Albanis, N. Zioulis, A. Dimou, D. Zarpalas, P. Daras, "Dronepose: Photorealistic Uav-assistant dataset synthesis for 3D pose estimation via a smooth silhouette loss," *Proceedings of the* Workshop on Computer Vision – ECCV 2020: Glasgow, UK, 23–28 August, 2020, pp. 663–681. https://doi.org/10.1007/978-3-030-66096-3_44.

[26] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C. L. Chang, M. Grundmann, "Mediapipe Hands: On-Device Real-Time Hand Tracking", arXiv, 2006. https://doi.org/10.48550/arXiv.2006.10214.

[27] K. Yang, B. Wei, Q. Wang, X. Ren, Y. Xu, H. Liu, "A 3-D depth information based human motion pose tracking algorithms," *Sensors & Transducers*, vol. 174, issue 7, 2014, pp. 253-260.

[28] H. Fesenko, V. Kharchenko, A. Sachenko, R. Hiromoto and V. Kochan, "An Internet of drone-based multi-version post-severe accident monitoring system: Structures and reliability," In book Dependable IoT for Human and Industry - Modeling, Architecting, Implementation. Editors: V. Kharchenko, A. L. Kor and A. Rucinski, River Publishers, 2018, pp. 197-218. https://doi.org/10.1201/9781003337843-12.

[29] I. Zhukov, B. Dolintse, S. Balakin, "Enhancing data processing methods to improve UAV positioning accuracy," *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 16, no. 3, pp. 100-110, 2024. https://doi.org/10.5815/ijigsp.2024.03.08.

[30] O. Fedorovych, et al., "Military logistics planning models for enemy targets attack by a swarm of combat drones," *Radioelectronic and Computer Systems*, vol. 2024, no. 1, pp. 207-216, 2024. https://doi.org/10.32620/reks.2024.1.16.

[31] M. K. Kabir, A. N. Binte Kabir, J. H. Rony, J. Uddin, "Drone detection from video streams using image processing techniques and YOLOv7," *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 16, no. 2, pp. 83-95, 2024. https://doi.org/10.5815/ijigsp.2024.02.07.

[32] I. Paliy, A. Sachenko, V. Koval and Y. Kurylyak, "Approach to face recognition using neural networks," *Proceedings of the 2005 IEEE Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Sofia, Bulgaria, 2005, pp. 112-115, https://doi.org/10.1109/IDAACS.2005.282951.

[33] O. Fedorovich, et al., "Modeling waves of a strike drones swarm for a massive attack on enemy targets," *Radioelectronic and Computer Systems*, vol. 2024, no. 2, pp. 203-212, 2024. https://doi.org/10.32620/reks.2024.2.16.

[34] Y. Sun, H. Fesenko, V. Kharchenko, L. Zhong, I. Kliushnikov, O. Illiashenko, O. Morozova, A. Sachenko, "UAV and IoT-based systems for the monitoring of industrial facilities using digital twins: Methodology, reliability models, and application," *Sensors*, vol. 22, 6444, 2022. https://doi.org/10.3390/s22176444.

***VOLODYMYR SAMOTYY** received a M.S. in Automation from Lviv Polytechnic National University, Ukraine in 1984, a Ph.D. in 1990, and a D.S. in computers, systems and networks, elements and devices of computers and control systems in 1997. He has been Professor since 2001. He is currently a Full Professor at the Department of Automation and Information Technologies, Cracow University of Technology, Poland, and the Department of Computerized Automatic Systems at Lviv Polytechnic National University, Ukraine. His research interests include evolutionary models, numerical methods, information security, and digital signal processing. ORCID: 0000-0003-2344-2576.*

***ULYANA DZELENDZYAK** received a M.S. in Applied Mathematics from Lviv Polytechnic National University, Ukraine in 1989, a PhD in 2006. Since 2009 he has been an Associate Professor of the Department of Computerized Automatic Systems at Lviv Polytechnic National University, Ukraine. Her research interests include evolutionary models, numerical methods, and digital signal processing. ORCID: 0000-0003-0529-8582.*

***NIKITA KISELOV** received a B.S in Internet of Things from Lviv Polytechnic National University, Ukraine in 2021, an M.S in Artificial Intelligence from University Paris-Saclay, France in 2023. His research interests include computer engineering, artificial intelligence, natural language processing and computer vision.*
ORCID: 0000-0003-0297-7811.

***OKSANA SHPAK** received a M.S. in Quality Management from Lviv Polytechnic National University, Ukraine in 2001, and a PhD in 2013. Since 2020 he has been an Assistant Professor of the Department of Computerized Automatic Systems at Lviv Polytechnic National University, Ukraine. Her research interest quality control, diesel and biodiesel and digital signal processing. ORCID: 0000-0002-2093-9029.*