Visual Communication with UAS: Recognizing Gestures from an Airborne Platform

Alexander Schelle(✉) and Peter Stütz

Institute of Flight Systems, University of the Bundeswehr Munich, Neubiberg, Germany
{alexander.schelle,peter.stuetz}@unibw.de

Abstract. Current tactical unmanned aerial systems receive their guidance and tasking information predominantly via radio links. To be able to communicate with these systems specific electronic devices are required. This work builds on the concept of visual communication of UAS to allow a person on ground commanding a nearby airborne vehicle to perform a specific reconnaissance task via gestures. A procedure to collect the necessary gestural command components is presented as well as a prototype image processing flow which is able to distinguish between neutral poses, static and dynamic 2D gestures. Prototype experiments prove the applicability of the proposed method on real life data from an airborne platform.

Keywords: UAS · Gesture recognition · Visual communication

1 Introduction

Nowadays tactical unmanned aerial systems (UAS) utilize radio links to receive mission relevant information from a ground control station, a mobile control device or from other manned aircrafts. As technical devices are required to communicate with airborne systems, there is currently no option to transfer the authority of the UAS to third parties lacking adequate equipment. For example infantryman on ground who require a temporal access to an asset for an up to date overview of the situation and processed image intelligence results in their area of operation. For such use case new ways of interaction have to be found.

A promising candidate for such new paradigm is given by visual communication, which, however, is constrained by narrow information bandwidth at only close distance. Hereby flags, light signals and hand gestures are commonly used tools to transmit information by optical means in general. The latter is thereby of peculiar interest, as this enables the abandonment of any additional equipment on ground. For that, the movements and gestures of a person on ground have to be captured with a suitable imaging sensor on board the UAV. Further this data has to be processed by a gesture recognition system to classify the gestural movements and finally translate them into commands to generate a complete task description. For this a performant computational system that can operate in real time is necessary to enable a low latency communication. The processed and interpreted data can then be used to generate a flight path and if applicable a plan of action for the mission sensors deployment. A downstream flight management system (FMS) will use this information for the flight guidance of the unmanned aerial vehicle. An overview of the necessary system components is shown in Fig. 1.

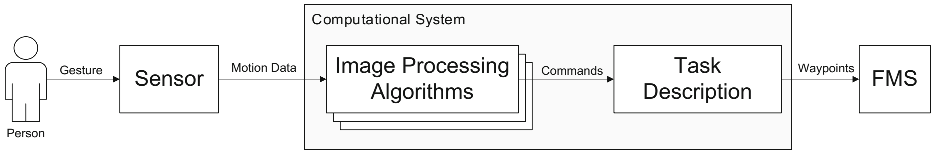

Fig. 1. System components for a gesture based commanding

Modern pose and gesture recognition system have reached a high level of reliability and accuracy [1–5]. But the majority of those systems were designed for indoor use and a static mounting of the sensor. Recognizing gestures from a flying platform outdoors, however, is a more challenging task, where the system has to deal with several constraints and uncertainties:

- Available additional weight and energy allowances are limited on aerial platforms, hence only lightweight and low power consuming components can be used.
- Concerning data processing, the sensor is in non-stop motion, so classical background subtraction methods can be applied only to a certain amount to detect movements of the operator. Therefore a more robust and reliable person detection and tracking is required.
- The gesture sensor does not always work in its optimal range, as the distance to the user changes constantly leading to a varying data accuracy.

Ongoing work on suitable and intuitive gestures for human drone interactions [6–8] is increasing thus confirming the rising demand for more natural interfaces. A concept for a bidirectional communication and a gestural commanding of a UAS for reconnaissance purposes has been presented recently [9]. The following chapters give a brief introduction to this concept and focus on the specifics of the used gestural syntax. A method to allow the discrimination of static and dynamic gestures needed for the commanding is presented there as well. Lastly a prototypical implementation of the proposed concepts components will be evaluated on real life data from an airborne platform.

