Gestural Transmission of Tasking Information to an Airborne UAV

Alexander Schelle(✉) and Peter Stütz

University of the Bundeswehr Munich, 85577 Neubiberg, Germany
{alexander.schelle,peter.stuetz}@unibw.de

Abstract. A system is presented that enables an authorized person on ground to transmit mission information to an airborne UAV within line of sight by using gestural expressions of both arms without the need for additional devices on ground. A miniaturized processing board with a discrete GPU is used to detect the body movements via a high resolution onboard camera and to translate them into relevant tasking information. Individual task elements are transmitted consecutively, including numerical and non-numerical information. A context aware gesture recognition approach is implemented to enable the reuse of gestures for different contexts in order to maintain a small gesture set. The system further features a bidirectional communication which allows to dispatch visual feedbacks and to query missing information visually via a LED matrix. Two experiments with different briefing contents in a static and dynamic setup have been conducted to proof the feasibility under real-life conditions.

Keywords: Visual communication · Gesture recognition
Human-UAV-interaction

1 Motivation

Unmanned aerial vehicles (UAV) that are being used for image intelligence purposes, receive their command and guidance information primarily via radio link from a ground control station or handheld devices. Without an adequate device, no access to the UAV is possible. In extraordinary and dangerous situations, like infantryman in unknown terrain or search and rescue personnel in disaster scenarios, this requirement is a disadvantage, since there is no option to transfer the authority of the UAV to a third party via alternative channels these days. New ways of interaction are starting to evolve by using the onboard sensors in combination with gesture recognition to allow a visual communication [1–3]. However, current solutions only utilize this visual channel for low level commands, such as triggering an image capture [4] or telling the UAV to move to a specific direction [5]. High level, mission briefing like communication under real-life conditions has not been demonstrated before. This paper covers a system to address this capability gap.

S. Yamamoto and H. Mori (Eds.): HIMI 2018, LNCS 10904, pp. 318–335, 2018.
https://doi.org/10.1007/978-3-319-92043-6_27

2 System Architecture

The proposed system architecture is designed to allow an authorized person on ground to communicate with an airborne UAV within line of sight by performing a set of gestural expressions to transmit mission relevant details. The UAV can sense the operator's movements and translate them into the needed information components to compose the mission objective on board in real-time. To fulfil this goal, the following requirements have to be met by the system:

- The system shall be robust to a dynamic surrounding, since the flying platform will not be steady at one place at all times.
- The hardware and software components must not exceed the payload weight limitations, however ensuring that the onboard processing power is sufficient for real-time applications.
- The deployed sensor shall be passive with a wide operational range to allow interaction at different distance levels.
- A prompt feedback mechanism shall be included to enable a bidirectional communication between the UAV and the operator. Response time shall not exceed 3 s.

The next chapters cover the relevant system components to satisfy these requirements.

2.1 Communication Model

To map the ongoing processes during the human-to-UAV interaction a communication model has to be designed. To understand the employed model, a few definitions have to be made first.

Operator. A person that is instructed and authorized to send mission details to the UAV is called in the following the *operator*.

Mission. A *mission* is a concatenation of single tasks that are processed by the UAV consecutively to achieve a specific goal (reconnaissance, transport, etc.). These tasks are transferred to the UAV during a *mission briefing*. In the current implementation a briefing starts with a conversation start command (in this case a "Hello" gesture) and always ends with the transmission of a "Return to" task as shown in Fig. 1.

Task. A task is a part of a mission that triggers the activation of a subsystem of the UAV under a specific condition. A "find humans" task for instance activates the onboard sensor system and image processing algorithm to detect persons on the ground once the UAV reaches a defined location. That task consists of two information elements: the command ("find") and an additional non-numerical information "humans". Some tasks can may also require additional numerical information (e.g. "maximum mission time"), where others do not require any additional information at all (e.g. "take off now").

Context. The context is the scope of an interaction. Together with an observation and the already received information this creates a meaning for an observation. If a command requires additional information, it defines the next context for that type of

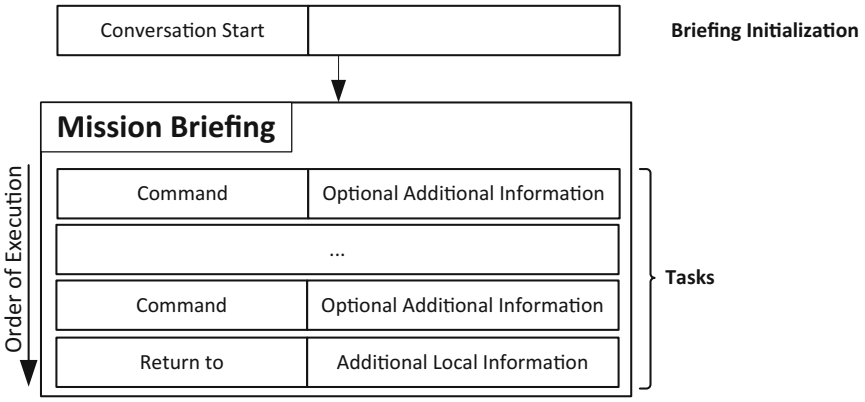

Fig. 1. Layout of a mission briefing

information. Possible contexts are for instance “direction”, “location”, “numerical information”, “tasking” or even “none”.

