DESIGN OF AI-DRIVEN COMPUTER VISION SOFTWARE FOR AERONAUTICAL TESTING

Eleonora Barbano
TXT E-TECH
Cologno Monzese, Milan, Italy

Valentina Maffei
TXT E-TECH
Cologno Monzese, Milan, Italy

David Frisini
TXT E-TECH
Cologno Monzese, Milan, Italy

ABSTRACT

In the aeronautical industry, the safety, functionality, and reliability of aircraft systems are fundamental. Traditional testing processes are often time-consuming and resource-intensive. This paper presents the design and implementation of AI-driven computer vision software aimed at optimizing these testing processes through automation. By leveraging advanced Convolutional Neural Networks (CNNs) and Optical Character Recognition (OCR) algorithms, the software can perform real-time detection and post-flight video analysis. Specifically, a YOLOv8 model has been fine-tuned for object detection tasks within various testing phases, including unit tests, system integration tests, and flight tests. The integration of PaddleOCR enhances the software functionalities adding text recognition capabilities. This approach not only speeds up the testing process but also significantly reduces human effort to manually collect data, leading to improved efficiency and human errors. The paper further explores the implementation details, challenges, and performance metrics of the applied software, demonstrating their potential to improve aeronautical testing procedures.

NOTATION

*

Acronyms

AI Artificial Intelligence

ARTO Automated Robotics for Testing Optimisation

CAS Crew Alerting System

CER Character Error Rate

CNN Convolutional Neural Network

CV Computer Vision

EFIS Electronic Flight Instrument System

FCU Flight Control Unit

FPS Frames Per Second

FTI Flight Test Inconvenience

HSV Hue-Saturation-Value

IC Image Classification

IS Image Segmentation

MCDU Multi-Function Control and Display Unit

ML Machine Learning

OD Object Detection

OCR Optical Character Recognition

PFD Primary Flight Display

RGB Red-Green-Blue

RoI Region of Interest

SUT System Under Test

TCP Tool Center Point

UFCP UP-FRONT CONTROL UNIT

INTRODUCTION

In the aeronautical industry, rigorous testing is essential to ensure aircraft systems' safety, functionality, and reliability. This process involves multiple stages of testing, as represented in Figure 1, each serving a distinct purpose. **Unit Tests** represent the first level of component tests, focusing on individual components or subsystems in isolation to verify that each part functions correctly according to its design

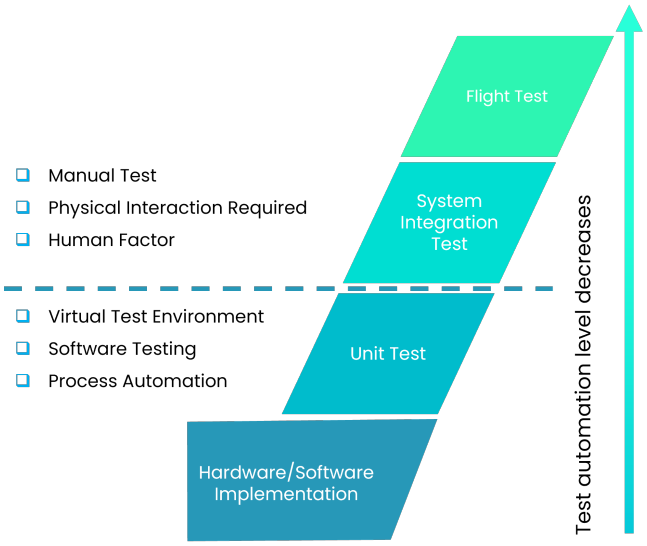

Figure 1. Validation model diagram. Credits: TXT E-Tech.

specifications. This involves testing software modules, electronic circuits, mechanical parts, and other discrete elements to identify and rectify defects early in the development process. Unit tests represent the first step for more complex integration efforts by ensuring that each unit operates as intended. Once this phase is complete, **System Integration Tests** are conducted. The primary goal is to evaluate the interactions and interfaces between integrated components, ensuring they work together seamlessly and reliably under expected operational conditions. This type of testing can reveal issues related to data flow, control signals, power distribution, and communication protocols between subsystems. Effective system integration testing is crucial for uncovering and addressing integration issues before they can affect overall system performance. The final and most comprehensive level of testing is the **Flight Test**. Conducted in real flight environments, flight tests validate the aircraft's overall performance, safety, and regulatory compliance. Flight testing includes a wide range of evaluations, such as aerodynamics, propulsion, avionics, flight control systems, and structural integrity under various flight conditions. Pilots and engineers closely monitor the aircraft's behavior during different phases of flight, including takeoff, cruising, maneuvers, and landing. The insights gained from flight tests are critical for certifying the aircraft's airworthiness and ensuring it meets all safety and performance criteria. In these activities, operators are involved in analyzing up to hundreds of in-flight video hours manually identifying alerts, warnings, and other relevant information on displays over the video timeline.