2 Approach

The concept for the visual communication with UAS suggests a syntax for gestural commands, that is composed of the four mandatory components *task declaration, direction, distance, post-task behavior* and the optionally component *time constraint*. In the intended application a serial execution of gestures that represent these components defines a specific reconnaissance task typical for a UAV mission. To obtain this information, two operational modes are recommended: a *detection mode* to spot potential interaction request of an operator on ground from medium altitudes using a thermal imager and an *interaction mode* that receives the actual gestural commands from a closer distance using a depth sensor. This work focuses on the systemic components that are necessary to realize the acquisition of the command components in the latter.

2.1 Gestural Syntax

The gestural components of the proposed syntax can be transmitted without a specific order basically. However some commands require a subsequent gesture of a specific type for definitions or specifications. For instance, if the surveillance task is to detect an object, the system must be informed on the object type of interest (humans, vehicles, etc.). Therefore each gestural command component is encapsulated into a gestural sequence that is passed along all processing blocks of the system and contains the information about possible dependencies. The structure of such a container is shown in the following Table 1.

Table 1. Structure of a gestural sequence container

Gestural command component	Dependence flag	Subsequent gesture type

The gestural sequence contains the component itself (*task declaration, direction, distance*, etc.). It is followed by a flag that represents the dependency to a subsequent gesture and a type definition for the following gesture. The gestural command components *direction* and *distance* are independent of other gestures, but the components *task declaration*, *post-task behavior* and *time constraint* determine the succeeding gestures. Table 2 gives a brief overview.

Table 2. Overview of gestural command components and their dependencies

	Task declaration	Direction	Distance	Post-Task behavior	Time constraint
Dependent on subsequent gesture	Yes	No	No	Yes	Yes
Subsequent gesture type	Static, Dynamic	-	-	Static, Dynamic	Static
Following subcomponent	Object definition	-	-	Directional	Numerical

Gesture Separation Signals

The gesture recognition system needs to recognize when one gesture ends and the next one begins. Therefore two criteria have been selected to prompt the separation. A separation signal gets triggered, once the operator's movement remains under a definable lateral threshold for a particular time. It can be assumed that the operator is then either in a neutral pose (*no input*), is pointing somewhere (*directional input* or *object type definition*) or is showing something (e.g. *numerical information*). The correct sizing of both thresholds is essential for the reliability of the method. Therefore these thresholds have to be designed adaptively in regard to the individual pace of the operator.

The separation signal will also be trigged, if the operator enters or leaves the so called *neutral pose space*. This space is a three-dimensional box that covers the operator completely for that case, when both of his arms are close to his body. To consider potential uncertainties introduced by noisy depth data or inaccurate body part position estimation, this space is slightly larger than the actual body dimensions. This principle is shown in Fig. 2.

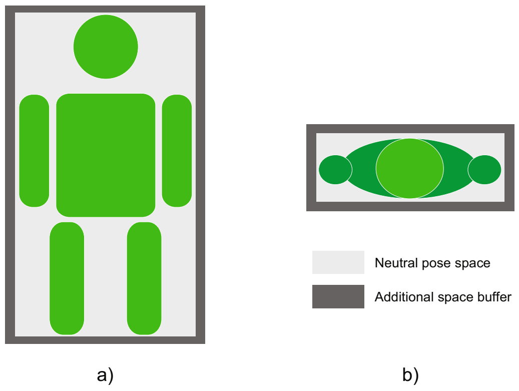

Fig. 2. Schematic frontal (a) and top view (b) of operator within the three-dimensional neutral pose space

2.2 Classifying Gestures

In gesture recognition systems the separation signals play an important function for the routing of the internal processing flow. The command components can be grouped in three classes: *neutral pose*, *static gestures* and *dynamic gestures*. The detection of each gesture class is subject to different processing flows that produce distinct computational expenses.