Meaning. A meaning is the interpretation of an observation for a given context. Since the capability for interpretations is ascribed commonly only to humans, this process can be described here as a translation of an observation to a meaning using multiple parameters for the presented system. Meanings can be for example “start conversation”, “location of operator” or “count fingers of left hand”.

Observation. An observation is the movement or pose of the operator sensed by the UAV. It is the interface between the human operator and the airborne system.

Figure 2 shows the communication model of the proposed system. The operator on ground knows about the type of mission (reconnaissance mission, transport mission, etc.) and the mission specific details. To be able to communicate these details to the UAV, he has to split the mission into multiple single tasks in a way, to represent each task by a set of gestural expression.

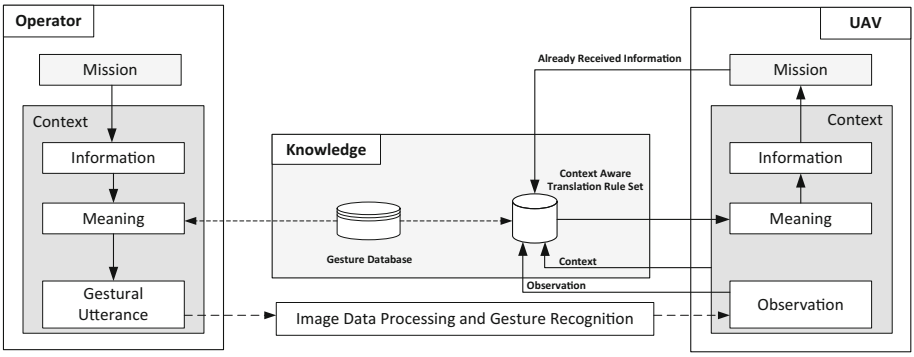

Fig. 2. Communication model

The actual display of the gesture is the gestural utterance, which is visible to the unmanned system and hence can be observed. Both the operator and the gesture recognition system on board the UAV are aware of the possible gestural expressions ("arm up", "arm down", "arm pointing to ground", etc.). This knowledge is implemented in the *Gesture Database* in Fig. 2. Such gesture templates however do not contain any meaning in the first place, instead they can be seen as a means of transportation for the information itself.

The meaning of an observed gesture is assigned in a subsequent step, using the *Context Aware Translation Rule Set*. It takes the current *observation*, *context*, the *gestural expression* itself as well as the already *received information* as an input and reasons for the *meaning* of the gestural expression under the given circumstances. This way, the same gestural expression can be used for different contexts and the amount of gestures to be memorized by the operator can be reduced.

Once the information is physically sent to the UAV via a gestural utterance, the *image data processing and gesture recognition* onboard transfers these body movements into *observations* which are then translated into *meanings*. If the system detects discrepancies (e.g. expects additional information, but receives instead a different task) it informs the operator about this conflict and asks (again) for the needed information. Once the system has received a valid and complete set of tasks for a mission, it finishes the conversation and starts its mission. Figure 3 illustrates this information transfer process from the operator to the UAV. A programmatic solution for the information processing on UAV side is described in Fig. 15 in the Appendix.

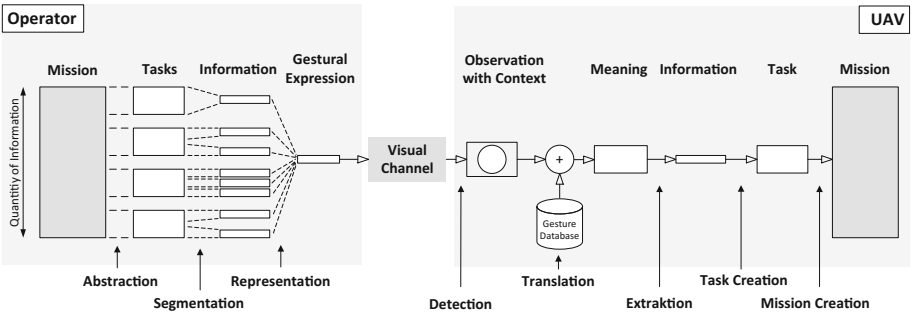

Fig. 3. Information transfer from operator to UAV

2.2 Pose Estimation

A key element for the transmission of gestural information is the detection and localization of the operator's body parts. This process is known as *pose estimation*. In a previous study [6] a depth sensor based approach has been considered to separate the operator from the background reliably and to find extreme points in the depth data to detect the operator's limbs. The results show the advantages of the additional distance information provided by the sensor technology, but also its range limitations, which force the operator to approach the airborne system to an uncomfortable close distance to get within sensor range.

Recent advances in this research domain have brought up a versatile real-time solution [7–9] for this problem, using commonly available 2D image sensors and avoiding the limitations of depth sensors. The open source framework *OpenPose* [10] offers that functionality and supports multiple models with different accuracies and computational demands. However, due to the high degree of parallelism used in that method, a graphics processing unit (GPU) supporting CUDA [11] is required to achieve a real-time processing. Essentially the selected model and the network size determine the performance. The proposed system utilizes the model learned on the MPI dataset [12] using a reduced network size of 288 × 144 pixel.

2.3 Authentication

As pictured in the introduction of the system architecture, accessing the capabilities of the UAV is only allowed for an authorized person. To include a rudimentary authorization mechanism, a color based approach has been selected for this system. The person that wears a high-visibility vest, like the ones used at construction sites, is assumed to be the operator.

