Ensuring Safety of Machine Learning Components Using Operational Design Domain

Christoph Torens*, Franz Jünger†, Sebastian Schirmer‡, Simon Schopferer§, Dmytro Zhukov¶, Johann C. Dauer‖,
German Aerospace Center (DLR), Institute of Flight Systems, Braunschweig, Germany

The introduction of machine learning in the aviation domain is an ongoing process. This is also true for safety-critical domains, especially for the area of Urban Air Mobility. A significant growth in number of air taxis and an increasing level of autonomy is to be expected allowing for operating a large number of air taxis in complex urban environments. Due to the complexity of the tasks and the environment, key autonomy functions will be realized using machine learning, for example the camera-based detection of objects. However, the safety assurance for avionics systems using machine learning components is challenging. This work investigates safety and verification aspects of machine learning components. A camera-based detection of humans on the ground, e.g. to assess a potential landing area, serves as an example for an machine learning-based autonomy functio and was integrated into an Unmanned Aircraft. In the context of this exemplary machine learning component, the concept of Operational Design Domain as recently adapted European Aviation Safety Agency in the context of machine learning assurance is described along with other key concepts of machine learning assurance. Furthermore, runtime assurance is used to monitor conformance to the Operational Design Domain during flight. The presented flight test results indicate that monitoring the Operational Design Domain can support performance as well as the safety of the operation.

I. Introduction

With recent advances in artificial intelligence (AI) and machine learning (ML) there is an increasing demand for the use of ML, even for safety-critical applications in the aviation domain. Current research is targeted towards the use of ML to reach high degrees of autonomy in the context of Urban Air Mobility (UAM), i.e. transportation services via air taxis in urban environments. However, a key research question is how ML-enabled autonomy can be safely applied in the context of UAM [1]. It is difficult to comply to rigorous safety requirements and the corresponding standards, as established in aviation for safety-critical systems and software. Furthermore, safety standards for ML applications are currently under development and give only first guidelines [2]. In this paper we use key concepts such as Operational Design Domain (ODD) introduced by these guidelines to derive an architecture that ensures proper operational conditions for an exemplary ML algorithm during flight by the use of runtime monitoring. This algorithm is a state-of-the-art object detector for detecting persons on images captured by a Unmanned Aircraft (UA)'s onboard camera. The components of the derived architecture filter inputs to the ML algorithm that are outside of its ODD based on navigation information and image metadata. Results indicate that using such an architecture to detect images outside of the ODD cannot safeguard the ML component. It improves the overall performance, i.e. reduces the number of misdetections, and may therefore contribute to the overall operational safety.

The remainder of this paper is structured as follows: First, Section III derives components of the architecture based on existing safety and trustworthiness guidelines. Then, Section IV presents the setup for the flight test, followed by the presentation of the software setup for the flight test in Section V. Finally, in Section VI results of the flight test are shown.

II. Related work

*Research Associate, Department Unmanned Aircraft, AIAA Senior Member.
†Research Associate, Department Unmanned Aircraft
‡Research Associate, Department Unmanned Aircraft
§Team Lead Safe Autonomy, Department Unmanned Aircraft
¶Research Associate, Department Unmanned Aircraft
‖Department Head, Department Unmanned Aircraft

A. ML Safety Assurance

In the last year, a literature review on the topic of ML safety [3] and an analysis of the existing and new regulation and guidance on ML safety has been published [4] by the authors. This work is a follow-up and extension to that aforementioned research. However, the overall framework and the main context for verification and safety of ML in aviation is set by the following documents. The European Aviation Safety Agency (EASA) published a number of documents regarding the safe use of ML, starting with the AI roadmap [5], and following up with more details on different aspects of ML safety with Concepts of Design Assurance for Neural Networks (CoDANN) [6], as well as CoDANN II [7], and finally a resulting Concept Paper First Usable Guidance for Level 1 Machine Learning Applications [2]. Furthermore, recently a report was published by FAA, Neural Network Based Runway Landing Guidance for General Aviation Autoland [8]. In addition to authorities such as EASA and FAA, also standardization organizations are currently working on standardization of AI in aviation, such as the joint RTCA/EuroCAE working group 114 [9]. The discussion of the safety aspects of the ML component in sections III and VI.B is performed with the background of these aforementioned documents.

Research, specifically regarding the certification of ML in relation to existing traditional standards, such as DO-178C, has also been done recently [10–12].

B. Operational Design Domain

Moreover, there is specifically research regarding the topic of the operational parameters, also called Operational Design Domain (ODD) and the safety a for operation . The concept of ODD was first used in the automotive domain, where a taxonomy for operational design domains was published in 2020 [13]. However, the related research on this concept and specifically in relation to safety [14–17] at this point seems to be exclusively in the automotive domain. There is not much research of the concept of ODD in the aviation domain. This paper is an example of a direct application of the ODD concept in the aviation domain. This concept will be further discussed in Section III and later utilized to check the inputs of the ML algorithm in Section VI.B. The first usage of ODD in the aviation domain was that the definition of ODD is given in the EASA document [2] as follows:

> Operating conditions under which a given AI-based system is specifically designed to function as intended, in line with the defined Concept of Operations (ConOps), including but not limited to environmental, geographical, and/or time-of-day restrictions. In short, the ODD defines the range of operating parameters within which the AI-based system is designed to operate, and as such, will only operate nominally when the parameters described within the ODD are satisfied. The ODD also considers correlations between operating parameters in order to refine the ranges between these parameters when appropriate; in other words, the range(s) for one or several operating parameters could depend on the value or range of another parameter. ([2], page 155)

A similar concept is that of out-of-distribution (OOD). Here the idea is to check if the input data is consistent with the trained data. The concept of OOD is a commonly researched metric for assessing inputs and increase reliability [18]. For the safety of the ML component it is important to supervise both aspects. In fact the EASA guidance document [2] requires with its objectives to supervise both concepts, as discussed in the following sections.

C. Runtime Assurance

Finally, there is research on safety monitoring and runtime assurance, which is an underlying principle, also for verifying the ODD during the operation [19, 20]. In combination of these concepts can be used to safeguard the ML component against unsafe inputs. The idea is to supervise if the input that is fed to the ML component is consistent with the intended operating conditions and limitations. An evaluation of such an ODD monitor is shown in Section VI.B.

III. Machine Learning Trustworthiness and Safety

In the context of a safety-critical domain, it is essential to show that the ML function is safe. The idea is to utilize the principles from the EASA guidance document [2]. This document introduces four building blocks for assuring trustworthiness: trustworthiness analysis, learning assurance, explainability, and safety risk mitigation. With this paper the focus will be on safety risk mitigation and safeguarding the ML component.

