Drone control using gestures

Jaka Cikač[1], Friedrich Fraundorfer[2], and Danijel Skočaj[1]

[1]*University of Ljubljana, Faculty of Computer and Information Science*
[2]*Graz University of Technology, Institute of Computer Graphics and Vision*
E-mail: jaka.cikac@gmail.com

Abstract

Quadcopters are becoming more popular and integrated into modern society. Currently we are familiar with controlling quadcopters via our mobile phones. In this work we develop a gesture control system that can be used on a low-cost quadcopter equipped with a simple RGB camera and a powerful embedded computer. The system is split into three modules - action detection with optical flow, human pose estimation with CNNs and gesture classification with relational features computed on the human pose. We assembled our own dataset called DS2017, in which 640 gestures are performed by 20 people. We show that action detection can detect actions sufficiently well, the human pose estimation works very well at high speed and gesture classification achieves high accuracy.

1 Introduction

As quadcopters get smaller and more affordable there are many new possibilities for their use. It is not uncommon to find drones at many university laboratories, research institutes and now even normal households. This opens up the possibility of making drones a part of people's lives to perform simple tasks or simply be very interesting toys. There are countless enthusiasts using drones equipped with the latest video capturing technology to provide stunning aerial views. However, there is always a needed component - *control*. Without the ability to control drones in a safe and efficient manner they are not really useful. There are now various ways of controlling the drones from manual flight (remote controller) to mobile application control. Manufacturers are even using VR to provide first person views of drone's flight using special goggles (FPV) and on-board cameras.

This work will focus on controlling the drones by using gestures. Hands free control would enable the operators to focus more on what they are doing, be it an activity that requires their physical engagement or rather just enjoying the scenery.

1.1 Problem definition

We set out to lay the ground work on how drones could be controlled using full-body gestures without any external devices, such as motion detectors and RGB-D cameras. Instead, the system is computer vision based. In order to be able to "see" the gestures, a drone should be equipped with an RGB monocular camera, from which a video feed of the user can be processed using computer vision, to estimate the human pose and further recognize and classify the gestures that are performed.

1.2 Related work

The field of gesture recognition has, in recent years, been explored to many depths, which can be seen from the many surveys that have appeared, such as [1].

One example of gesture controlling drones is [2]. In their work authors researched, which gestures seem natural for users to interact with the drone and what kind of modality (gestures or voice) the users used. They found that users used the same gestures as people use for conveying information to their pets or even interpersonal gestures such as come here, stop, come closer, move left, etc.

The main limitations of gesture control on drones come from computational power required on the on-board processor, it's camera system and it's payload capacity. For that reason most gesture control systems that were developed, require external devices and an off-line computer that could run the recognition algorithms. One of such examples is [3] where authors used a Parrot AR Drone and Microsoft Kinect to evaluate different metaphors for conveying controls for a variety of flying operations supported by the UAV.

Researchers in [4] created an implementation of gesture control using an inexpensive drone, the Parrot AR, and it's on-board RGB camera. The system was based on face tracking and hand gestures. Since the Parrot AR is lacking an on-board computer, everything needed to be processed on an off-line PC. The user would also have to wear colored gloves, so that the drone would be able to detect and track them. It also limited drone's movements in a very restricted space near the user.

One example of gesture control where, in theory, the UAV is not limited to the area near the user is [5], where authors wanted to track and recognize gestures for signaling aircraft. They created a dataset of NATOPS aircraft handling signals and a unified framework for body and hand tracking. Our work was partially inspired by this framework, but there are major differences. Authors of [5] use a 3D camera that is stationed on the ground. However we aim to implement a similar framework directly on the UAV with only a monochrome camera. In

this paper we present the main functionalities of the developed system and summarize the experimental results; further details are given in [6].

2 System design

The gesture control system is composed of three main components. The first important component of the system is *action detection*. We use action detection in order to improve computational efficiency of the whole system. The second component is *pose estimation* in order to obtain important features for *gesture classification* which infers the performed gesture. This final component provides the command to the drone. We need to detect when the action or gesture is taking place in a video sequence so that we can focus the processing power on a particular segment on which we use complex pose estimation algorithms that provide us with important features. The result of pose estimation is a set of joint positions as coordinates in an image. We use a state of the art method for pose estimation [7]. This method is based on deep convolutional neural networks (DCNNs). This choice obliges us to use a specific powerful GPU, that is not available on the drone. Despite this the system is still ready for real-time and on-line deployment. Gesture classification has been implemented on top of pose features in a bag of words approach. We learn gestures as words and then try to match a newly detected action to these words using an SVM classifier. In the following subsections we describe in more detail each component of the gesture control system and its underlying methods. A system overview diagram is shown on Figure 1.