To find the operator in a group of persons, the color of the clothing of each detected person is measured at three points across its upper body part (see Fig. 4). To be robust to changes in illumination each color measurement is converted into the HSV (hue, saturation, value) color space and compared to a reference hue value. If all three measurements are similar to the reference, the person is assumed to be an operator.

More elaborated methods that utilize biometrics for the authorization via face [13], skeleton, body shape [14] or gait recognition [15] can be used for that purpose as well. However, tests with an implementation of a deep metric learning based face recognition [16, 17] showed, that the overall system performance decreases by a factor of about 20% for the chosen system hardware configuration. Furthermore, a face recognition demands high requirements on image quality and resolution which cannot be met in all use cases (especially for long-distance interactions, where the optical magnification is not sufficient to maintain a minimum resolution of the operator's face). Therefore, the computationally lightweight and robust solution using special clothing for authorization has been chosen.

2.4 Gesture Recognition

The pose estimation delivers the detected 2D joint coordinates (Fig. 4) including a confidence value. The angles spanned by the joints *hip-shoulder-elbow* and *shoulder-elbow-wrist* are determined and used as features for each body side. The gesture recognition is divided into two processing chains, a static recognition for pointing and holding type poses and a dynamic recognition for poses that change over time.

Static Gesture Recognition. To detect static gestures, a feature comparison approach is applied. The advantage of this method is the low computational cost (since only two features have to be compared with a reference) and the avoidance of a learning phase, meaning that only reference features have to be defined. Figure 5 visualizes the realized

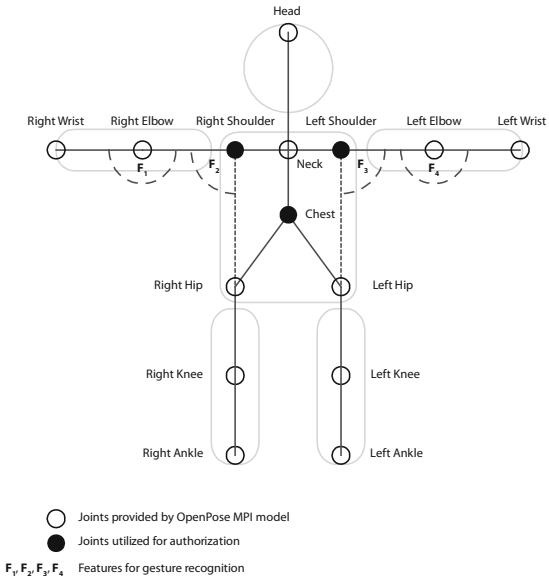

Fig. 4. Provided joints by *OpenPose* and their utilization in the system

gesture set. The inner circle represents the commands that can be followed by one or more additional numerical or non-numerical information, displayed in the second and third ring.

Dynamic Gesture Recognition. To detect dynamic gestures, features have to be analyzed over time. A commonly used and proven approach for this is *Dynamic Time Warping* [18]. This method is suitable for real-time processing and is included in the *Gesture Recognition Toolkit* [19]. Preliminary studies have shown that the achieved frame rate of about 4.5 frames per second (fps) however is not sufficient for the detection of dynamic gestures performed at a common pace (i.e. the operator would need to wave unnaturally slow to get a reliable detection of a "hello" gesture). For that reason, dynamic gestures are not considered in the current state of system implementation. However, dynamic gestures shall be included in the next development iteration of the system once the processing rate can be raised to 10+ fps.

2.5 Finger Detection

For the transmission of numerical information, showing and counting of fingers is a common communication method for humans. Therefore, a hand and finger detection has been included in the system. The implemented method is a modified version of [20], which uses hull curve and convexity defects of a shape to find finger tips in it (see Fig. 6). The location of the hands is estimated based on the wrist information of the pose estimation and the segmentation is performed via skin color thresholding [21].

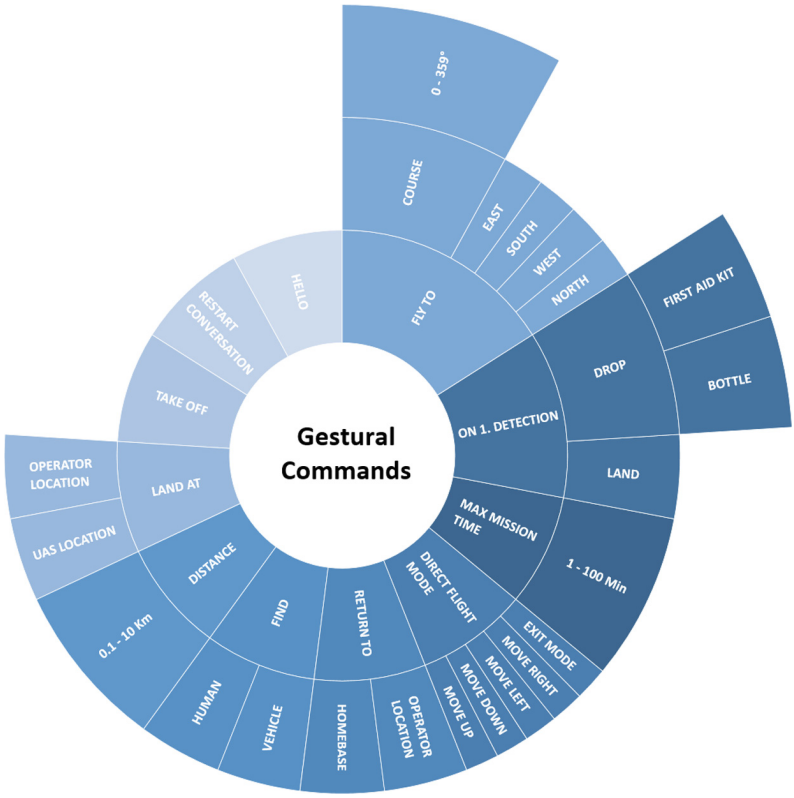

Fig. 5. Supported gestural tasking commands and the task dependent additional information