For instance, to detect a neutral pose, it takes here the lowest computational cost, as the processing flow only needs to find the position of the operator and to check, whether all of his body parts are inside the created neutral pose space. Static gestures like pointing include more computational load, since the appropriate body part (left arm, right arm) has to be found and its pointing direction needs to be estimated. Detecting a dynamic gesture demands the highest computational costs in this context. In addition to the processing steps for a static gesture detection, the temporal progression of the moving body part has to be considered and analyzed as well. For that purpose the following adaptive gestural command aggregation approach is proposed. It consists of different processing modules that enable a variable data processing based on the demand for gestural command components:

2D Person Tracker
This module detects a person in the raw two-dimensional input stream and delivers the position of the operator as a region of interest (ROI) for the subsequent blocks.

3D Body Tracker
This module utilizes the ROI from the 2D person tracker to detect the operator's body in the depth data stream and to create a body model and to perform a position estimation for the body parts, e.g. head, feet and arms.

Gesture Separator
The gesture separator is the first analysis block that can detect a neutral pose and inform the downstream modules about a gesture change, for instance a change from a gesture to a neutral pose and vice versa.

Processing Flow Composer

The processing flow composer is the control center of the gesture recognition system. It sets up the demand for one or more specific gestural command components and their optional dependencies. The basis for the decision-making is the received input from the gesture separator (*gesture change*) and the feedback from the task validator (*missing information*). The generated demand is then send to the gesture type selector.

Gesture Type Selector

Based on the demand from the processing flow composer this module starts the appropriate processing flow to gather the requested gestural command component.

Task Validator

The task validator receives and validates all detected and recognized gestural command components. It has to check all inputs for plausibility and feasibility. If command components are missing, this module gives feedback to the processing flow composer.

This setup enables an adaptive demand-driven and processing flow to contemplate only those processing steps that are needed for the information aggregation. A system overview of all proposed modules and their connections is illustrated in Fig. 3.

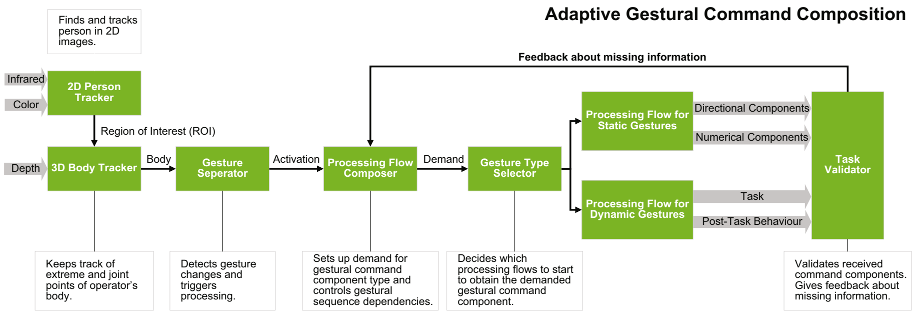
Adaptive Gestural Command Composition
Finds and tracks person in 2D images.
Feedback about missing information
Infrared
Color
2D Person Tracker
Region of Interest (ROI)
Depth
3D Body Tracker
Body
Gesture Seperator
Activation
Processing Flow Composer
Demand
Gesture Type Selector
Processing Flow for Static Gestures
Directional Components
Numerical Components
Processing Flow for Dynamic Gestures
Task
Post-Task Behaviour
Task Validator
Keeps track of extreme and joint points of operator's body.
Detects gesture changes and triggers processing.
Sets up demand for gestural command component type and controls gestural sequence dependencies.
Decides which processing flows to start to obtain the demanded gestural command component.
Validates received command components. Gives feedback about missing information.