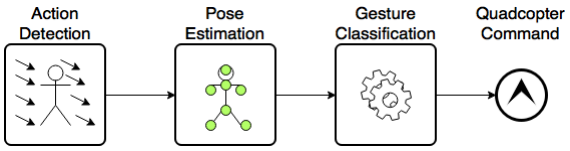


Figure 1: Three modules of the gesture control system: action detection, pose estimation, and gesture classification.

3 Methodology

3.1 Action Detection with person tracking

Action detection is split into three sub modules. We first detect the person with a *person detector* based on Histograms of oriented gradients [8], which provides initialization coordinates for a short-term tracker Adaptive Scale Mean Shift (ASMS) [9] that keeps track of a person through subsequent frames. We chose this particular tracker because it is extremely fast and performs well due to its awareness of background appearance and scale adaptation. Since tracking is a hard task, we make it easier by re-detecting a person every number of frames. After we are sure that we have the person localized we employ a dense grid around the person in which we calculate the optical flow using the Lucas-Kanade method. We filter the optical flow using RANSAC [10], so that only the person generated optical flow points remain. After filtering we count the optical flow points in 5 consecutive frames that form a chunk and classify it with a linear SVM, which decides if there was enough points, or not, to consider the chunk as an action. Chunks are buffered into circular buffers, which provide continuous functionality and additional filtering of false positives and false negatives. A flow chart of action detection with its submodules is shown on Figure 2. Frames on which action was detected are forwarded to a real-time pose estimation module.

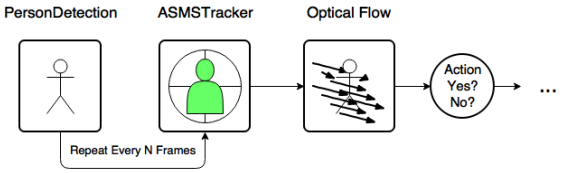


Figure 2: Overview of the action detection sub modules: person detection, visual tracking, optical flow estimation, determining action presence.

3.2 Human Pose Estimation

We chose a method called Real-time Multi-Person 2D Pose Estimation using Part Affinity Fields [7] (MPE-PAF) as the main method for real-time person estimation. It was the first method to deliver real-time performance for pose estimation and it works for multiple people at the same time. Like CPMs [11] it is a method based on deep convolutional neural networks and inherits many ideas of CPMs, which were proven to have a very accurate pose estimation performance. MPE-PAF is a bottom-up approach that does not require a person detector. It has an architecture for jointly learning parts detection and parts association, which are then parsed with a greedy parsing algorithm to produce human pose estimation. This approach is not as accurate as CPMs but it does not perform much worse. Its main advantage is of course speed. Benchmarks were ran on a laptop version of NVIDIA GeForce 1080-GTX GPU, with which the authors obtain real-time performance of 8.8 FPS for a video with 19 people, making it an ideal choice for a real-time gesture control system.

3.3 Gesture Classification

After the pose is estimated, we are given a set of body joints for each frame in the detected action, which allows for certain features to be computed. The descriptor, which is a combined set of various features, computed on the body joint positions, is referred to as *Pose Features*. This descriptor combines positional features and relational features that describe geometry between combinations of joints. It was shown that a combination of these two types of features produce the best results for gesture classification [12]. All joint positions are normalized to $[0, 1]$. After normalization of joint positions, pose features are computed. In this work we compute features on a set, which is annotated with 15 joints per frame. We

calculate position features, distance features, orientation features, angle features and temporal features.

Position features are simply the normalized joint coordinates. *Distance features* are computed between combinations of pairs of joints. For joint pairs the distance between the joints is computed. *Orientation features* between joint pairs are computed as the orientation of the vector connecting two joints with respect to the neck-to-belly orientation. Then *angle features* are computed, represented by the inner angle of two vectors that span triplets of joints. Finally *temporal features* are computed. Temporal features consist of differences of features between two or more adjacent frames. In total this gives us 3180 features per frame, which make up the Pose Features frame descriptor.

After computing Pose Features we perform K-means clustering to quantize the descriptors into clusters. After the centers of clusters are returned we calculate the distance of each frame descriptor to each cluster and find the closest cluster center. After this is done for each frame, we end up with a histogram representing the whole video (action). The histogram is then normalized. Histograms are collected into a database and used as features for training the SVM. We use a radial basis function for the kernel.

