Realtime Human-UAV Interaction Using Deep Learning

Ali Maher[1], Ce Li[2], Hanwen Hu[1], and Baochang Zhang[1(✉)]

[1] School of Automation Science and Electrical Engineering, Beihang University, Beijing, China
ali_mtc@hotmail.com, huhanwenxxx@163.com, bczhang@buaa.edu.cn
[2] China University of Mining and Technology, Beijing, China
celi@cumtb.edu.cn

Abstract. In this paper, we propose a realtime human gesture identification for controlling a micro UAV in a GPS denied environment. Exploiting the breakthrough of deep convolution network in computer vision, we develop a robust Human-UAV Interaction (HUI) system that can detect and identify a person gesture to control a micro UAV in real time. We also build a new dataset with 23 participants to train or fine-tune the deep neural networks for human gesture detection. Based on the collected dataset, the state-of-art YOLOv2 detection network is tailored to detect the face and two hands locations of a human. Then, an interpreter approach is proposed to infer the gesture from detection results, in which each interpreted gesture is equivalent to a UAV flying command. Real flight experiments performed by non-expert users with the Bebop 2 micro UAV have approved our proposal for HUI. The gesture detection deep model with a demo will be publicly available to aid the research work.

Keywords: Human gesture · Micro UAV · YOLO · Deep learning

1 Introduction

The Unmanned Air Vehicle (UAV), also knows as a drone have mainly two types of control; Remotely piloted and autonomous flight by means of On Board Computer (OBC). Most remotely piloted UAVs are driven by means of a physical remote controller. However, the Natural User Interface (NUI) has been proposed recently instead of the physical remote to drive the UAV. Visual human gesture is one of the most appealing method to build a HUI system. It is a computer vision with body language challenges based mainly on how accurately detect and interpret visual human gestures. In this work, we propose a Human-UAV Interaction based on deep detection framework running on a local machine's GPUs. The proposed interpreter developed in the system maps the detected human gesture to a certain UAV flying command. Our contribution mainly are three folds: (1) We introduce an intuitive realtime system for controlling a low cost UAV using our pre-designed set of human gestures. (2) We optimize and

J. Zhou et al. (Eds.): CCBR 2017, LNCS 10568, pp. 511–519, 2017.
https://doi.org/10.1007/978-3-319-69923-3_55

deploy a state-of-the-art object detection architecture, YOLOv2 [1], in the context of Human-UAV interaction. (3) We build a large dataset for human gesture experiments that is efficient and adequate for deep detection framework training. Figure 1(a) shows our proposal being tested indoor with two different UAVs.

Fig. 1. (a) Our HUI proposal being tested with Parrot AR. drone 2.0 *(up)* and Bebop 2 *(down)*. (b) Samples from our dataset were taken in different environments.

The rest of the paper is structured as follows. Section 2 reviews and compares other similar works. Section 3 introduces our human gesture dataset. Section 4 presents the overall system architecture and explains the gesture's interpreter in detail. Section 5 describes the implementation of the proposed deep detection network. Sections 6 and 7 are experimental results and conclusions, respectively.

2 Related Work

Plentiful amounts of research projects proposed a UAV navigation using human gestures based interface. Most of them [2–4] have used the Microsoft Kinect sensor device at the ground station. They exploited its depth and RGB images to deduce the human skeleton model. Although, many gestures can be built from the skeleton model. But, involving the Kinect with its driver and API library made the system costlier and has a noticeable latency. The author of [5] introduced a HUI proposal based on leap motion sensor which has two monochromatic IR to track hand and fingers. Then, leap motion controller software will analyze tracked gestures on the ground station. Our HUI proposal differs from those; we have used a conventional USB digital camera to acquire human gestures at the ground station. Moreover, we can use the UAV's onboard camera directly instead of the USB one with our proposal, which cannot be applied for Kinect or

leap motion based systems. In [6], the author used the UAV's onboard camera to detect and track the operator's face using a Harr cascade classifier with a Kalman filter. Then, applying color-based segmentation to detect colored (by means of gloves) right and left hands. Our proposal also differs from that proposal. Where, we detected the operator's hands and face by means of high level features which have more discriminating power than the hand crafted one. Also, we applied a simple trick in our interpreter to discriminate between left and right hand not by wearing a colored glove which eased hand-tracking by color matching. So far as we know, HUI with hand gestures based on deep detection network has not been proposed yet. In this paper, we focus on involving YOLOv2 framework in controlling a micro UAV with simple and intuitive human gesture.