The analysis of these results is essential but can be expensive, time-consuming, and repetitive. The iterative nature of these tests requires significant resources and meticulous attention to detail to ensure every component and system meets stringent safety and performance standards. However, advancements in Artificial Intelligence and Machine Learning offer promising solutions to speed up these phases. By automating defect detection, predictive maintenance, and anomaly identification, ML algorithms can reduce the need for repetitive manual analysis and accelerate the testing process. Additionally, ML algorithms can analyze vast amounts of test data to identify patterns and predict potential failures, further enhancing the efficiency of the testing process. As a result, integrating ML into aeronautical testing can lead to substantial cost savings, improved accuracy, and faster development cycles.

COMPUTER VISION ALGORITHMS

Computer Vision is a field of Artificial Intelligence that enables computers to interpret and make decisions based on visual data. Similar to the human learning process, CV systems can analyze images, videos, or other visual inputs to extract meaningful information. The ability of CV algorithms to interpret and understand visual information, driven by improvements in deep learning, has led to a revolution in various domains supporting and sometimes surpassing human performances (Refs. 1–3). To achieve this level of understanding, CV relies on sophisticated architectures, such as Autoencoder for unsupervised learning or Convolutional Neural Networks for supervised learning. In the field of supervised learning, CNNs are specialized models designed to process and analyze visual information by learning from labeled data, with the labels defined according to the algorithm's goal. In this context images, used as input data, are represented as matrices, where each element corresponds to a pixel in color code. By applying filters that traverse the matrix, CNNs can detect edges, textures, shapes, and other features. These are combined in non-linear ways, enabling the models to recognize complex elements within the visual input. In supervised learning, CNNs can be used for:

- Image Classification: a label, class or category is assigned to an entire image based on its visual content. The model learns from the supervised label to identify the main object or scene in the image and classify it into one of the predefined categories.
- Object Detection: each class is assigned to a specific portion of the image based on its visual content, locating by drawing usually rectangular bounding boxes around each detected object. This process provides both the class label and the position of each object in the image. The model learns from the supervised label not only to identify but also to localize different objects in the image and classify them into one of the predefined categories.
- Image Segmentation: each class is assigned to a single pixel of an image, often used to identify and separate different objects or areas within the image. The model learns from the supervised label of each pixel the shape of the classified objects and their position.

In addition, CNNs can be used alone, or in combination with other architectures (i.e. Recurrent Neural Networks, Long

Short-Term Memory Neural Networks) to detect, recognize, and convert the text present in images or video (i.e. scanned documents, photos with road signs...) in string variables. This field is known as Optical Character Recognition.

This paper demonstrates the efficacy and advantages of applying CV deep learning models within the aeronautical industry during unit testing, system integration testing, and post-flight analysis processes. To develop supportive software for testing engineers, firstly it is crucial to automate the recognition of elements and features such as buttons, switches, levers, knobs, and displays within different instruments and inside various cockpit environments. Based on these CV requirements in aeronautical testing, the advantages and disadvantages of applying different supervised CNN models are summarized in Table 1. Image Classification, Object Detection, Image Segmentation, and Optical Character Recognition functionalities are compared.

Table 1. Comparison between different CV algorithms.

CV requirements	IC	OD	IS	OCR
Computational time impact	low	medium	high	medium
Recognise SUT (i.e. FCU, MCDU)	✓	✓	✓	×
Recognize component on SUT (i.e. button on FCU, display on MCDU)	×	✓	✓	×
Identify component state on SUT (i.e. button ON on FCU, page DATA on MCDU)	×	✓	✓	×
Read component content on SUT (i.e. button "AP 1" ON on FCU, "GPS1" Position on MCDU)	×	×	×	✓
Read alert or warning messages	×	×	×	✓