For feature classification we therefore use a Bag of Words method trained on such descriptors. In the inference stage a similar procedure is used. After the pose is estimated on the frames that were deemed to contain an action, Pose Features are computed on each of them and the closest cluster center is found for the assembled descriptor. Then a histogram is assembled, normalized and then sent to the SVM for classification, which produces the result - a given action by the user.

4 Experimental results

4.1 DS2017 dataset

In our experiments we used 4 main gestures, namely *up*, *down*, *left* and *right*. For these 4 gestures we have created a *controlled gestures* dataset and an *intuitive* dataset. Each of these two categories is further split in a set of videos with a *steady camera* and the second with a *moving camera*, simulating the movements of the drone. Therefore we have 4 sub-sets in total, *controlled-steady set*, *controlled-unsteady set*, *intuitive-steady set* and *intuitive-unsteady set*.

The DS2017 consists of videos of 20 people, performing 640 gestures in 80 videos. Every video is annotated for action detection with 5 frames granularity and split into gestures then categorized in one of the 4 main categories. We limited the scenes to relatively homogeneous backgrounds, with equal illumination throughout the video. Although there are some examples where this does not hold. There is always a single person in the video, which is performing a gesture and in general the person does not move around on the scene. Two examples of video sequences showing the gesture *up* are presented in Fig. 3.

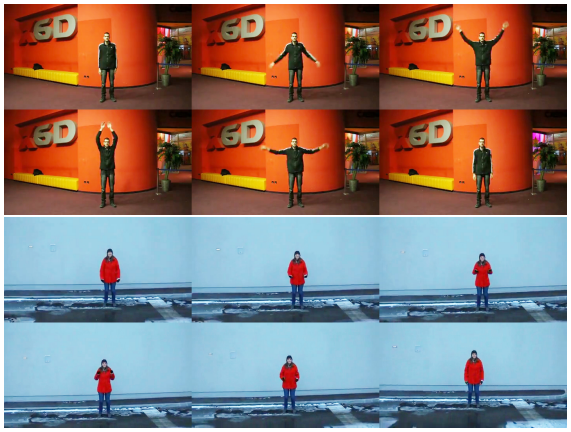


Figure 3: DS2017 dataset; two examples of gesture *up*: controlled (top) and intuitive (bottom).

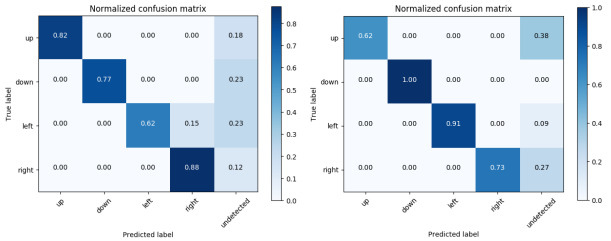
The purpose of a *controlled gesture* set is to have a very distinct set of gestures, that are very exuberantly performed in a standardized manner, potentially resulting in more optical flow points and have similar Pose Features descriptors, even when different people perform the gesture. It is meant to serve as a standardized controlled experiment, on which the algorithms would be evaluated. A more challenging part of the dataset are *intuitive gestures*, which are more natural to how a human would gesture a drone, intuitive and more fluent. These actions were not thoroughly regulated or explained to the people who would perform them in order to encourage variability of gestures from person to person.

We created two types of such datasets considering different acquisition conditions. *Steady* sub-sets were taken on a stabilized camera, so that there is no movement of the background. *Unsteady* videos were taken without any stabilization and random movements of the camera were introduced. These movements are meant to simulate a flying drone, with issues with stabilization or influence from the windy environment.

4.2 Performance evaluation

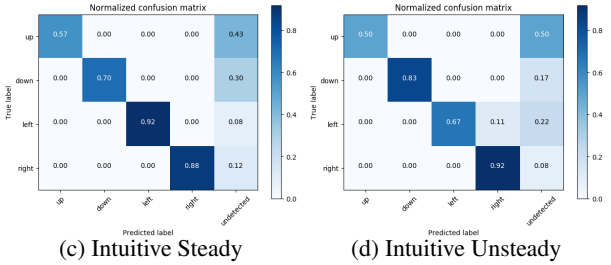
We evaluated the proposed system on our dataset DS2017. Here we report the classification accuracy after the video is sent through the whole pipeline. We evaluate the gesture sets separated on *steady* and *unsteady* sets. The action detection algorithm detected the action in **83,13%** of cases, or on 133 out of 160 test gesture videos in total. In the pose estimation model most issues occur with gestures *intuitive down* and *controlled down* due to the crisscrossing of the person's hands in front of their body. The pose estimation also struggles when there are hard shadows and under exposed frames. The movement of the drone does not impact the quality of the pose estimation algorithm.