Fig. 3. Overview of the adaptive gestural command aggregation approach

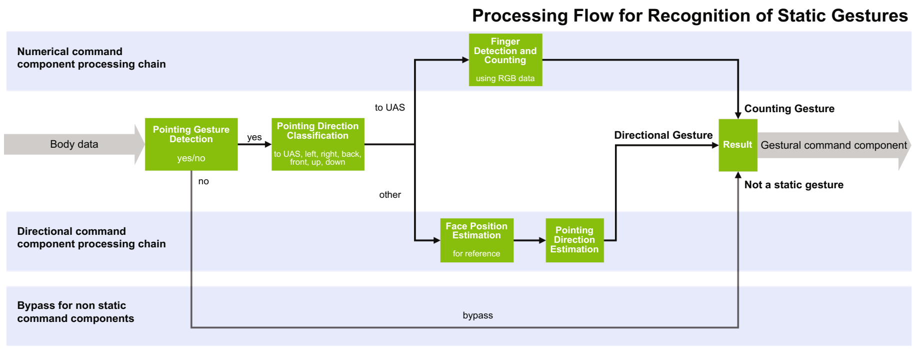
Processing Flow for Recognition of Static Gestures
Numerical command component processing chain
Finger Detection and Counting
using RGB data
to UAS
Counting Gesture
Body data
Pointing Gesture Detection
yes/no
yes
Pointing Direction Classification
to UAS, left, right, back, front, up, down
Directional Gesture
Result
Gestural command component
no
other
Not a static gesture
Directional command component processing chain
Face Position Estimation
for reference
Pointing Direction Estimation
Bypass for non static command components
bypass

Fig. 4. Exemplary overview of the processing flow for the detection of static gestures

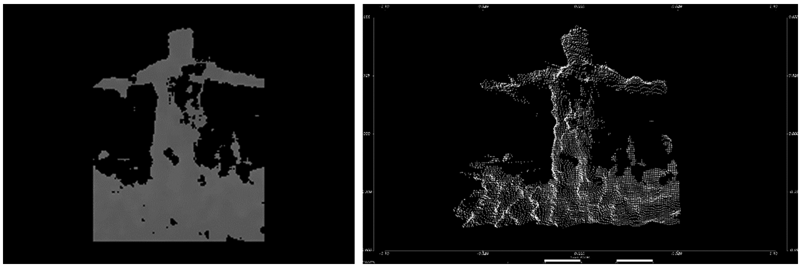

Fig. 5. Two representations of the depth data in comparison: 2D-matrix (left) and point cloud (right)

Exemplary Processing Flow for Static Gestures
The demand-driven principle of the approach from the previous chapter can be applied to both of the processing flows for static and dynamic gestures as well. For example, if the operator performs a pointing gesture ("*Go east*") followed by a counting gesture ("*500 m*"), the directional command component flow (Fig. 4) can be excluded since the directional information is already transmitted and the costly face position estimation can be excluded, which is needed for reference.

3 Experimental Validation

The following chapter covers the first module implementation of the proposed approach for a discrimination between static and dynamic gestures from an airborne platform and its validation on real life data.

3.1 Implementation

The *Intel RealSense R200* camera was selected for the data acquisition. Its advantage is the multisensory all-in-one solution that features besides a high resolution color sensor also two infrared sensitive sensors in a stereoscopic setup to generate its depth data making it suitable for outdoor applications. The gesture recognition in interaction mode relies predominantly on the data of one of the two infrared sensors and the depth data of the mentioned camera. This data can be represented in two ways. On one side it can be seen as a grayscale image where the depth information is coded into the intensity value of each pixel. This so called 2.5D representation allows the integration of the data in existing image processing algorithms.

On the other side its data can also be processed in a three dimensional representation. Using the intrinsic and extrinsic camera parameters of the camera, the depth data can be transformed into a world coordinate system creating a specific position for each point of the image in 3D space. The result of this transformation is known as *point cloud.* Both representation forms are shown in Fig. 5 for comparison.

The second variant is computationally more expensive, but enables elaborated possibilities for gesture recognition. For this the point cloud representation was favored first.