A. AI Trustworthiness Analysis

For the ongoing work, a final flight demonstration will show a scenario of UAM in a (simulated) urban environment. A possible use case is the landing of a air taxi at a vertiport. As mentioned earlier, one use case of the detection of humans is to reduce the risk during the landing operation at the vertiport. However, in this case the ML function is utilized as a safety-critical component. EASA states in the building block of AI Trustworthiness Analysis, Safety Assessment (C.2.2) the following Anticipated MOC-SA-02-1:

> In performing the safety assessment, the applicant should address the following aspects:
>
> • ...
> • Analyse and mitigate the effect of AI/ML constituent exposure to input data outside of the ODD;
>
> ([2], page 23)

Furthermore, EASA states Anticipated MOC-SA-04-1:

> In performing the safety support assessment, the applicant should address the following aspects:
>
> • ...
> • analyse and mitigate the effect of AI/ML constituent exposure to input data outside of the ODD;
>
> ([2], page 26)

This means, that the impact of input data outside of the ODD and resulting failures of the ML function on the system must be analyzed. In our context, we would have the following two failures as a possibility. A false positive detection of a human and a false negative detection of a human. A false positive detection would mean that in our use case the system would not be able to land on a vertiport automatically. This would not immediately result in an unsafe situation, since there are multiple contingencies for the situation. Possible solutions would include landing on an alternative landing site, requesting human support from a pilot or remote pilot, waiting in a safe hover position and reevaluating the situation after some time. A false-negative detection would mean that the system would trigger an automated landing, although a human is in the vicinity, this would result in an unsafe situation. Therefore, it is necessary to monitor the input and detect if the data is outside the ODD.

B. Safety Risk Mitigation

As discussed in the last section, it is necessary to monitor the input data and check if the data is outside of the ODD. The corresponding objectives in the air are documented as follows:

> Objective SRM-02: The applicant should establish SRM means as identified in Objective SRM-01.
> Anticipated MOC SRM-02-1: The following means may be used to gain confidence that the residual risk is properly mitigated:
>
> • monitoring of the output of the AI/ML constituent and passivation of the AI-based system with recovery through a traditional backup system (e.g. safety net);
> • when relevant, the possibility may be given to the end user to switch off the AI/ML-based function to avoid being distracted by erroneous outputs.
> • The SRM functions should be evaluated as part of the safety assessment, and, if necessary, appropriate safety requirements should be defined and verified. This may include independence requirements to guarantee an appropriate level of independence of the SRM architectural mitigations from the AI/ML constituent.
>
> ([2], page 73)

With this research, the monitoring is done on additional sensor data, such as altitude, see Section V.E, GPS geofence as well as GPS tunnel, see Section V.D. Furthermore, the image itself is being analyzed, see Section V.C. A possible mitigation in this case would be to invalidate any output from the ML function. The automated landing would be disabled completely and control would be escalated to a human pilot.

C. AI Explainability

The function of monitoring can also support explainability objectives. The following objectives for explainability would be relevant:

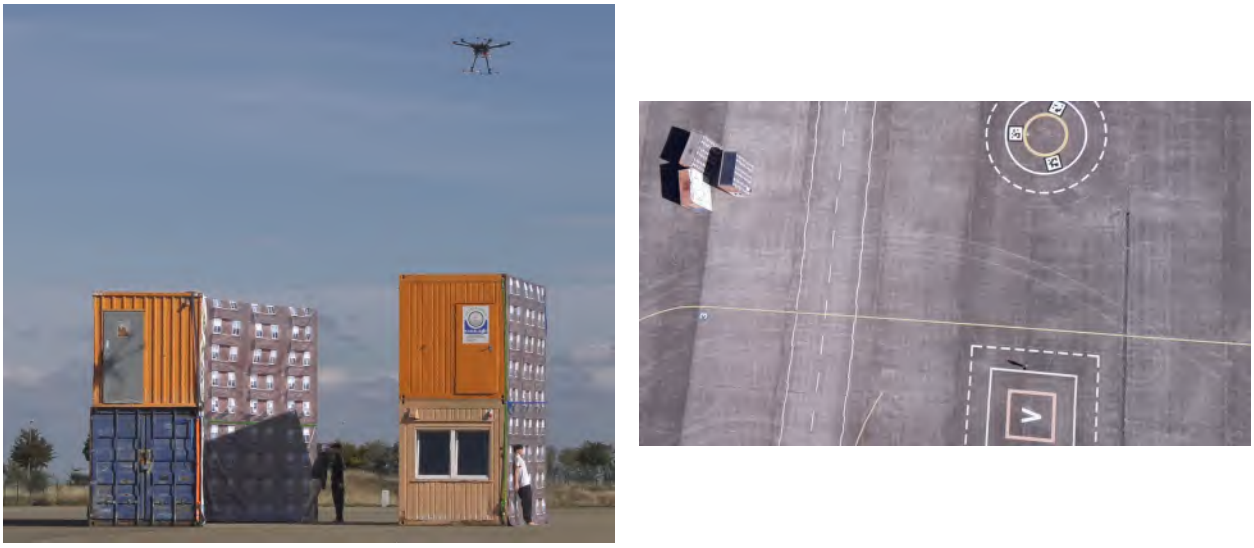

Fig. 1 On the left, the container cluster surrounded with mannequins and UA in mid air. On the right, top view of the vertiports and the container cluster.

> Objective EXP-14: The AI-based system inputs should be monitored to be within the operational boundaries in which the AI/ML constituent performance is guaranteed, and deviations should be indicated to the relevant users and end users.
> Objective EXP-15: The AI-based system outputs should be monitored to be within the specified operational performance boundaries, and deviations should be indicated to the relevant users and end users.
> Objective EXP-16: The training and instructions available for the human should include procedures to act on the possible outputs of the ODD and performance monitoring.
> Objective EXP-17: Information concerning unsafe system operating conditions should be provided to the human end user to enable them to take appropriate corrective action in a timely manner.
> ([2], page 71)

For the purpose of explainability of the ML component, the pilot should be aware if the current system state is currently inside or outside of ODD parameters. Furthermore, the pilot should be trained to handle situations of exiting the ODD. Basically, as soon as the system is leaving the ODD, the ML functionality can no longer be trusted. However, it is possible to include multiple boundaries or warning levels before exiting the ODD completely. This would give the pilot additional time as well as information on how to handle the current situation and thus improve situational awareness.

IV. Flight Test

The objective of the flight test was to generate sets of images that are within and outside the ODD to investigate the impact of ODD filtering. Operational parameters included altitude, velocity, camera angle, and image metadata such as brightness. The flight campaign was held on the first week of October 2022 at the *National Experimental Test Center of Unmanned Aircraft Systems* near Cochstedt. The test area includes two vertiports and a container cluster of six units, see Fig. 1. The persons that should be detected have been represented by mannequins which look very similar to real humans on aerial images an allow for safer, cheaper and more flexible testing. The mannequins are attached to a base plate that prevents them from falling over. Due to its grey color, the base plate blends in with the concrete surface at the airport and is barely visible on aerial images. In total, twelve flights have been completed during the flight tests at different day-times, see Table 1. All flights were conducted with the same waypoint coordinates. However, the altitude, flight speed, camera angle and positioning of the mannequins have been changed between flights. Additionally, three layers of transparent adhesive tape have been added onto the camera lens to simulate a blur effect during the last flight. Across all flights 6993 images have been recorded with the onboard camera.