3 Human Gesture Dataset

Our dataset includes 6223 images focusing on human's hands and faces. There are 23 volunteers involved in the collection of this dataset. The camera resolution is of 1920×1080, 30 frames per second, and the duration of each video is about 30 s. There are 48 videos were taken into two backgrounds - indoor (18) and outdoor (38). We obtain one frame from videos every 5 frames, then all the images are filtered in order to discard redundant data. Aiming to get the key information about human's hands and faces, we crop images to remove the extra background. Ultimately, each image's resolution is 1200×1000 or less. Figure 1(b) presents some random samples from our dataset.

4 System Architecture Overview

Our HUI proposal tele-navigate the UAV in a denied GPS environment with the hand postures. Figure 2 shows a high level description of our proposed system and its fundamental components. All of them were implemented in the

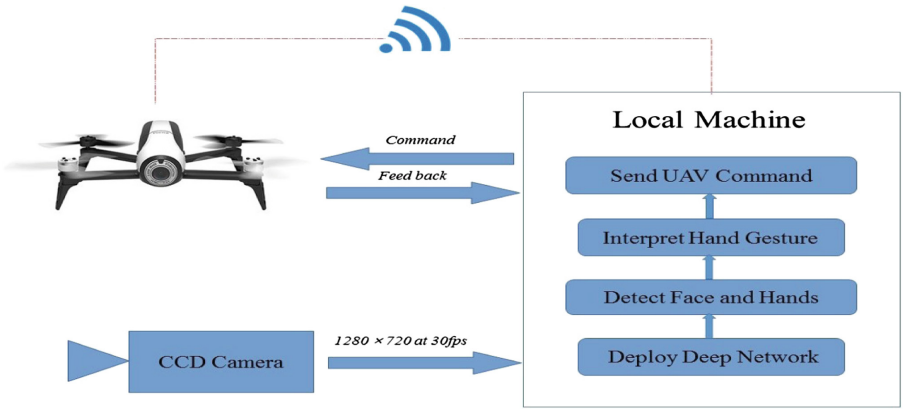

Fig. 2. High level architecture of our proposed HUI with Bebop 2 drone.

Robot Operating System (ROS) framework as nodes. Where, each node can (publish/subscribe) communicate with the other nodes via ROS transport protocol [7]. The ground station is a laptop (with an Intel core i7-5500 @ 2.4 GHz and equipped with NVIDIA GPU GeForce GTX 840M). The proposal being tested with two Parrot's UAVs; AR. drone 2 and Bebop 2.

4.1 Gesture Interpreter

The deep detection network will detect the operator gesture in each frame, therefore it will work as an observation model [8,9] but in the absence of the motion model. So, the detected gesture in the current frame and the previous one does not relate to each other. This led us to use static (posture) not dynamic (sequence of postures) gesture recognition in our proposal. Where, the control command depends on detected gesture in the current frame and will not change until the threshold is breached again by a new gesture [10].

4.2 Gesture Design

The operator's face, right and left hands were used as gesture source. The deep detection node will publish three bounding boxes annotated with hand, face and hand. The interpreter will take the hand which has lower horizontal location (x-value) as a *right* hand and the other one as a *left* hand. We use the ratios between vertical centers (y-values) of the three bounding boxes to build the gestures.