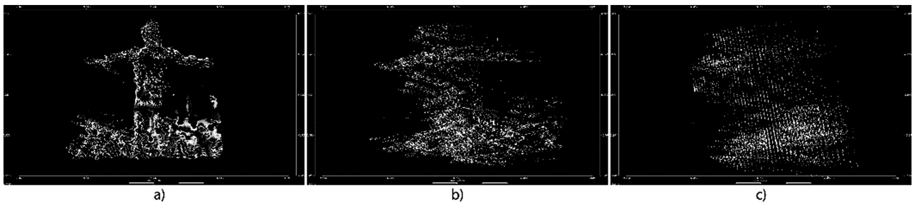

Fig. 6. Operator in point cloud representation from three viewpoints: (a) front, (b) front right, (c) right

However, later experiments have revealed the sensors maximum depth range of about 10 m, but also its rapidly decreasing depth accuracy on that far end of the operational area. The integration of computationally highly expensive multi-stage filtering methods in the implementation would be necessary and therefore inhibits a real time execution. Figure 6 illustrates the noisy depth data from three different viewpoints. The image on the right side (Fig. 6c) shows the prominent quantization artefacts from the stereoscopic approach that manifest as thin plates along the z-axis.

For that reason the 2.5D representation has been chosen for the prototypical gesture recognition implementation.

2D Person Detection and Tracking

The implemented method for person detection and tracking adapted from previous work as describe in [9], namely a combination of a HOG based object detector [10] and a modified correlation tracker [11]. This applies for the utilized UAV and deployed sensor of the experimental setup as well (Fig. 7).

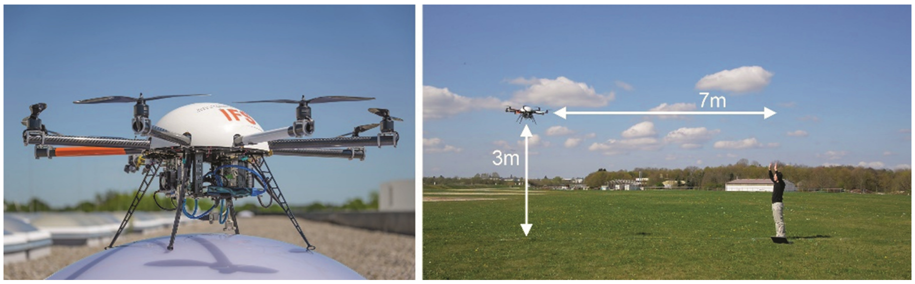

Fig. 7. Utilized UAV with multisensory camera system and experimental setup in interaction mode

Body Part Estimation

Based on the result of the upstream 2D person detector and tracker, the region of interest is first enlarged to fit all body parts within it (the HOG detector is learned only on the silhouette of persons) and then extracted from the depth stream (Fig. 8a, b). The lower part is removed in the next step, which includes noisy and interfering depth data from the ground (Fig. 8c). After that, the background is removed as well, based on the median distance of the operator to the camera (Fig. 8d). A conversion to a lower bitrate followed by a binarization step creates then a suitable base for a subsequent contour analysis (Fig. 8e).

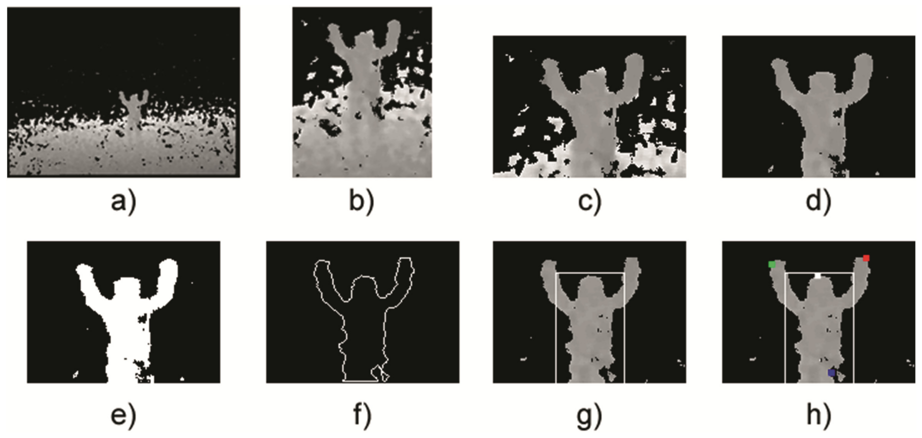

Fig. 8. Involved processing steps for body part estimation (see text for detailed description), (h) shows the detected extreme points and the related body parts: green = left arm, red = right arm, blue = legs, white = head (Color figure online)

This process consist of a blob and contour detection that looks for large connected objects of a given minimum and maximum size and perimeter. Empty regions with no depth data inside the body shape (black spots on the operator's right side in Fig. 8a–e) can be challenging for this processing step as they sometimes divide body parts from the corpus and lead to incorrect detections of the arms, for instance. But this hurdle can be overcome by a preceding blob group analysis and connection.