	Altitude	Velocity	Camera angle	Mannequins	Notes
1st flight	40 m	5 m/s	Vertical	None	
2nd flight	40 m	5 m/s	Vertical	Position 1	
3rd flight	20 m	5 m/s	Vertical	Position 1	Route to last waypoint at altitude of 30 m
4th flight	20 m	15 m/s	Vertical	Position 1	
5th flight	80 m	5 m/s	Vertical	Position 1	
6th flight	10 m	5 m/s	Vertical	Position 1	
7th flight	40 m	5 m/s	Vertical	Position 2	Aborted after 2 min due to power issues
8th flight	40 m	5 m/s	Vertical	Position 2	
9th flight	40 m	5 m/s	45° tilted forward	Position 2	
10th flight	40 m	5 m/s	45° tilted backward	Position 2	
11th flight	15 m	15 m/s	Vertical	Position 2	
12th flight	40 m	5 m/s	Vertical	Position 2	Adhesive tape for blurr effect

Table 1 Table of performed flight tests with variations in ODD parameters, altitude, velocity and camera angle. Additionally, the first flight was performed without mannequins, and the position of the mannequins was switched with the 7th flight and following flights.

Fig. 2 Pictures of the UA after full integration of the components.

A. Unmanned Aircraft

This section describes the hardware setup. The UA was originally designed for the City-ATM project [21]. However, the UA is also suitable as the basic technology demonstrator for UAM. As the UA a hexacopter configuration is used, see Fig. 2. It is a modified version of the DJI Matrice 600 Pro hexacopter which has been equipped with an open-hardware flight controller and an additional data link for payload. Furthermore, additional safety-relevant systems including a FLARM were added. The modified flight controller is a Pixhawk 4. It uses the PX4 flight stack as autopilot software. As a ground control station (GCS) software QGroundControl was selected. It runs on a dedicated tablet PC, which serves as a GCS for the drone. The data link is implemented via a Wi-Fi module with a a ground unit (connected to the PC on the ground) and an air unit (mounted on the UA). As a replacement for the DJI remote control, a usual remote control (RC) including a suitable receiver was installed. An additional safety-relevant system is the parachute which was specifically designed and tested by manufacture and than integrated on the DJI M600 Pro. It is a light weight and modular unit with integrated automatic parachute trigger. It also carries on board a Micro-Electro-Mechanical Systems (MEMS) and Global Positioning System (GPS) sensors to the flight state and release the parachute if necessary. Additionally, a flight volume monitoring (geofencing) is integrated into the automatic termination function for both the autopilot and the parachute system. The parachute system in combination with the DJI M600 air frame is also certified according to

the ASTM F3322-18 standard. To generally sensitise the manned aviation in the surrounding area, a FLARM and a landing gear with LED-lighting were integrated into the drone. These measures improve the visibility of UA for other participants in the air space. The payload rail is located under the drone and includes the following items, see Fig. 3: Raspberry Pi 4, Raspberry Pi camera, camera mount, FLARM, Wi-Fi module, Intel-Neural Compute Stick 2 (NCS2) Stick, Battery Eliminator Circuit (BEC) and the fuse-box. All these components are powered by the batteries of the drone. The camera mount was designed to allow a variable camera angle. The angle step was chosen to be 15 degrees.

B. Machine Learning Component

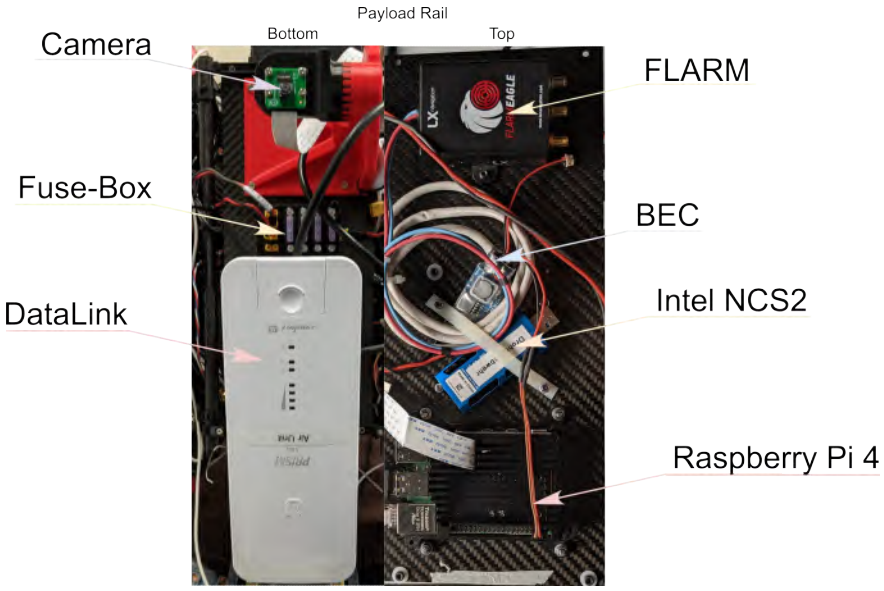

Fig. 3 Image of the payload rail, including the Raspberry Pi and the camera.

The ML component includes a Raspberry Pi 4 with 8GB of RAM, a Raspberry Pi Camera Module 2 and a Intel NCS2. The NCS2 is a USB-based deep learning inference kit and self-contained AI accelerator that delivers dedicated deep neural network processing capabilities with a low power consumption. The NCS2 uses the Movidius Myriad X Vision Processing Unit (VPU) to provide high performance neural network inference. The device is supported by the OpenVINO toolkit and can be used for a wide range of deep learning applications. The Raspberry Pi Camera Module 2 is a 8 megapixel camera that supports 720p video at 60 frames per second (FPS). The camera is directly connected to the camera serial interface (CSI) port on the Raspberry Pi.

V. Software Setup

The software setup is displayed in Fig. 4. The setup consists of a GCS, a Pixhawks and a Raspberry Pi. First of all, the GCS and the Pixhawk are to command and control the UA. Moreover, the Pixhawk has a special version of PX4 installed which allows to send sensor information to the Raspberry. The main software components have all been implemented on the Raspberry and will be explained with more detail in the following sections.

A. MicroRTPS Messaging

All the software components are implemented as Robot Operating System 2 (ROS2) modules. This enables a modular structure as well as reliable communication between the software components. By default, the Micro Air Vehicle Link (MAVLink) messages generated by the Pixhawk can not be distributed as ROS messages. However, the PX4 software includes a Data Distribution Service (DDS) interface which is also referred to as the microRTPS bridge. This bridge implements the Real-Time Publish Subscribe (RTPS) protocol which provides publisher-subscriber communication similar to the messages in ROS. The usage of the microRTPS bridge requires to run a software called microRTPS client on the Pixhawk and its counterpart, the microRTPS agent, on the Raspberry. The microRTPS agent