2.6 Feedback

To provide a feedback mechanism for bidirectional communication, a visual approach has been chosen using a bright LED matrix. To give the operator an acknowledgement for a recognized command, the translated gestural observation is displayed for two seconds followed by a prompt in case additional information is needed or the tasking sequence is not completed. Due to the limited available space for payload, the selected matrix with a width and height of 32 × 8 pixels can display only five letters at a time, hence limiting the allowed word length. Scrolling text horizontally had been tested beforehand and showed multiple disadvantages. Waiting for the information to scroll through extends the communication process. More importantly, it increases the risk for the operator to overlook important parts of the message, if he turns his eyes away from the UAV during the feedback phase. To enhance the readability further, common aeronautical abbreviations (e.g. “CRS” for course), interrogatives or combinations of the needed information followed by a question mark are used (Fig. 7). Experiments under direct sunlight have proven a good readability using green LEDs for distances of up to 75 m with normal vision.

Fig. 6. Visualization of the onboard gesture recognition and finger detection algorithm, from top to bottom line: 1. CNTLR = Count fingers on left and right side, 2. movement status, 3. current context, 4. Number of detected fingers and confidence for each hand, 5. S = detected observation of static gesture recognition, D = result of dynamic gesture recognition (here none, since operator is not moving), yellow circles = detected finger tips (Color figure online)

Fig. 7. Multicopter UAV with stabilized sensor system, processing board and LED matrix for visual feedback

2.7 Hardware Design

A flying platform[1] has been selected that can carry up to 6 kg of payload. It features an autopilot that supports several automated flight modes, such as hovering, automatic take off/landing and waypoint navigation. However, for the experiments described in this paper the automated flight capabilities of the UAV have not been used. Instead, a safety pilot takes over the task of keeping the UAV in an appropriate distance in front

[1] DJI Matrice 600 Pro.

of the operator at this stage. The payload includes a stabilized gimbal system[2] that carries a GigE Vision camera[3] with a resolution of 1920 × 1080 pixels and a 20x optical magnification. Furthermore the gimbal hosts the main processing unit, which is a miniaturized multicore system featuring a discrete GPU, the LED matrix[4] and a controller board[5] that drives the matrix. All payload components are powered by the gimbal system, except the LED matrix, which receives its power from the multicopter itself due to the high current demands (Fig. 8). A WLAN connection is used only for the startup of the onboard software modules and for debugging purposes.

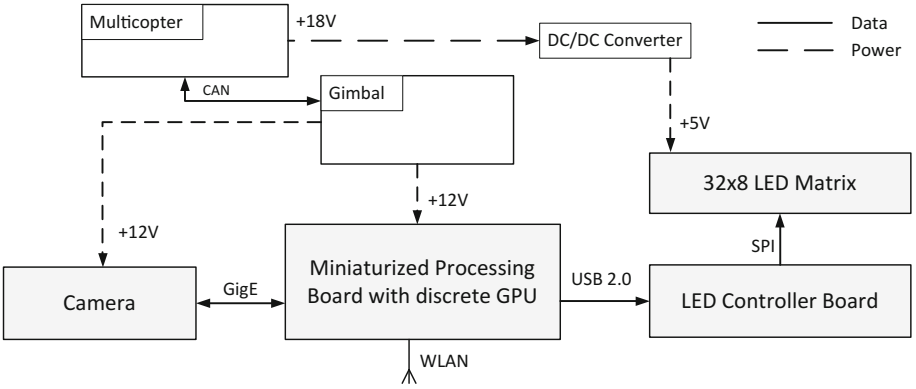

Fig. 8. Involved hardware on board the UAV

2.8 Software Design

The Robot Operating System (ROS) [22] serves as communication infrastructure for all software modules. The software is organized in three major processing nodes (Fig. 9):

Image Acquisition. This node communicates with the GigE Vision camera and provides the received image data to other nodes.

Pose Estimation. This node receives image data and publishes the detected joints to the network. The implementation is a modified version of [23].

VisCom. This is the main node of the visual communication system and is divided into different processing modules. Since the pose estimation node provides its results depending on the order of detection, the *person tracking module* assigns the received skeletal data to each person across frames. The *operator authorization module* determines the authorized person in the image and forwards its joint data to the *gesture recognition module*. This module calculates the features and performs the static and

[2] Nvidia Jetson TX2 with Connect Tech Astro carrier and breakout board.

[3] Sony FCB-EH6300.

[4] Adafruit Flexible 8 × 32 NeoPixel RGB LED Matrix.

[5] Arduino Mega 2560.