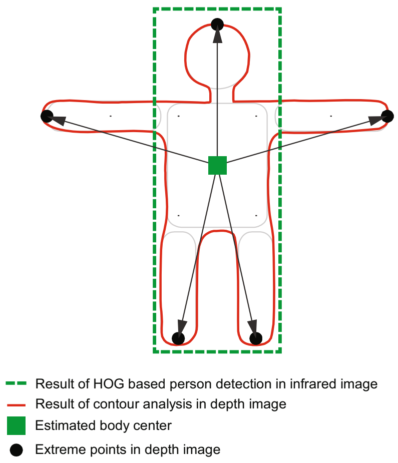
Result of HOG based person detection in infrared image
Result of contour analysis in depth image
Estimated body center
Extreme points in depth image

Fig. 9. Schematic illustration for the estimation of extreme points

The found contours are represented as a list of connected 3D points (2D position plus distance from the camera) (Fig. 8f). The 2D information is then used to find and mark those candidates that show the largest distance to the estimated body center, called extreme points (see Fig. 9). A following plausibility check (feet are under the body center, head is over it, etc.) assigns then each candidate the appropriate body part (Fig. 8h). These found positions are passed through a Kalman-Filter to smooth outliers from the detection caused by noisy depth data. The white rectangle in Fig. 8g–h represents the calculated neutral pose space for the following neutral pose detection.

3.2 Classification

Once the position of the extremities is found, the movements of the person can be already grouped in three classes: *static gestures, dynamic gestures* and a *neutral pose*. Static gestures include low or no fluctuations of the body position data whereas dynamic gestures involve more body movements.

Here a neutral pose is recognized, when the operator keeps his arms down and close to his body and therefore leaving them inside the neutral pose space. Every time he raises his arms out of this space, a new gesture separation signal is send to the system that triggers the following processing steps to look for static and dynamic gestures.

The position of each detected body part is stored in a vector of a defined length, in this case 30 elements. A static gesture signal is triggered if two conditions are true:

1. All positions within the vector are valid (no missed detections) and
2. 50% of all positions are within a defined tolerance area

If both criteria are met, the pointing direction in reference to the head position is then estimated and a (static) pointing gesture signal is emitted.

Classification of Dynamic Gestures

To perform a classification of dynamic gestures, more processing steps are necessary. To take in account the temporal progression of the extreme points, their lateral change can be represented as a two dimensional trajectory (Fig. 10).

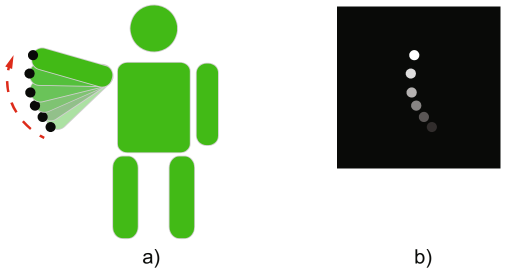

Fig. 10. Schematic figure of a waving operator (a) with the corresponding motion history image (b)

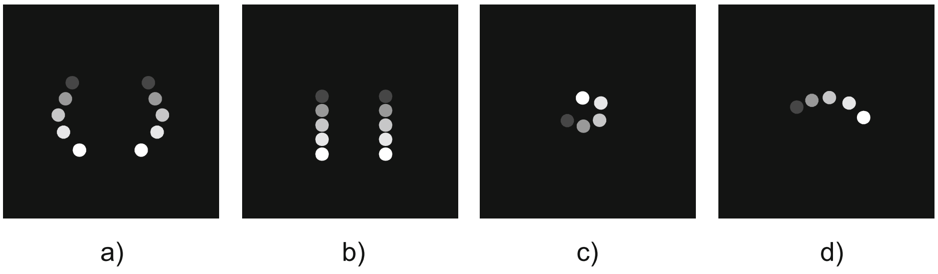

Fig. 11. Gestures represented as motion history images: (a) waving with both arms sideways, (b) movement from top to bottom with both arms, (c) circling with one arm above the head, d) waving with one arm above the head