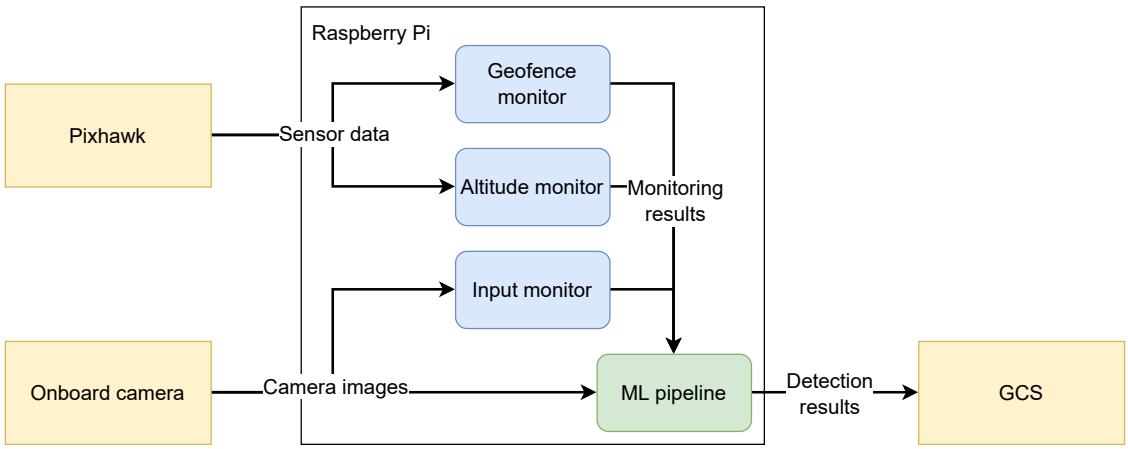

Fig. 4 Hardware (angular) and software (rounded) components of flight test setup

receives so called uORB messages from the Pixhawk and provides them as ROS messages for the other software components. The message types that should be supported by the microRTPS brigde can be specified according to the use case. For the geofence and altitude monitoring we require the message types containing the global position in World Geodetic System 1984 (WGS84) coordinates and the local position in North East Down (NED) coordinates.

B. ML Pipeline with Onboard Detection

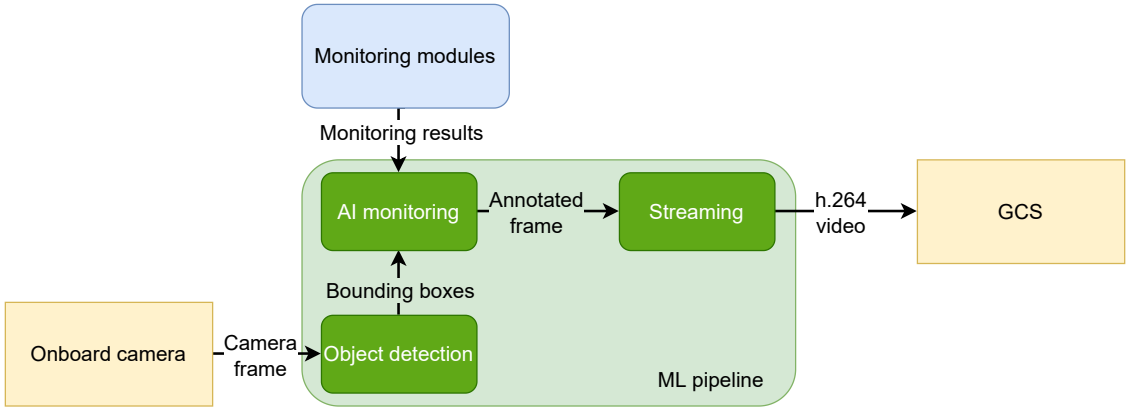

Fig. 5 High-level structure of ML Pipeline

The ML pipeline is a core component of the software setup and is responsible for the onboard detection of humans on aerial images. As depicted in Fig. 5, the ML pipeline directly reads frames from the connected Raspberry Pi camera and uses an object detection algorithm to determine bounding boxes for each person present in the current frame. Combined with the results from the input monitor, the geofence monitor and the altitude monitor, these bounding boxes are then fed into an AI monitoring module which determines whether the results generated by the object detection are trustworthy or not. The bounding boxes and the monitoring results are then drawn onto the frame. Afterwards, this annotated frame is streamed to the GCS where it can be analyzed by a GCS operator.

The object detection algorithm is implemented using a neural network based on the YOLOv4-tiny architecture [22], a reduced version of the YOLOv4 architecture with lower performance requirements. This network is trained using aerial images from the HERIDAL database [23] as well as synthetically generated images. However, the training images mostly represent rural areas, which does not match the flight test scenario on a airport. The size of the input images has been customized to 1280 x 960 pixels. All of the training has been done using the darknet framework*. The darknet

*https://github.com/AlexeyAB/darknet

weight and configuration files are directly supported by OpenCV[†], which allows to load and execute the trained model with OpenCV. There are multiple backends implemented in OpenCV to run the object detection on different target hardware. The model can be executed on a CPU, a GPU or on a VPU like the NCS2. Execution on other hardware than a CPU requires the support of additional frameworks. For example the execution on a Nvidia GPU requires the CUDA framework [‡] and the execution on a NCS2 requires the OpenVino framework[§]. In both cases the respective framework needs to be installed locally and OpenCV must be compiled with the support for the respective framework to run the model on the desired hardware. To bypass the limited CPU power of the Raspberry Pi, a NCS2 is being used which is directly connected to the Raspberry Pi. With the NCS2, it is possible to detect humans at a rate of up to 1.3 FPS. That means that the time between a captured image and the detection results is lower than 1 s which should be fast enough for slowly landing UA. However, for a real-life system a rate of at least 5 FPS would be desirable. Still, using the NCS2 is a major improvement compared to running the model directly on the CPU of the Raspberry Pi which only provides a rate of 0.25 FPS. Using the NCS2 also reduces the load on the CPU and leaves more resources for the other software components like the monitoring modules.

The outputs of these monitoring modules are used to verify the object detection results. The outputs include information about the altitude and the position of the UA as well as properties of the current camera frame. A comparison of the monitoring outputs and predefined ODD boundaries is being done in the AI monitoring component, which adds annotations to the current frame including the bounding boxes of the detected persons and information on whether the ODD boundaries have been violated. For example, it can be checked whether the current altitude matches the altitudes of the images used for the training of the neural network.

The frames with annotations are then processed by a streaming component which converts them to an h.264 video stream that is transmitted to the second GCS via UDP. The conversion and the streaming have both been implemented with the GStreamer framework[¶]. To avoid a heavy CPU usage for the h.264 conversion, the built-in hardware encoding of the Raspberry Pi is utilized. This enables an efficient video stream between Raspberry Pi and the second GCS. The visualization of the video stream in the GCS software can be seen in Fig. 6.

C. Input Monitor

The images that are being fed into the neural network could suffer from multiple negative effects such as low contrast, blur, under- and overexposure or noise from damaged camera sensor. Depending on the intensity of these effects, a reliable detection might become impossible. To filter out images with such effects the input monitor reads the current frame from the connected Raspberry Pi camera and computes several properties of the frame. These properties include the brightness, the saturation, the entropy and the amount of edges in the frame. Additionally, the image sharpness and the contrast can be checked. The computation of these properties is as follows.

Measuring the brightness of an image could be used to check if under- or overexposure exist in an image. It can be computed by converting the image to grayscale and calculating the mean value of all pixels. To get a value range between 0 and 1 the mean value is divided by 255. A very low value could indicate underexposure while a high value would indicate overexposure.