Gesture classification accuracy is considered with undetected gestures taken into account, to show the complete system performance. In other words, if the action detection module failed to detect an on-going action, it is treated as miss-classification. For each category the classification accuracy is presented in a confusion matrix,



(a) Control Steady

(b) Control Unsteady



(c) Intuitive Steady

(d) Intuitive Unsteady

Figure 4: Confusion matrices for the evaluation of the gesture control system on DS2017 categories, which include the undetected gesture rates. If the gesture was not detected with the action detection module, it is treated as miss-classified and presented as category "undetected".

which also includes undetected gestures, on Figure 4. If action detection successfully detects the action however, the classification accuracy is 96.8%.

The highest overall accuracy is achieved on the *Controlled unsteady* gesture set, with an overall accuracy of 81%, as shown on Figure 4b. The lowest classification accuracy is achieved on *Intuitive unsteady* gesture set, as shown on Figure 4d, with an overall accuracy of 73%. The results confirm that the action detection module is struggling with detecting gesture *up*, which decreases the classification accuracy in all categories, except for category *Control steady*, where the gesture *right* has the lowest accuracy. The overall accuracy achieved for *Intuitive steady* gesture set is 76%. Overall accuracy for *Controlled steady* gesture set is 77%.

4.3 Runtime evaluation

From a detected action to the final gesture prediction it takes **2,14 s**, measured on a PC with a Core i7 4770K (3.5 GHz). The whole pipeline would run slower on a quadcopter, due to a less powerful CPU. However, most time is spent for pose estimation (*2,03 s* for a gesture on 25 frames), run on an external PC, so the total time would not be much longer. The entire pipeline would work in real-time on an on-board computer, if equipped with a sufficiently powerful GPU, such as a future version of NVIDIA Jetson TX 2.

5 Conclusion

We developed a three-phase gesture control system, by first detecting when the action is happening on the video, with a fast person detector, a fast tracker and optical flow. After detecting the action, we use a state of the art method to estimate the human pose on the frames that contain an action. The method uses DCNNs and is able to achieve real-time performance. After getting the locations of joints in the human pose, we compute relational Pose Features, that provide features for SVM classification, which predicts the final gesture.

We present the dataset DS2017 that features 640 gestures performed by 20 people. The system is able to detect actions in 83% of cases, having more success with the controlled gestures, while intuitive gestures are a harder challenge. We also found that the method [7] is extremely good in estimating the human pose on our dataset, struggling only when the hands criss-cross, as seen in gestures "down". We are also able to reach a high classification accuracy of 96.8% on our DS2017 dataset for the final gesture prediction.

References

[1] H. Cheng, L. Yang, Z. Liu, Survey on 3D hand gesture recognition, IEEE Transactions on Circuits and Systems for Video Technology 26 (9) (2016) 1659–1673.

[2] J. R. Cauchard, K. Y. Zhai, J. A. Landay, et al., Drone & me: an exploration into natural human-drone interaction, in: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM, 2015, pp. 361–365.

[3] K. Pfeil, S. L. Koh, J. LaViola, Exploring 3D gesture metaphors for interaction with unmanned aerial vehicles, in: Proceedings of the 2013 international conference on Intelligent user interfaces, ACM, 2013, pp. 257–266.

[4] J. Nagi, A. Giusti, G. A. Di Caro, L. M. Gambardella, Hri in the sky: controlling uavs using face poses and hand gestures.

[5] Y. Song, D. Demirdjian, R. Davis, Tracking body and hands for gesture recognition: Natops aircraft handling signals database, in: Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on, IEEE, 2011, pp. 500–506.

[6] J. Cikač, Drone control using gestures, Master's thesis, University of Ljubljana, Faculty of Computer and Information Science, Slovenia (2017).

[7] Z. Cao, T. Simon, S.-E. Wei, Y. Sheikh, Realtime multi-person 2D pose estimation using part affinity fields, in: CVPR, 2017.

[8] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1, IEEE, 2005, pp. 886–893.

[9] T. Vojir, J. Noskova, J. Matas, Robust scale-adaptive mean-shift for tracking, Pattern Recognition Letters 49 (2014) 250–258.

[10] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395.

[11] S.-E. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh, Convolutional pose machines, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4724–4732.

[12] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, M. J. Black, Towards understanding action recognition, in: International Conf. on Computer Vision (ICCV), 2013, pp. 3192–3199.