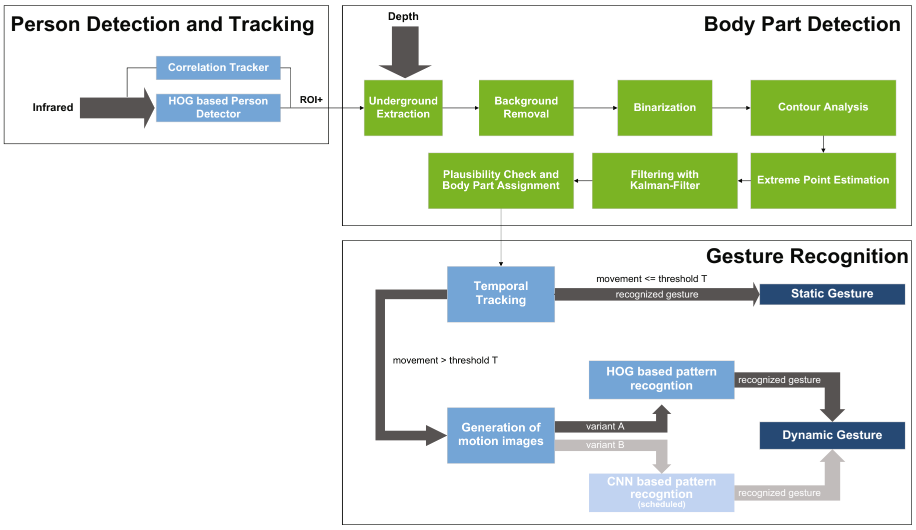

Fig. 12. Overall processing flow for the prototypical gesture recognition implementation

If we plot every position of an extreme point over time as an intensity value into a separate image, we can use the result as an input for a learned image based object detector to assign a motion image to a specific dynamic gesture. These resulting images are named here as motion history images (Fig. 11).

Besides the already used HOG detector for the person detection, a trained convolutional neural network (CNN) [12, 13] would qualify for this task as well. As the expected image size of the motion history images is rather small (not more than 100 by 100 pixels) the necessary number of feature layers should be relatively low and hence computationally cheap. But this point hast to be investigated in the next works. An overview for the complete image processing flow with implemented and scheduled steps is shown in Fig. 12.

3.3 Results

The implementations has been executed on a mainstream Intel i7 powered system without GPU acceleration and heavy software optimization using the recorded on board data from the real flight experiment. Despite the recorded frame rate being 60 Hz, the processing rate dropped to only about 35 Hz, still qualifying as real time. One reason is the early extraction of the ROI hence reducing the effective image dimension to about 155 by 115 pixel for the processing and therefore reducing the computational load. Furthermore the integrated high resolution color stream has not been part of the processing flow at this stage.

The deployed method to detect and track the operator within the infrared image stream performed well without losses through the whole video sequence (*Operator Sequence 1*, frames 539 – 1854). The neutral pose could be detected best (true positive rate TPR = 91.8%) followed by the detection of static gestures (TPR = 79.5%). Dynamic gestures could be detected in most cases by a motion analysis with a TPR of 70.5%. Only the recognition of dynamic gestures using a learned HOG detector on the motion images did not work as expected and showed poor detection rates. One reason might have been the very low effective resolution of 30 by 30 pixel of the used motion images for learning of the detector. A scaling and interpolation of the included body part positions might improve the detection rate. A temporal comparison of the performed gestures featuring the ground truth and the detections is shown in Fig. 13.