An image with low saturation could be an indicator for an reduced set of distinct features in an image, e.g. in case of hazy weather. To compute the saturation, the image is converted to the Hue Saturation Value (HSV) color space before the mean value of the saturation channel is calculated. Again the saturation is normalized by dividing by 255.

Low contrast could also have a negative impact on the detection performance of the neural network. As there is no standardized method to compute the contrast of an image, the contrast is substituted by the entropy. The first step for the entropy computation is a conversion to grayscale. After that, the histogram of the image is computed. The probability of each pixel value can be determined by dividing the number of pixels with the same value by the total amount of pixel. With these probabilities we can compute the entropy with equation 1 where H corresponds to the entropy and $p(i)$ to the probability of one pixel value i.

$$H = -\sum_{i=0}^{255} p(i) \cdot \log_2 p(i) \tag{1}$$

[†]https://opencv.org/

[‡]https://developer.nvidia.com/cuda-toolkit

[§]https://docs.openvino.ai

[¶]https://gstreamer.freedesktop.org/

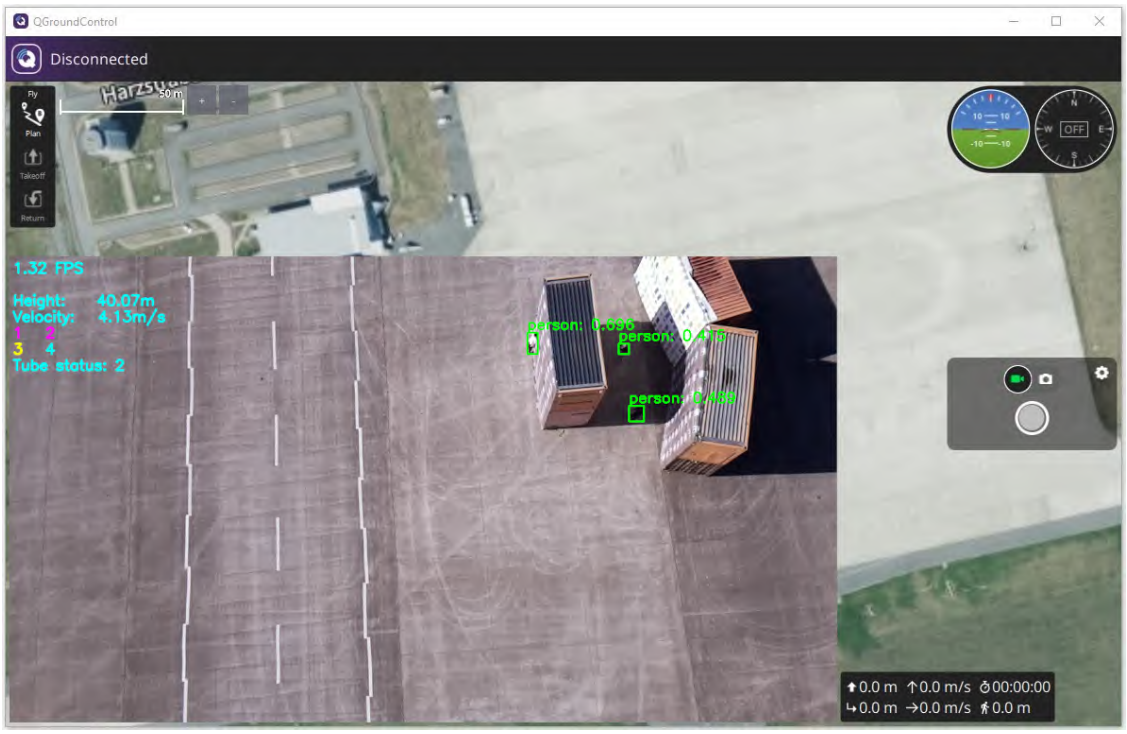

Fig. 6 Software demonstrator prototype, showing a screenshot of the QGroundControl software, streaming an image from the onboard camera. The map of the flight test area is shown in the back and the streamed image is shown in the front.

To normalize the entropy, its value is divided by the maximum possible entropy H_{max} which corresponds to $H_{max} = -\log_2 \frac{1}{256}$.

Similar to the other metrics, the amount of edges in a frame are determined by first converting the image to grayscale. After that, a Laplace filter with a kernel size of 3 is applied to the grayscale image. Then the histogram of the filtered image is computed. The bright pixel values in the histogram correspond to edges while the dark pixels are areas without edges. We defined that all pixel values below 25 in the histogram are areas without edges while everything else corresponds to an edge. So the edge value would be 0 if all values in the histogram are below 25 or 1 if all values in the histogram would be above 25.

D. Geofence Monitor

Operation information are an important credential when setting up the training of a neural network. For instance, training data of human detection in a field do not have to cover all the features that a flight in a city has. Yet, if assumptions of the operation are made, they need to be checked *at runtime* to prevent entering a flight area where unforeseen features might occur. To reduce the likelihood of these unforeseen features and a shift in input distribution, the geofence monitor is used.

A geofence represents virtual barriers in space that the system under scrutiny is not allowed to cross. If the operation consists of different tasks that can be spatially separated it makes sense for an improved situational awareness to have overlapping geofences where a crossing represents the traversal from one task to another. Fig. 7 depicts four overlapping geofences that were used for the flight test, see Section IV. *Geofence 1* contains the first vertiport, *Geofence 2* contains the second vertiport, *Geofence 3* contains the container cluster, and in *Geofence 4* the waypoints mission stopped. Each subplot represents the point of view of a respective geofence highlighted in blue. The flight path is depicted in green, orange, and red referring to inside geofence, close to geofence violation, and outside geofence, respectively. The figure shows that the waypoint mission was correctly tracked by the geofence monitor. The mission first started in *Geofence 1*, moved to *Geofence 2*, then to *Geofence 4*, then to *Geofence 3*, and finally ended up in the initial geofence.

Complementing the geofence of the operation area, geofencing of the planned waypoint mission was also used,

which we refer to as tubes. Fig. 8 shows the tubing results for our flight test. The pre-planned waypoint mission is depicted in blue. The flight is indicated by a color varying line where green, orange, and red indicates when the UA was inside, near to the border of the tube, and outside of the tube, respectively. The numbers in the figure that range from zero to nine reflect the order of the waypoints. After the last waypoint is reached the waypoint mission stops. The figure shows that we were able to follow the pre-planned waypoint mission, i.e. , we stayed inside the tube. Only when the last waypoint was reached and the UA returned to the home position did we leave the tube. Besides tracking the position within the tube, we also notified when too much time expires until the next waypoint was reached in respect to a ϵ distance to the position of the waypoint. This can be seen in Fig. 9 where brighter colors indicate a larger value of the timer. For instance, the timer was not reset at *Waypoint 3* and *Waypoint 9* since they were not reached within the ϵ distance. Yet, all other waypoints were sufficiently reached, resetting the timer.