IC would be performative for classifying the singular systems under test such as FCU or MCDU, but it would be inefficient for recognizing their internal components such as knobs, buttons, or display elements. To give an example, a unit test requires to press "AP 1" button on FCU to check if its status switches consistently from OFF to ON. In this case, a camera frames the entire FCU system while an operator presses the button. IC applied on the frame would classify the image as an FCU, but the inference via IC can only determine the classification of the entire picture. To check the status of a single component such as a button would be hardly completed by using a wide generic IC. On the other hand, OD and IS would be regardless used in this context, their main difference is represented by the real-time processing constraint. The computational complexity and time required to perform image segmentation on the FCU frame might be too high in this context, where additionally no high level of precision at the edges is required: it is necessary to localize the area of buttons, not exactly the pixels that compose them. Moreover, the data variability is moderate: tests can indeed be performed in different cockpits and luminosity conditions of the environment, but a button is a "simple" object compared to features detectable by IS models (i.e. in medical imaging context, where different types of tissues, organs, or abnormalities must be segmented, solving more complex problems). For these reasons, to automate aeronautical tests the focus is on the application of CNN models for OD, representing a reasonable compromise between short computational time and the complexity of the tasks. The inference via OD of the FCU image can highlight with rectangular RoI the area of each button. Additionally, it is essential to integrate an OCR model to create a comprehensive and complete support system for test engineers. This integration is required despite the medium level of complexity of these types of models, but it aims to guarantee the system with complete visual capabilities. CV software must be able to read texts on buttons, levers, and on display such as alerts, and warning messages, and to associate each component with its meaning: OD recognizes an area of the image (RoI) as a button ON and OCR reads "AP 1" on it. The system can acquire the information that if "AP 1" button is in ON state, autopilot mode is engaged.

The use of OD and OCR models in the software incurs significant computational expense due to their medium computational time impact. OD models can be computationally intensive, especially if they need to perform real-time inference, requiring substantial processing power for frame-by-frame analysis. Similarly, OCR model demands considerable resources for accurately detecting and recognizing text in each frame. The combined use of these models necessitates robust hardware specifications to ensure efficient and timely processing. Table 2 provides the technical specifications of the computer used during software development and applications. Through the use of this powerful machine, it is possible to analyze both real-time frames for unit and system integration tests, as well as heavy videos longer than an hour in post-flight analysis. These hardware components ensure that the software can handle the intensive computational load required to reach high levels of accuracy and efficiency. This is fundamental to enable the CV system reaction with a minimal delay.

Table 2. Computer Characteristics

Characteristics	
RAM	128 GB
SSD	1 T + 1 T
GPU	GeForce RTX 4090
VRAM	24 GB
CPU	Intel Core i9 13th

Object Detection and OCR Implementation for Real-Time Cockpit Analysis

Concerning Object Detection, YOLOv8, developed by Ultralytics, represents one of the latest advancements in real-time OD and IS (Refs. 4, 5). Largely used for their exceptional balance between speed and accuracy, these models introduce numerous enhancements over their predecessors, including improved architectures, better handling of small objects, and more efficient use of computational resources. In addition, the 8th version has been chosen because it is the latest release at the time of algorithm implementation. YOLOv8 architectures are specialized for various tasks, ranging from IC and OD to more complex assignments like oriented OD, instance segmentation, and pose/keypoint detection. For the

reasons discussed in the previous paragraph, the pre-trained weights of one of the available OD task architectures have been chosen and fine-tuned on specific datasets. In particular, YOLOv8n (nano), the smallest model in the YOLOv8 proposals, has been selected. Thanks to its compact size ($\sim$ 3 million parameters pre-trained on 80 generic classes on COCO val2017 dataset (Ref. 6)), it can process images quickly, making it ideal for the real-time applications necessary in aeronautical unit and system integration test fields. Since the frames observed by the cameras can be different for luminosity or angle, but the features that have to be recognized inside cockpit environments are usually standard between different prototypes (i.e. levers, switch, buttons, element on displays...), the choice has fallen on a smaller architecture in favor of a higher speed in inference.