$$R_1 = \frac{Y_{cf}}{Y_{rh}}, R_2 = \frac{Y_{cf}}{Y_{lh}} \tag{1}$$

Figure 3(a) demonstrates how the R_1 and R_2 (yellow lines margins) are determined from the bounding boxes of the detected gesture. Thus, these ratios provide the scale invariance, so, it does not matter where the operator will stand in front of the ground station camera. A total of nine natural static hands postures are designed for the UAV controlling. Table 1 summarizes the correspondence between hand postures and UAV controlling commands. Figure 3(b) shows different gesture postures designed using human's hands and face. We state three important rules in designing the gesture interpreter as follows: (1) To avoid the effect of the false positive gesture, the interpreter will be invoked if and only if (two hands and one face) are detected and with high confidence scores. (2) We use the UAV current state flags (Taking off, Landed, Hovering, Flying) to avoid the conflict which may come from disordered commands (taking off should be after landing or controlling command should be after hovering, etc.). (3) Hovering is a default command when the operator gesture does not satisfy any threshold mentioned in Table 1. The UAV states are acquired within UAV navigation data which measured by its on board sensors. Finally, the interpreter maps the detected human gesture to a controlling command as mentioned previously.

Fig. 3. (a) Demonstrates of how the ratios R_1 and R_2 are determined from the bounding boxes. (b) Different gestures are designed to control the UAV. (Color figure online)

Table 1. The correspondence between postures and the UAV controlling commands.

Hand's postures w.r.t face	R_1	R_2	Corresponding command
All Aligned vertically	0.9 : 1.1	0.9 : 1.1	Take-Off/ Hovering
Two hands aligned vertically and beneath the face	0.4 : 0.6	0.4 : 0.6	Landing
Two hands aligned vertically and above the face	2.0 : 2.2	2.0 : 2.2	Fly up
Right hand aligned with the face and left hand above	0.9 : 1.1	> 2.4	Fly backward
Left hand aligned with the face and right hand above	> 2.4	0.9 : 1.1	Fly forward
Right hand aligned with the face and left hand beneath	0.9 : 1.1	< 0.5	Fly left
Left hand aligned with the face and right hand beneath	< 0.5	0.9 : 1.1	Fly right
Right hand above the face and left hand beneath	> 2.4	< 0.5	Yaw CCW
Left hand above the face and right hand beneath	< 0.5	> 2.4	Yaw CW

5 Deep Detection Network

We trained Faster RCNN [11] and Tiny-Yolo [1] by 3733 images from our dataset. The remaining unseen frames (2490) were used in testing to get an indication

for the network accuracy. Although, Tiny-YOLO is fast, but it achieved gesture detection with a speed of *15* FPS when running on our ground station laptop. Tiny-YOLO comprises 9 convolutional layers, only 6 of them followed by pooling layers for down sampling. We modified Tiny-Yolo architecture to be 6 convolutional layers, only 4 of them followed by pooling layers. This will aid our detection system in two aspects: (1) 6-layers Tiny-YOLO achieved gesture detection with speed more than *21* FPS when running on our ground station laptop. (2) Reducing number of the pooling layers will preserve more spatial information (such as face and hand postures) which is crucial for our application. The running time was measured in terms of the ROS topic publishing rate of the deep detection node. Thus, the pre-processing for image conversion, image loading and parsing of the detection results are included.

The architecture of the proposed 6-layers is set in the Yolo configuration file as follows: *C1,P1,C2,P2,C3,P3,C4,P4,C5,C6*. Where, C is for convolutional layer and P for pooling. Leaky ReLU activation was used with all convolutional layers, except last one ($C6$) was activated by a linear function. All kernels sizes are 3×3 with stride 1 and padding 1 except for last layer the kernel size is 1×1. The number of filters (feature maps) for C1,C2,C3,C4,C5,C6 are 16,32,64,128,1024,14 respectively. We tailored the anchors (reference boxes) to fit the detected gestures aspects ratios and scales. Where, k-mean clustering algorithm was applied to get the centroids (reference boxes) of our training dataset. We used the silhouette [12] average score to determine the best number of centroids (k). We got two anchors $(k = 2)$ for each network (Tiny-YOLO and 6 layers Tiny-YOLO).