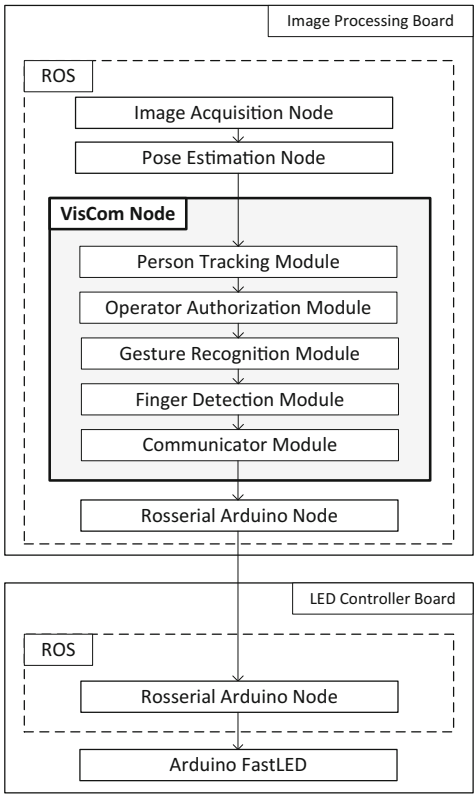

Fig. 9. Software nodes and modules on board the UAV

dynamic gesture recognition. The recognized observations for both body sides are then combined with the results of the *finger detection module* and forwarded to the *communicator module*. This handles the translation of observations, context, available gestures and already received information to a mission relevant meaning. Furthermore, this module controls the visual feedback signals by dispatching display commands to the LED controller board via a *rosserial arduino node*, which acts as an interface between both boards. The actual control of the LED matrix is performed by the *FastLED* library [24].

3 Experiments

Two experiments have been conducted to test the system performance under real life conditions. Multiple missions with different numbers of tasks and objectives were defined for testing with briefing durations from 32 to 209 s. One of them is shown in Table 1. Parts of the communication process are depicted in the image sequences in Figs. 10, 11 and 12.

Table 1. Exemplary mission

Meaning	Context	Observation for arm		Feedback	Question
		Left	Right		
Conversation start	None	Hanging	L-shape up	HELLO	TASK?
Fly	Task	Pointing out	Pointing out	FLY	TO?
Course	Direction	Palm touching other palm, low	Palm touching other palm, low	CRS	C-
1	Numerical information	Hanging	Pointing to chest, 1 finger visible	C 1-	
2	Numerical information	Hanging	Pointing to chest, 2 fingers visible	C 12-	
2	Numerical information	Hanging	Pointing to chest, 2 fingers visible	C 122	DIST?
for 1 km	Numerical information	Hanging	Pointing to chest, 2 fingers visible	1 km	THEN?
Find	Task	Hanging	2 fingers pointing to eyes	FIND	WHAT?
Human	Additional information	Pointing to chest	Pointing to chest	HUMAN	THEN?
On first detection	Task	L-shape up	Hanging	1. DET	DO?
Drop bottle	Additional information	Hanging	L-shape down	DRP B	THEN?
Return to	Task	L-shape down	L-shape down	RET	LOC?
My location	Direction	Hanging	Pointing to ground	URLOC	OK

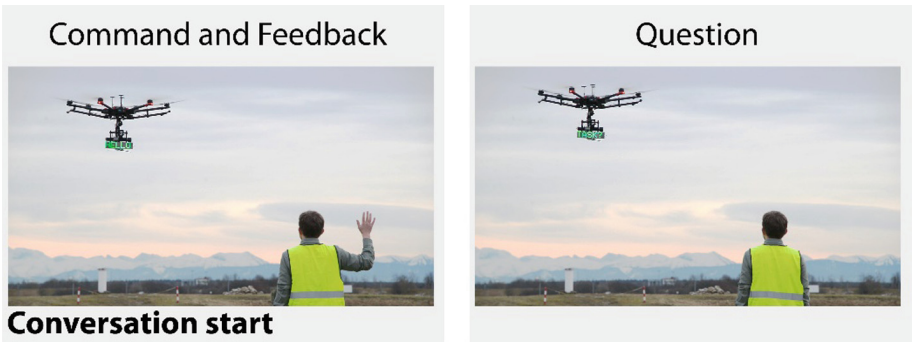

Fig. 10. Operator starting the mission briefing with airborne UAV, UAV asks for a task

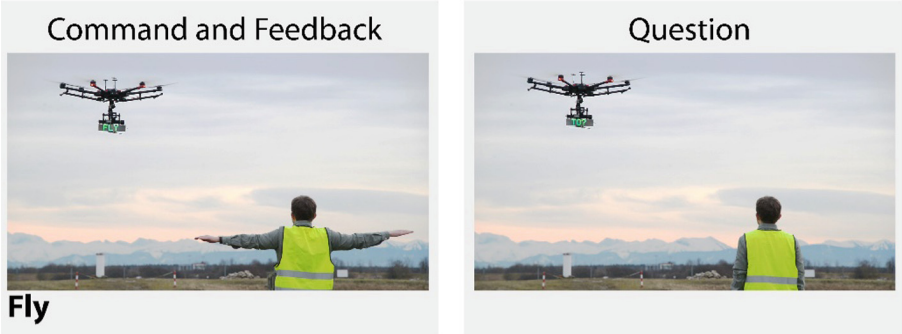

Fig. 11. Operator tasking UAV to fly to a specific direction, UAV responds with question