Geofencing of the operation area as well as the pre-planned flight trajectory allows to reduce the likelihood of unforeseen features a neural network might encounter. It is one means to assure that the input distribution of a neural network is similar to the one during training. Another means is the Altitude monitor presented next.

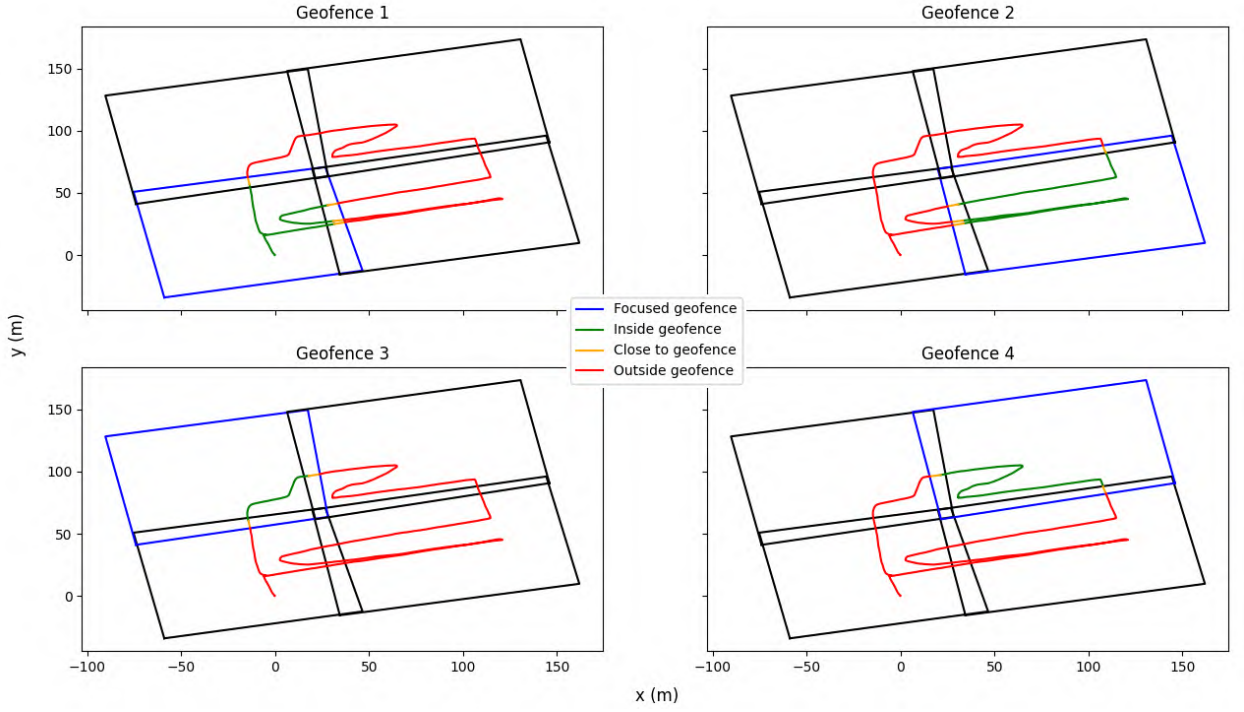

Fig. 7 Top-view of the overlapping geofences used for the flight test. In each subfigure, the monitor's outputs are given for the respective geofence highlighted in blue where green, orange, and red represent inside, close to crossing, and crossed monitor evaluations.

E. Altitude Monitor

The neural network is trained with images that have been taken at specific altitudes. To ensure the safe functionality of the object detection algorithm, the current altitude is compared with the altitudes of the images that have been used in the training process. For example, if images with altitudes between 20 and 50 meters have been used in the training process, the object detection might not work properly below 20 m and above 50 m. As mentioned in section V.A the Pixhawk outputs a global position in WGS84 coordinates. However, the altitude in this global position measures the distance to WGS84 ellipsoid and not the distance to the ground. To obtain the altitude above ground, the altitude monitor loads a terrain map from the flight area. With this terrain map, the altitude above ground can be computed by subtracting the terrain height at the current position from the WGS84 altitude. The resulting height above ground is then provided for the AI monitor in the ML pipeline described in section V.B.

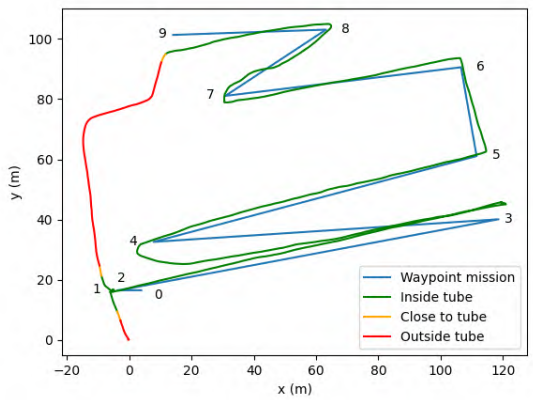

Fig. 8 Top-view of the tubing results. We remained within the tube (green) until the waypoint mission stopped at Waypoint 9. Then, the UA left the tube to fly to the home position.

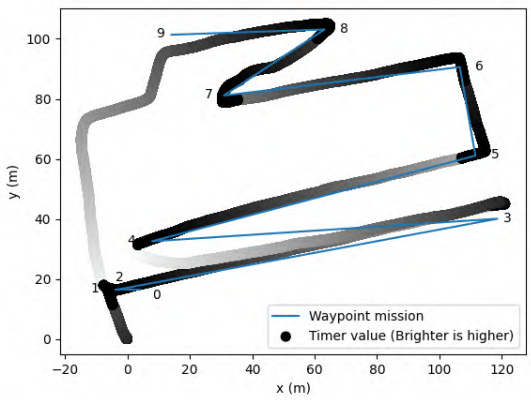

Fig. 9 Top-view of the tubing timer. At each waypoint, a timer is started that is reset when the next waypoint is reached. The brighter colors of the flight show that we didn't reach Waypoint 3 and Waypoint 9 sufficiently to reset the timer. All other waypoints reset the timer.

VI. Results

As mentioned before in section IV, a total amount of 6993 images has been recorded across flights. These images have been used to evaluate the overall software setup. First, some of the recorded images have been used to evaluate the performance of the ML component and the influence of the ODD on the detection performance. Second, the properties of the recorded images have been analyzed to determine if filtering of blurred images using the input monitor could be feasible.

A. Preliminary Evaluation of the ML Component

To evaluate the performance of the ML algorithm, 587 images were randomly selected from the recorded images as a test set. Due to the missing mannequins in the first flight and the artificial blur in the last flight, detections for these flights were not possible. Therefore, the images from these flights were not considered for the test set. The 587 images in the test set have been labeled manually and have been used to compute the precision and recall scores of the ML component for the flight tests. The precision P and the recall R are defined as $P = \frac{TP}{TP+FP}$ and $R = \frac{TP}{TP+FN}$ respectively, where TP corresponds to the number of true positives, FP to the number of false positives and FN to the number of false negatives.