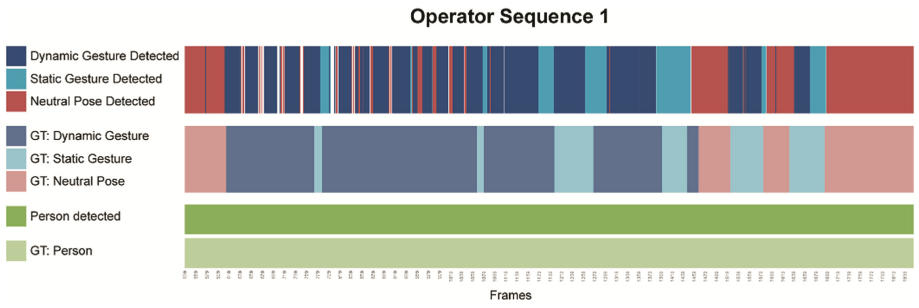

Fig. 13. Overview of the detection results of the recorded video sequence including the ground truth (GT)

4 Conclusion

This work proposed a method to collect gestural command components for the commanding of airborne UAS. An adaptive command component composition approach has been presented that is capable of distinguishing between different gesture types as well as an exemplary processing flow for the demand-driven information gathering of static 2D gestures. Lastly the gesture separation part of the proposed method has been evaluated on real life data from an airborne platform with a prototypical implementation. The used multisensory camera system performed well in the infrared domain but showed deficiencies in the depth stream at the given distances. Other sensor systems have to be considers in the next development steps. Future work will gradually implement all proposed concept parts into the system to establish an operational prototype for a complete gesture based UAS tasking.

References

1. Escalera, S., Athitsos, V., Guyon, I.: Challenges in multimodal gesture recognition. J. Mach. Learn. Res. **17**, 1–54 (2016)
2. Cicirelli, G., Attolico, C., Guaragnella, C., D'Orazio, T.: A Kinect-based gesture recognition approach for a natural human robot interface. Int. J. Adv. Rob. Syst. **12** (2015). doi: 10.5772/59974
3. Yu, T.H., Kim, T.K., Cipolla, R.: Unconstrained monocular 3D human pose estimation by action detection and cross-modality regression forest. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3642–3649 (2013)
4. Bünger, M.: Evaluation of Skeleton Trackers and Gesture Recognition for Human-robot Interaction. Master thesis, Aalborg University, Aalborg (2013)
5. Schwarz, L.A., Mkhitaryan, A., Mateus, D., Navab, N.: Human skeleton tracking from depth data using geodesic distances and optical flow. Image Vis. Comput. **30**(3), 217–226 (2012)
6. Obaid, M., Kistler, F., Kasparavičiūtė, G., Yantaç, A.E., Fjeld, M.: How would you gesture navigate a drone? a user-centered approach to control a drone. In: Proceedings of the 20th International Academic Mindtrek Conference, pp. 113–121. ACM (2016)
7. Peshkova, E., Hitz, M., Ahlström, D.: Exploring user-defined gestures and voice commands to control an unmanned aerial vehicle. In: Poppe, R., Meyer, J.-J., Veltkamp, R., Dastani, M. (eds.) INTETAIN 2016 2016. LNICSSITE, vol. 178, pp. 47–62. Springer, Cham (2017). doi: 10.1007/978-3-319-49616-0_5
8. Cauchard, J.R., Zhai, K.Y., Landay, J.A.: Drone & me: an exploration into natural human-drone interaction. In: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pp. 361–365. ACM (2015)
9. Schelle, A., Stütz, P.: Modelling visual communication with UAS. In: Hodicky, J. (ed.) MESAS 2016. LNCS, vol. 9991, pp. 81–98. Springer, Cham (2016). doi:10.1007/978-3-319-47605-6_7
10. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, pp. 886–893 (2005)
11. Danelljan, M., Häger, G., Shahbaz Khan, F., Felsberg, M.: Accurate scale estimation for robust visual tracking. In: British Machine Vision Conference, Nottingham, 1–5 September 2014
12. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: a convolutional neural-network approach. IEEE Trans. Neural Networks **8**(1), 98–113 (1997)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)