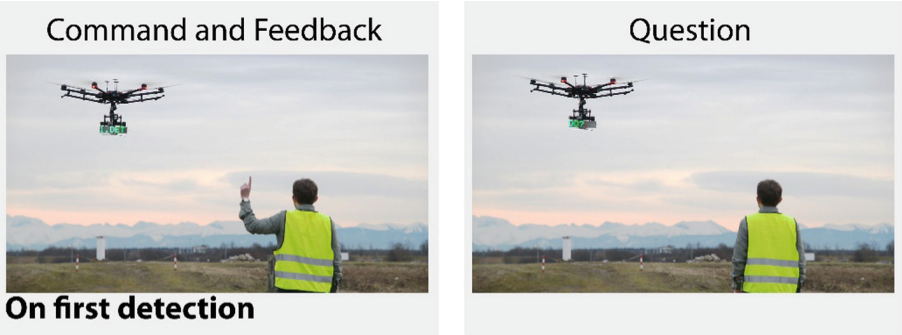

Fig. 12. Operator tasks UAV to perform an action on the first detection of a human, UAV asks for the action

3.1 Static Performance

The first experiment was conducted with a static platform to test the system performance under optimal conditions without lateral and horizontal movements. For that, the payload had been detached from the flying platform and mounted on the roof of a van in a height of about 2.5 m. The operator was standing in a distance of 20 m away from the van and commanded multiple missions with different numbers of tasks.

3.2 Dynamic Performance

The second experiment included the whole UAV with the stabilized system mounted underneath the flying platform and hovering in a height of 2 to 3 m above the ground to evaluate the influence of the platform movement on the detection and recognition capabilities. The distance between the UAV and the operator was about 22 to 25 m. Fluctuations resulted from windy conditions on that day.

3.3 Results

The experimental results were assessed regarding two major aspects: performance of pose estimation and the response time from the beginning of a gestural movement to the feedback display.

Performance of Pose Estimation. Choosing the appropriate network size for the pose estimation is an accuracy-performance tradeoff (see explanation in chapter 2.2). The larger the size is, the higher the detection rate for small objects gets, but at the same time the processing rate drops significantly. On the other hand, decreasing the network size improves the processing rate, but demands bigger objects in the image frame. The chosen size of 288 × 144 pixels requires an operator height of about 85% of the frame height for a reliable pose estimation with a sufficient accuracy for the gesture recognition. This can be achieved by a closer hovering next to the operator (not recommended for safety reasons) or a higher optical magnification, that has been selected here. Due to the inference of the convolutional neural network approach, a measured average processing delay of 0.74 s is involved for every frame for the used processing board at maximum CPU and GPU clock rates.

Response Time. To reduce the false positive rate, most of the implemented algorithms use a circular buffer and a confidence value $c \geq 0.8$. Hence, given a buffer size of $b = 5$ and a mean processing performance of $\bar{f} = 4.5\,\text{Hz}$, a first detection in an ideal use case can be expected after:

$$T = \frac{c \times b}{\bar{f}} = 0.\bar{8}\ s \tag{1}$$

Figures 13 and 14 show the response times for various non-numerical and numerical gestural commands for the static respectively airborne system. All time measurements start in neutral pose, i.e. both arms are hanging, and stop once the LED matrix displays a feedback. So, this measurement includes the transformation from the neutral pose to the final gesture.

The static setup represents the ideal situation with no vehicle movements and an optimal resolution of the operator. Thus, most non-numerical commands are recognized after an average duration of 1.4 s (delay of pose estimation not included), while the numerical commands are decoded with a much greater delay. The reason for this can be found in the implemented color based hand and finger detection, which is sensitive to self-shadowing and skin color-like objects close to the area around both wrists. If these objects occur in that area or shadows cover the hands significantly, the false detection rate tends to rise considerably. The minimum confidence constraint extends then the detection phase and results in a delayed feedback, which misleads the operator to question the correctness of his gestural utterance. For that reason, briefings have been aborted after 15 s of no feedback display.

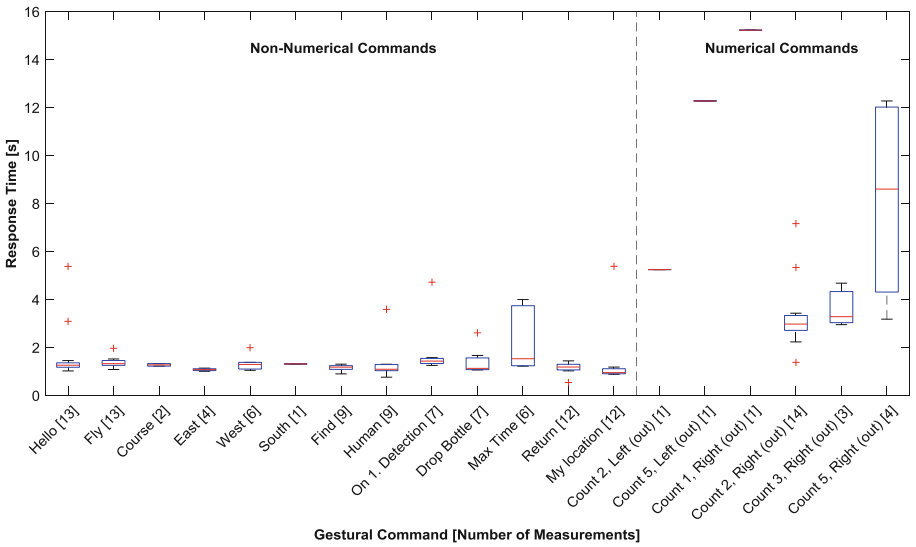

Fig. 13. Response times for gestural commands from first movement to display for static setup, delay of pose estimation not included