In the context of object detection, a true positive exists when the bounding box of a detected object matches the bounding box of a ground truth object. A metric that evaluates the similarity of two bounding boxes is the intersection over union (IoU) which is defined as $IoU = \frac{A \cap B}{A \cup B}$ with A as the first bounding box and B as the second bounding box. Typically, an IoU threshold is used to determine whether two bounding boxes match each other. In this paper, a threshold of 0.5 is being used, which means that two bounding boxes are evaluated as true positive when their IoU score is larger or equal than 0.5. False positives on the other hand are bounding boxes that have been detected by the object detection algorithm, but can not be matched with a ground truth bounding box. In contrast, false negatives are the amount of ground truth bounding boxes that have not been matched with any detected bounding box.

In total, the ground truth labels contains 684 objects. The object detection algorithm managed to detect 120 of those objects with an IoU score larger than 0.5 which results in 120 true positives, 406 false positives and 564 false negatives. The resulting precision of the object detection algorithm is 0.228 and the recall is 0.175, see Fig. 10. However, if the images with altitudes higher than 50 m or lower than 20 m as well as images with other camera angles and velocities higher than 10 m s^{-1} are ignored, only the images within the ODD remain. In that case, the precision increases to 0.392 and the recall increases to 0.305.

Despite overall low performance scores, a positive impact of filtering out images outside the ODD for the detection performance is clearly visible. The relatively low performance is probably mainly caused by the training data which mostly represents rural areas that are dissimilar to the environment of an airport. Therefore, retraining the neural

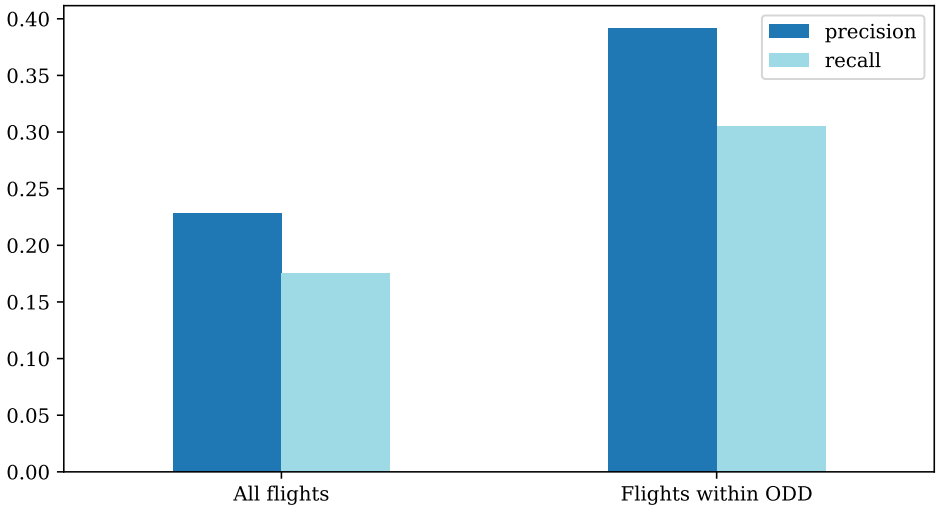

Fig. 10 Comparison of recall and precision between all flights and all flights within the ODD

network with airport related images or images with more urban environments could improve the detection performance of the ML component significantly.

B. Preliminary Evaluation of the ML Safety Aspects

The use case of human detection, with aforementioned results is currently not on an adequate performance and safety level for a practical use. However, this use case can be used to research the safety aspects of the ML component. From the results it shows that there is a significant difference in precision and recall over all images compared to the precision and recall over the images that are specifically inside the ODD. Therefore, the object detection algorithm has better results for images that comply with the ODD compared to images that violate the ODD. Furthermore, within this research we are currently focusing on monitoring of different aspects. By monitoring the ODD during the operation, we safeguard the ML component against input images that would not result in an adequate performance and for which the ML algorithm is not intended for, such as discussed within the current EASA guidelines, see also section III.A.

As a specific use case, a sensor error has been emulated. For the last flight, three layers of transparent adhesive tape have been used to simulate a blur effect. To check whether the blur can be detected using image properties, the recorded images haven been analyzed using the input monitor described in section V.C. For each recorded image, four properties have been computed: the brightness, the saturation, the entropy and the amount of edges in the image. These properties are displayed in two box plots which can be seen in Fig. 11. The first box plot refers to the images from the flight with the adhesive tape and the second box plot refers to the images from the other flights without the adhesive tape. In both plots the distributions of the brightness, the saturation and the entropy are fairly similar. Therefore these properties would not be suitable to distinguish between normal images and blurred images. However, the edges property differs significantly between the two plots. The highest edges value for the images with blur is 0.01 while 0.1 is the lowest recorded value for the remaining images. This demonstrates that a detection of blur in images using information about the amount of edges in an image might be feasible. This can safeguard the ML component and significantly improve the overall performance of the ML component. However, at this moment the loop for filtering images that are outside of the ODD is not yet closed in flight.

VII. Conclusion and Outlook

In this paper, the current state of safety and assurance aspects for autonomy and ML in the context of UAM have been discussed. An exemplary ML-based autonomy function, the detection of persons on the ground during flight for assessing potential landing areas, has been used to lay out and implement a system architecture that assures

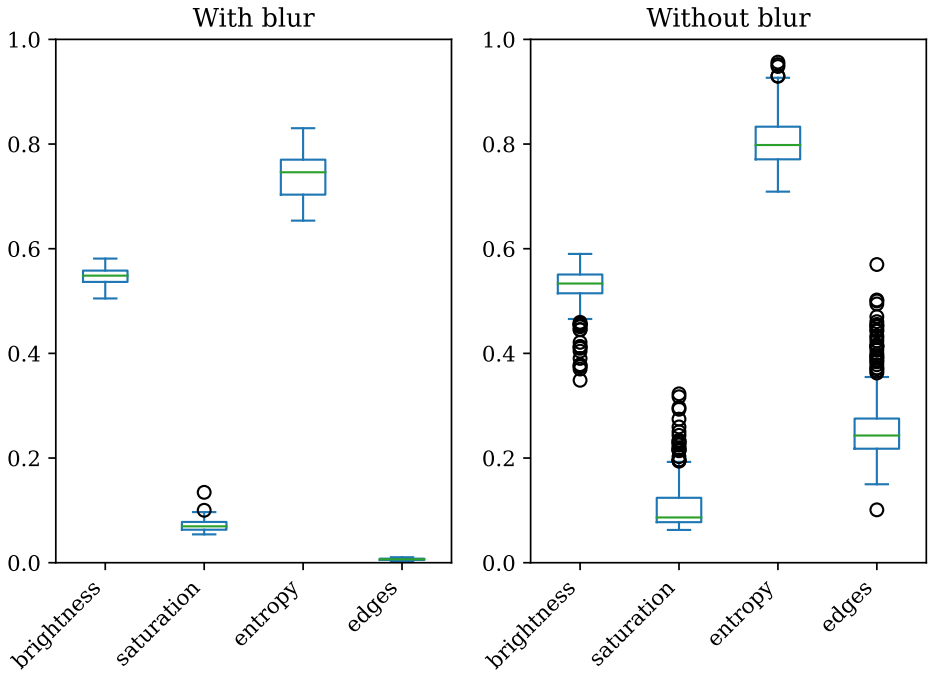

Fig. 11 Image properties of flight test data

proper operational conditions for the ML algorithm. Within this context, key concepts of EASA guidelines on AI trustworthiness analysis, safety risk mitigation and AI explainability have been discussed. A special focus is set on monitoring conformance to the algorithm's ODD at runtime. The ML functionality itself, as well as the safety risk mitigation using ODD monitoring has been tested in flight tests. The ML performance during the flight tests has been analyzed. Furthermore, the effects of monitoring exemplary ODD properties have been evaluated. In the experimental setup of this work, the ODD monitoring and assurance is not yet integrated with the autopilot. In future work, this loop will be closed to allow for automated risk mitigations in case of non-conformance to ODD requirements. Although, achieving full compliance to EASA guidelines was outside the scope of this work, selected objectives could be analyzed, implemented and flight tested. Future work will build upon this and discuss more assurance aspects for the safe use of ML in UA and the UAM domain.