6 Experiments

6.1 Experiments on Face and Hands Detection

Tailoring the bounding box is very important in our application, where the success of the interpreter relies on how accurately the detection network localizes the gesture with a tight bounding box. Figure 4 shows how the anchors adaptation affects the detection results. Multi-scale training is activated (resize the network input image) from 320×320 up to 608×608 with a step of 32. This will aid the deep network to detect properly the operator gesture with many resolutions (gives some flexibility to use different digital cameras) at the ground station.

Faster RCNN is trained and tested to have a baseline for a comparison with the trained Tiny-YOLO and 6-layers Tiny-YOLO. Table 2 presents the mean Average Precession mAP for Faster RCNN, Tiny-YOLO and proposed 6-layers Tiny-YOLO on our data testing set. From Table 2 obviously, we can achieve the required real time for gesture detection beside preserving the same detection accuracy in terms of mAP. It is noticed that the detection results are almost high and equal. Where, the detection (not recognition) of human physical characteristic (such as the face) with a massive training set is not a complex task.

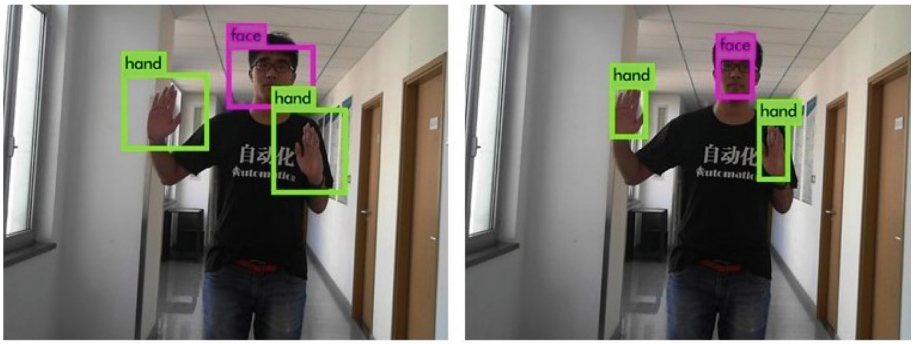

Fig. 4. Shows detection results using (a) (VOC) anchors, (b) our determined anchors.

Table 2. Detection accuracy results, the speed was measured on our local machine.

Method	Hand	Face	mAP	Speed (FPS)
Faster R-CNN	0.9034	0.9088	0.9061	1
Tiny-YOLO	**0.909**	**0.909**	**0.909**	15
6-layers Tiny-YOLO	0.9085	0.9088	0.9087	**21**

Inferring that, our proposed dataset for human gesture is large enough and sufficient. Finally, in our system, we used the proposed 6-layers Tiny-YOlO with 21 FPS, which achieves real time interaction in controlling the micro UAV.

6.2 Experiments on Human-UAV Interaction

We use the same evaluation done by [2,4] to judge the performance of our HUI proposal. Where, real indoor flight test with gesture control was conducted inside a room of 6.5 m × 15 m × 3.5 m Dimensions. So, the UAV position was not tracked by the GPS. We collected all navigation data measured by Bebop's onboard sensors which mentioned before. To test all proposed gesture, a non-expert user performed gestures to fly the Bebop 2 UAV in this sequence; take-off, flying up, flying forward, flying left, flying backward, flying right, yawing 180° CCW, yawing 180° CW then landing. Figure 5(a) shows the 3D plot of the acquired (x, y, z) position data. The total flight time for the mentioned flying sequence was around 85 s. Figure 5(b) illustrates the UAV's attitude *(yaw, pitch, roll)* during the flight test. Where, the UAV made pitch and roll angles to fly forward/backward and right/left, respectively. Before the end of the flight (from the second 67 to 77), the UAV made yawing 180° CW then CCW.