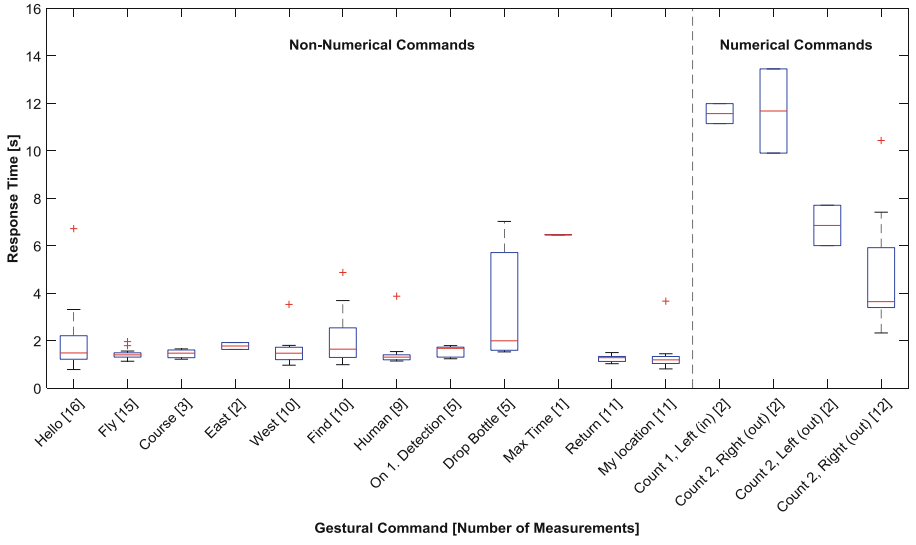

Fig. 14. Response times for gestural commands from first movement to display for airborne setup, delay of pose estimation not included

As expected the airborne setup shows longer average response times. The main reasons are the constant changes in distance to the operator due to manual UAV control and windy conditions, resulting in fluctuations of resolution. Furthermore, the non-optimal positioning of the operator within the frame lead to partial cut offs of the head or the feet of the operator, as the possibilities of the safety pilot to control the multicopter and the gimbal system at the same time were limited. Nevertheless, most non-numerical gestural commands fluctuate around an acceptable response time of 2 s.

4 Conclusion

The presented system allows an authorized person on ground to transmit mission briefing information to an airborne UAV by using his arms and fingers. The developed context aware gesture recognition method delivered promising results in first real-life experiments, helping to keep the false detection rate low and to reduce the amount of gestures to memorize for the operator. Using a bright LED matrix for visual feedback turned out to be a convenient way to improve the communication process and to reduce the uncertainty about the system state. The required robustness against the movement of the flying platform was proven, under the assumption that the operator stays within the image frame. The tradeoff between accuracy and processing rate demands a specific operator resolution. For that reason, the image framing is essential, especially when using optical magnification and operating the UAV in manual mode. These limitations will be taken care of in future steps by handing off the authority for the operator framing to a software tracking module. To improve the recognition of numerical information, more computational power will be added to enable more advanced and light invariant methods for finger detection. Finally, the integration of the autopilot into the system will allow the execution of the commanded tasks from the mission briefing.

Appendix

See Fig. 15.