Acknowledgment

This work was partially supported by the Aviation Research Program LuFo of the German Federal Ministry for Economic Affairs and Energy as part of "Volocopter Sicherheits-Technologie zur robusten eVTOL Flugzustandsabsicherung durch formales Monitoring"(No. 20Q1963C).

References

[1] Torens, C., Volkert, A., Becker, D., Gerbeth, D., Schalk, L., Garcia Crespillo, O., Zhu, C., Stelkens-Kobsch, T., Gehrke, T., Metz, I. C., and Dauer, J., "HorizonUAM: Safety and Security Considerations for Urban Air Mobility," AIAA AVIATION Forum, American Institute of Aeronautics and Astronautics, 2021. URL `https://arc.aiaa.org/doi/10.2514/6.2021-3199`.

[2] European Union Aviation Safety Agency (EASA), "Concept Paper First Usable Guidance for Level 1 Machine Learning Applications," `https://www.easa.europa.eu/downloads/134357/en`, 2021.

[3] Torens, C., Juenger, F., Schirmer, S., Schopferer, S., Maienschein, T. D., and Dauer, J. C., "Machine Learning Verification

and Safety for Unmanned Aircraft - A Literature Study," *AIAA SCITECH 2022 Forum*, American Institute of Aeronautics and Astronautics, 2022. https://doi.org/10.2514/6.2022-1133.

[4] Torens, C., Durak, U., and Dauer, J. C., "Guidelines and Regulatory Framework for Machine Learning in Aviation," *AIAA SCITECH 2022 Forum*, American Institute of Aeronautics and Astronautics, 2022. https://doi.org/10.2514/6.2022-1132.

[5] European Union Aviation Safety Agency (EASA), "Artificial Intelligence Roadmap, A Human-Centric Approach to AI in Aviation, Version 1.0," https://www.easa.europa.eu/newsroom-and-events/news/easa-artificial-intelligence-roadmap-10-published, 2020.

[6] European Union Aviation Safety Agency (EASA), "Concepts of Design Assurance for Neural Networks (CoDANN)," https://www.easa.europa.eu/sites/default/files/dfu/EASA-DDLN-Concepts-of-Design-Assurance-for-Neural-Networks-CoDANN.pdf, 2020.

[7] European Union Aviation Safety Agency (EASA), "Concepts of Design Assurance for Neural Networks (CoDANN) II," https://www.easa.europa.eu/document-library/general-publications/concepts-design-assurance-neural-networks-codann-ii, 2021.

[8] Federal Aviation Agency (FAA) and Daedalean, "Neural Network Based Runway Landing Guidance for General Aviation Autoland," Tech. Rep. at, 2022. URL https://daedalean.ai/tpost/g2s3nhz4u1-daedalean-concluded-a-joint-research-pro.

[9] SAE G-34, Artificial Intelligence in Aviation, "Artificial Intelligence in Aeronautical Systems: Statement of Concerns," Tech. rep., SAE International, 2021. https://doi.org/10.4271/AIR6988.

[10] Dmitriev, K., Schumann, J., and Holzapfel, F., "Toward Certification of Machine-Learning Systems for Low Criticality Airborne Applications," *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, 2021, pp. 1–7. https://doi.org/10.1109/DASC52595.2021.9594467.

[11] Dmitriev, K., Schumann, J., and Holzapfel, F., "Toward Design Assurance of Machine-Learning Airborne Systems," *AIAA SCITECH 2022 Forum*, 2022, p. 1134.

[12] Dmitriev, K., Schumann, J., and Holzapfel, F., "Towards Design Assurance Level C for Machine-Learning Airborne Applications," *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*, 2022, pp. 1–6. https://doi.org/10.1109/DASC55683.2022.9925741.

[13] The British Standards Institution, Center for connected and autonomous vehicles, "PAS 1883:2020 Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) – Specification," , 2020.

[14] Colwell, Ian, "Runtime Restriction of the Operational Design Domain: A Safety Concept for Automated Vehicles," Master's thesis, 2018. URL http://hdl.handle.net/10012/13398.

[15] Gyllenhammar, M., Johansson, R., Warg, F., Chen, D., Heyn, H.-M., Sanfridson, M., Söderberg, J., Thorsén, A., and Ursing, S., "Towards an operational design domain that supports the safety argumentation of an automated driving system," *10th European Congress on Embedded Real Time Systems (ERTS 2020)*, 2020.

[16] Yu, W., Li, J., Peng, L.-M., Xiong, X., Yang, K., and Wang, H., "SOTIF risk mitigation based on unified ODD monitoring for autonomous vehicles," *Journal of Intelligent and Connected Vehicles*, , No. ahead-of-print, 2022.

[17] Koopman, P., and Fratrik, F., "How many operational design domains, objects, and events?" *Safeai@ aaai*, Vol. 4, 2019.

[18] Yang, J., Zhou, K., Li, Y., and Liu, Z., "Papers with Code - Generalized Out-of-Distribution Detection: A Survey," , 2021. URL https://paperswithcode.com/paper/generalized-out-of-distribution-detection-a.

[19] ASTM F38, "Standard Practice for Methods to Safely Bound Behavior of Aircraft Systems Containing Complex Functions Using Run-Time Assurance," https://www.astm.org/f3269-21.html, 2021. https://doi.org/10.1520/F3269-21.

[20] Nagarajan, P., Kannan, S. K., Torens, C., Vukas, M. E., and Wilber, G. F., "ASTM F3269 - An Industry Standard on Run Time Assurance for Aircraft Systems," *AIAA Scitech 2021 Forum*, American Institute of Aeronautics and Astronautics, 2021. https://doi.org/10.2514/6.2021-0525.

[21] Kern, S., Geister, D., and Korn, B., "City-ATM — Demonstration of Traffic Management in Urban Airspace in case of bridge inspection," *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–10. https://doi.org/10.1109/DASC43569.2019.9081663.

[22] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M., "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint arXiv:2004.10934, 2020.

[23] Dunja Božić-Štulić, Željko Marušić, and Sven Gotovac, "Deep Learning Approach on Aerial Imagery in Supporting Land Search and Rescue Missions," *International Journal of Computer Vision*, 2019.