Regarding OCR, the performances of two models have been compared: Tesseract-OCR (Ref. 7) and PaddleOCR (Ref. 8). Tesseract is an open-source OCR engine (by Hewlett-Packard and later maintained by Google). It has a large user community mainly due to its high accuracy, adaptability to many different languages, and uncommon inference speed. Despite these, Tesseract's model reaches the best performances if applied to PDF files, scanned book pages, or continuous texts with high and constant contrast with the background. Tesseract's performances can easily degrade with complex layouts, low-quality images, sparse texts, or text with changing background color. Image pre-processing with Tesseract is crucial to improve OCR accuracy, such as converting data to binary (black and white) format, enhancing contrast, or reducing noise. It is also necessary to apply straightening to the images where text is not placed in a perfect horizontal shape. For the context of the applications, it is almost impossible to fix a priori these parameters, especially in a real-time context when the camera frames each time at slightly different conditions of light and position. After numerous trials of applying Tesseract to frames captured in the cockpit during real-time testing phases, it has been concluded that despite its incomparable speed in inference, the extensive and complex pre-processing required for the images makes it impractical. This has led to the exploration of alternative OCR models (i.e. EasyOCR (Ref. 9), Keras (Ref. 10), Paddle (Ref. 8)). Paddle (by Baidu as part of the PaddlePaddle deep learning framework) has the main advantage of having low computational time in inference and high accuracy in text recognition, even with not pre-processed frames for different luminosity conditions. In addition, by employing this model, it is possible to leverage any GPU present in the physical machine, thereby further accelerating text recognition. Its main limitations are noisy or low-resolution images suggesting that to reach optimal performance it is crucial to provide PaddleOCR with high-quality images or videos. When this is not possible, implementing image enhancement or text correction techniques can mitigate some of the recognition issues caused by low resolution.

Fuzzywuzzy (Ref. 11) library based on Levenshtein (Ref. 12) distance has been used to construct matching algorithms between the "raw words" found by Paddle and a complete list of real solutions. The Levenshtein distance quantifies the difference between two strings by determining the minimum number of operations needed to convert one string into the other. These operations include inserting or deleting a letter, switching two adjacent characters, or substituting a character. A smaller Levenshtein distance indicates greater similarity between the strings. In this way, it is possible to replace a potential nonsense word read by Paddle, but however similar to the one displayed, with the correct text. For instance, due to low resolution, the letter 'i' can be confused with the letter 'l' as the dot on the 'i' can be blended into the body by the noise, likewise 'E' and 'F' can be misunderstood. In warning and error messages that appear on the displays, Paddle may read a message like "Engine Fail" as "Fnglne Fall" or similar. In this case, the Levenshtein distance is 3: replace F, l, and l with E, i, and i. By giving the raw output in input to a matching algorithm, it is possible to return the correct string "Engine Fail" knowing a priori the correct and complete list of all messages or texts that may appear on the screen. Different matching algorithm implements the Levenshtein distance, but in this context, the simple ratio metric has been used. This metric has been chosen because it does not consider word order or partial matches. In the example above, "Engine Fail" and "Fnglne Fall" have a similarity ratio equal to 70. The implemented algorithm assigns a Levensthtein ratio to each world of a prefilled dictionary and returns the first N occurrences exceeding an established threshold in Levenshtein simple ratio metric. The main weakness of this solution is that the list of texts appearing in a cockpit environment can be long and contain different but very similar messages (i.e. "DC Gen 1", "DC Gen 2", "DC Gen 3", "DC Gen 1-2-3"), so it is possible to have more than one message with a Levensthein ratio higher than the set threshold. It can also happen to have more than one message with the same Levensthein ratio. To further harden the matching algorithm, for the text that can only appear in color-code (i.e. "Engine Fail" can be displayed only in yellow) a color extraction function has been added to the system. The error messages have been divided into different dictionaries depending on the color in which they are displayed (for the performed tests red, yellow, cyan, and white in gravity descending order). Each text message recognized by Paddle is cropped around its RoI, provided by the inference of the model together with the raw text. Each pixel inside the RoI is mapped in HSV (Hue, Saturation, Value) color space, shown in Figure 2. Each existing color for analysis in digital imaging can be visualized as a point inside a different color space. In this phase, mapping the color in HSV simplifies the process of isolating bright areas corresponding to text and filtering out the darker background by cutting the lower part of the pie-chart representation. When the bright text has been isolated, the average color of the pixels of text is measured in RGB cube color space, shown in Figure 3. In this model, each axis corresponds to one of the primary colors: red (x), green (y), and blue (z). Each pixel of text is evaluated in its R, G and B component, to calculate the R, G an B mean value of the text. The Euclidean distance between the obtained point coordinates and the predefined colors at the vertices is even-

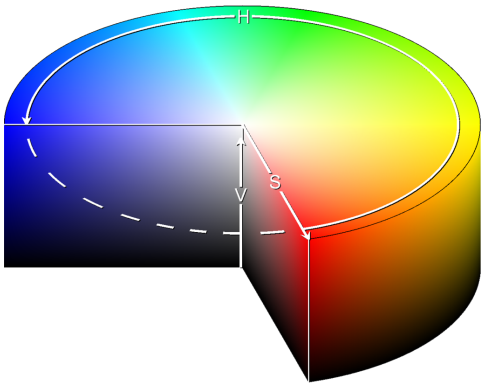

Figure 2. HSV color model. Credits: Wikipedia.