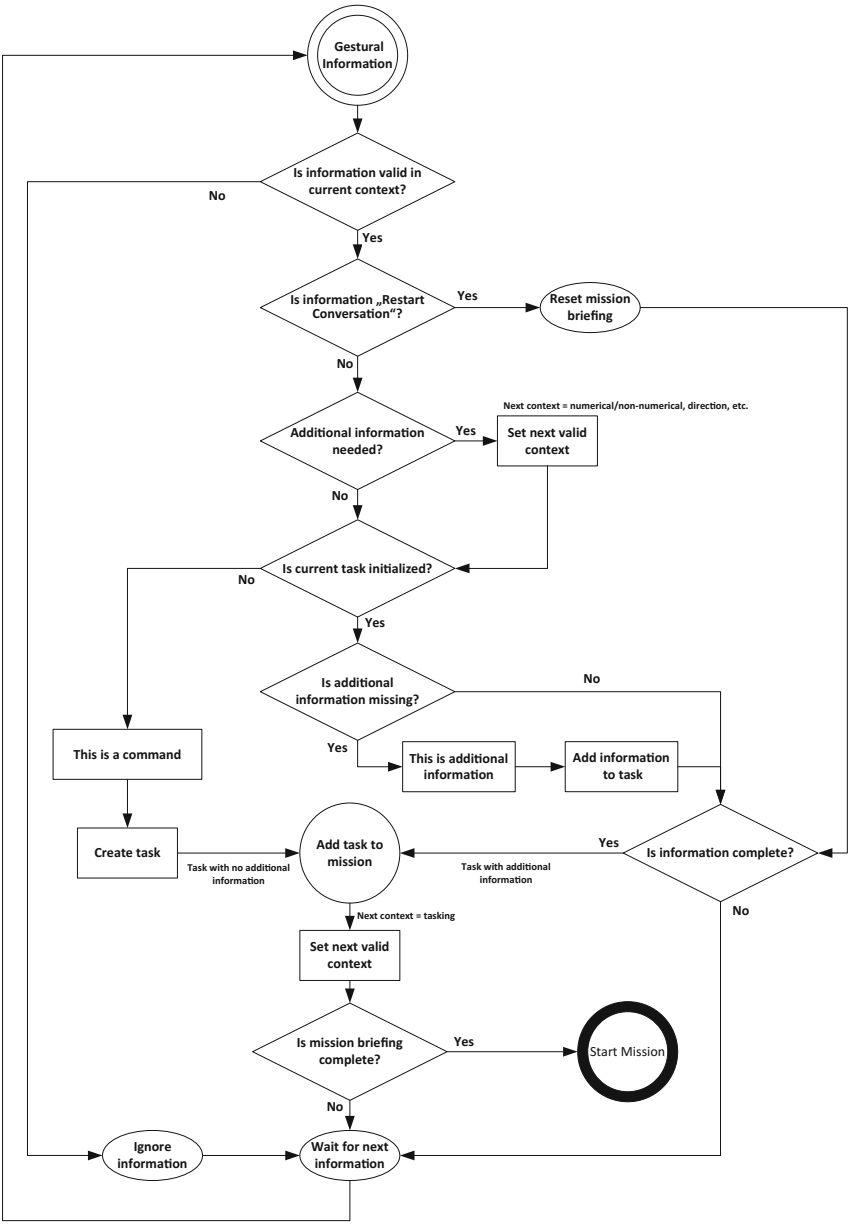

Fig. 15. Decision tree for the information processing on board the UAV

References

1. Schelle, A., Stütz, P.: Modelling visual communication with UAS. In: Hodicky, J. (ed.) MESAS 2016. LNCS, vol. 9991, pp. 81–98. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47605-6_7
2. Monajjemi, M., Mohaimenianpour, S., Vaughan, R.: UAV, come to me. End-to-end, multi-scale situated HRI with an uninstrumented human and a distant UAV. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4410–4417 (2016). https://doi.org/10.1109/iros.2016.7759649
3. Monajjemi, V.M., et al.: HRI in the sky: creating and commanding teams of UAVs with a vision-mediated gestural interface. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 03 November 2013, pp. 617–623. IEEE (2013)
4. Wang, T., Wang, M.: Remote control method and terminal Patent US9493232B2 (2016)
5. Nagi, J., et al.: HRI in the sky: controlling UAVs using face poses and hand gestures. In: HRI, pp. 252–253 (2014)
6. Schelle, A., Stütz, P.: Visual communication with UAS: recognizing gestures from an airborne platform. In: Lackey, S., Chen, J. (eds.) VAMR 2017. LNCS, vol. 10280, pp. 173–184. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57987-0_14
7. Cao, Z., et al.: Realtime multi-person 2D pose estimation using part affinity fields. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
8. Wei, S.-E., et al.: Convolutional pose machines. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4724–4732 (2016)
9. Simon, T., et al.: Hand keypoint detection in single images using multiview bootstrapping. arXiv preprint arXiv:1704.07809 (2017)
10. Cao, Z., Simon, T., Wei, S.-E., Sheikh, Y.: OpenPose: real-time multi-person keypoint detection library for body, face, and hands estimation. Carnegie Mellon University, Perceptual Computing Laboratory (2018). https://github.com/CMU-Perceptual-Computing-Lab/openpose
11. Kirk, D.: NVIDIA CUDA software and GPU parallel computing architecture. In: Proceedings of the 6th International Symposium on Memory management, Montreal, Quebec, Canada, pp. 103–104. ACM, New York (2007). https://doi.org/10.1145/1296907.1296909
12. Andriluka, M., et al.: 2D human pose estimation. new benchmark and state of the art analysis. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3686–3693 (2014). https://doi.org/10.1109/cvpr.2014.471
13. Liu, W., et al.: SphereFace. Deep hypersphere embedding for face recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
14. Godil, A., Grother, P., Ressler, S.: Human identification from body shape. In: Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling, 3DIM 2003, pp. 386–392 (2003)
15. Bouchrika, I.: On using gait biometrics for re-identification in automated visual surveillance. In: Computer Vision: Concepts, Methodologies, Tools, and Applications, pp. 2363–2386. IGI Global (2018)
16. King, D.E.: Dlib-ml: a machine learning toolkit. J. Mach. Learn. Res. **10**, 1755–1758 (2009)
17. Shen, C., Kim, J., Wang, L.: A scalable dual approach to semidefinite metric learning. In: CVPR 2011, Providence, RI, USA, pp. 2601–2608. IEEE (2011)
18. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, vol. 10, pp. 359–370. AAAI Press, Seattle (1994)

19. Gillian, N.E., Paradiso, J.A.: The gesture recognition toolkit. J. Mach. Learn. Res. **15**(1), 3483–3487 (2014)
20. Yeo, H.-S., Lee, B.-G., Lim, H.: Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. Multimed. Tools Appl. (2015). https://doi.org/10.1007/s11042-013-1501-1
21. bin Abdul Rahman, N.A., Wei, K.C., See, J.: RGB-H-CBCR skin colour model for human face detection. Faculty of Information Technology, Multimedia University, vol. 4 (2007)
22. Quigley, M., et al.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
23. Munaro, M., et al.: OpenPTrack: people tracking for heterogeneous networks of color-depth cameras. In: IAS-13 Workshop Proceedings of the 1st International Workshop on 3D Robot Perception with Point Cloud Library, Padova, Italy, pp. 235–247 (2014)
24. Garcia, D., Kriegsman, M.: FastLED (2015). https://github.com/FastLED/FastLED