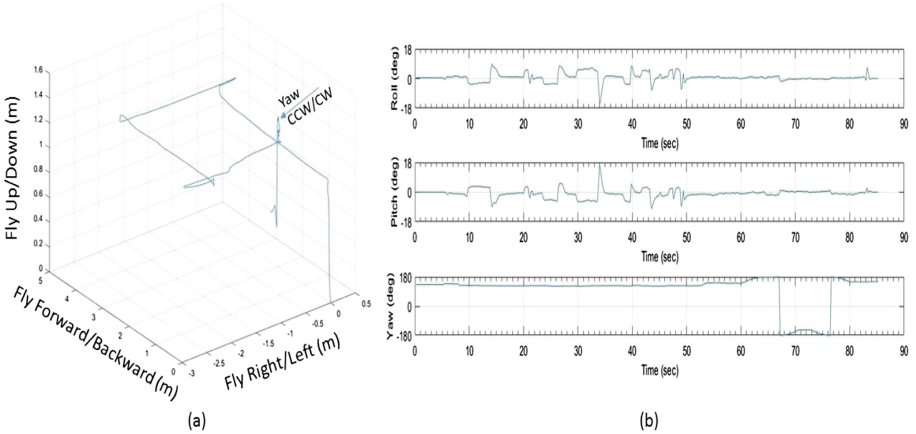

Fig. 5. (a) 3D plot of the acquired position data for the UAV during real flight test, (b) Illustrates the UAV attitude during the real flight test *roll*, *pitch* and *yaw*.

7 Conclusion

In this paper, we introduce an intuitive realtime system for controlling a low cost UAV using human gestures. With our new collected dataset built for human gesture, we design an interpreter mapping each gesture to a controlling command. We investigate how to deploy the state-of-the-art object detection framework YOLOv2 in the context of HUI. Our proposal with that investigation achieved real time and preserved high gesture detection accuracy.

Acknowledgments. The work was supported in part by the Natural Science Foundation of China under Contract 61672079, 61473086 and 61601466. The work of B. Zhang was supported in part by the Program for New Century Excellent Talents University within the Ministry of Education, China, and in part by the Beijing Municipal Science and Technology Commission under Grant Z161100001616005. Baochang Zhang is the correspondence.

References

1. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. arXiv preprint arXiv:1612.08242 (2016)
2. Mashood, A., Noura, H., Jawhar, I., Mohamed, N.: A gesture based kinect for quadrotor control. In: 2015 International Conference on Information and Communication Technology Research (ICTRC), pp. 298–301. IEEE (2015)
3. Boudjit, K., Larbes, C., Alouache, M.: Control of flight operation of a quad rotor AR. drone using depth map from microsoft kinect sensor. Int. J. Eng. Innov. Technol. (IJEIT) **3**, 15–19 (2008)
4. Sanna, A., Lamberti, F., Paravati, G., Manuri, F.: A kinect-based natural interface for quadrotor control. Entertain. Comput. **4**(3), 179–186 (2013)

5. Suarez Fernandez, R., Sanchez Lopez, J.L., Sampedro, C., Bavle, H., Molina, M., Campoy Cervera, P.: Natural user interfaces for human-drone multi-modal interaction. In: Proceedings of 2016 International Conference on Unmanned Aircraft Systems (ICUAS), ETSI_Informatica (2016)
6. Nagi, J., Giusti, A., Di Caro, G.A., Gambardella, L.M.: Human control of UAVs using face pose estimates and hand gestures. In: Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction, pp. 252–253. ACM (2014)
7. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software, vol. 3, p. 5, Kobe (2009)
8. Zhang, B., Perina, A., Li, Z., Murino, V., Liu, J., Ji, R.: Bounding multiple gaussians uncertainty with application to object tracking. Int. J. Comput. Vision **118**(3), 364–379 (2016)
9. Zhang, B., Li, Z., Perina, A., Del Bue, A., Murino, V., Liu, J.: Adaptive local movement modeling for robust object tracking. IEEE Trans. Circuits Syst. Video Technol. **27**(7), 1515–1526 (2017)
10. Zhang, B., Yang, Y., Chen, C., Yang, L., Han, J., Shao, L.: Action recognition using 3D histograms of texture and a multi-class boosting classifier. IEEE Trans. Image Process. **26**(10), 4648–4660 (2017)
11. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems. pp. 91–99 (2015)
12. Zhou, H.B., Gao, J.T.: Automatic method for determining cluster number based on silhouette coefficient. In: Advanced Materials Research, vol. 951, pp. 227–230. Trans Tech Publ (2014)