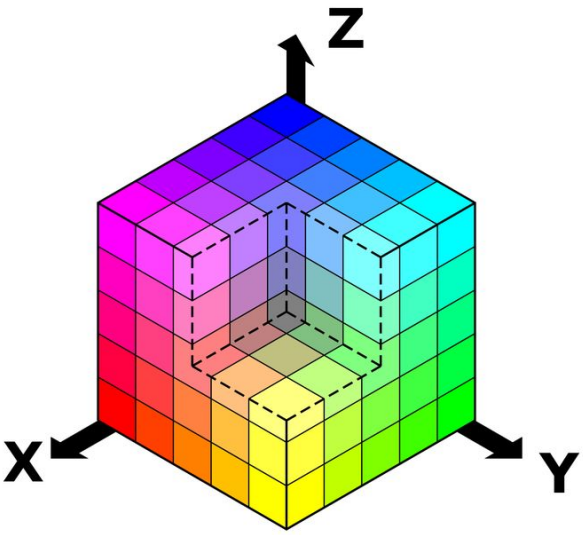

Figure 3. RGB color model. Credits: Wikipedia

tually calculated to determine the closest match with red, yellow, cyan, and white. The color extraction reduces the length of the dictionaries, so optimize the search speed and increase also the probability for Fuzzywuzzy to find the correct text.

Finally, the RegEx library has been used to correct misread characters by replacing them with appropriate alternatives. For instance, in scenarios where it is crucial to recognize altitude numbers, PaddleOCR should ideally detect only numeric digits and not alphabetic characters. However, if the system erroneously reads an 'O' instead of a '0', the library can be employed to substitute the 'O' with a '0' and vice versa in other contexts. Additionally, it is also possible to exclude characters that are not expected to appear on the screens or within a RoI, such as special characters or accented letters. This method ensures a cleaner and more accurate recognition of text by preemptively filtering out erroneous characters based on predefined criteria. Note that a sharp solution must be applied with caution avoiding cases where a dropout of a character leads to wrong a test outcome.

Computer Vision Software Recognition Strategy

In Figure 4 the consequential steps that compose the CV software implementation generically discussed in the previous paragraphs are summarized in a flowchart. The CV software logic provides that at the activation of the system, a camera starts to stream. The complete analysis of a frame consists of two steps. The first obliged passage is the inference via OD model to detect the classes of interest. The optional second step is the inference via OCR model to read possible texts in the areas of the items in exam. In the example shown in the flowchart, the logic for a test on the button "AP 1" ON state is taken into account. The frame is given in input to the fine-tuned YOLOv8n model, which has been previously trained to detect generic buttons in ON and OFF states. The model accurately identifies the RoI of five buttons in OFF state and of one button in ON state, along with their respective confidence scores for belonging to the corresponding class. In this test, the OCR step is required because it is mandatory to understand whether it is indeed "AP 1" the button in the ON state. The RoI of the button ON item is used as a reference to crop the frame. In case of noise, the cropped frame can be previously pre-processed to increase the sharpness or modify the exposition. The cropped frame is given in input to the PaddleOCR model without fine-tuning. The raw text read by Paddle is then compared with a list of possible button texts, and "AP 1" is finally associated with the coordinates of the button ON RoI. This information is given back to the framework, that receives the success of the test: "AP 1" is activated. This logic can be extended to each component of each SUT inside a cockpit. In addition, dedicated functions can be developed to satisfy specific tasks.

While in post-flight analysis for flight tests, it is sufficient to have only a robust CV framework for video study, in unit tests and system integration tests, it would be convenient to automate also the necessary actions on the SUT to prepare the environment for the visual tests. For instance, in the flowchart provided, it would be advantageous to include and automate the action of pressing the "AP 1" button before initiating the visual test. To fully automate the testing procedures a system that pairs CV algorithms and a collaborative robot has been studied and implemented. ARTO system, as previously documented in the literature (Refs. 13, 14), has already been demonstrated to be a robust framework integrating various subsystems within an aircraft thanks to a multiple-system architecture. In this paper, the capabilities of the ARTO system are further explored by focusing on its real-time CV system.

IMPLEMENTATION AND RESULTS

Unit Test

Unit testing is the first crucial phase in the validation process for the aeronautical industry, focusing on the verification of individual components or subsystems' operation. While essential, this type of testing is composed of elementary steps,

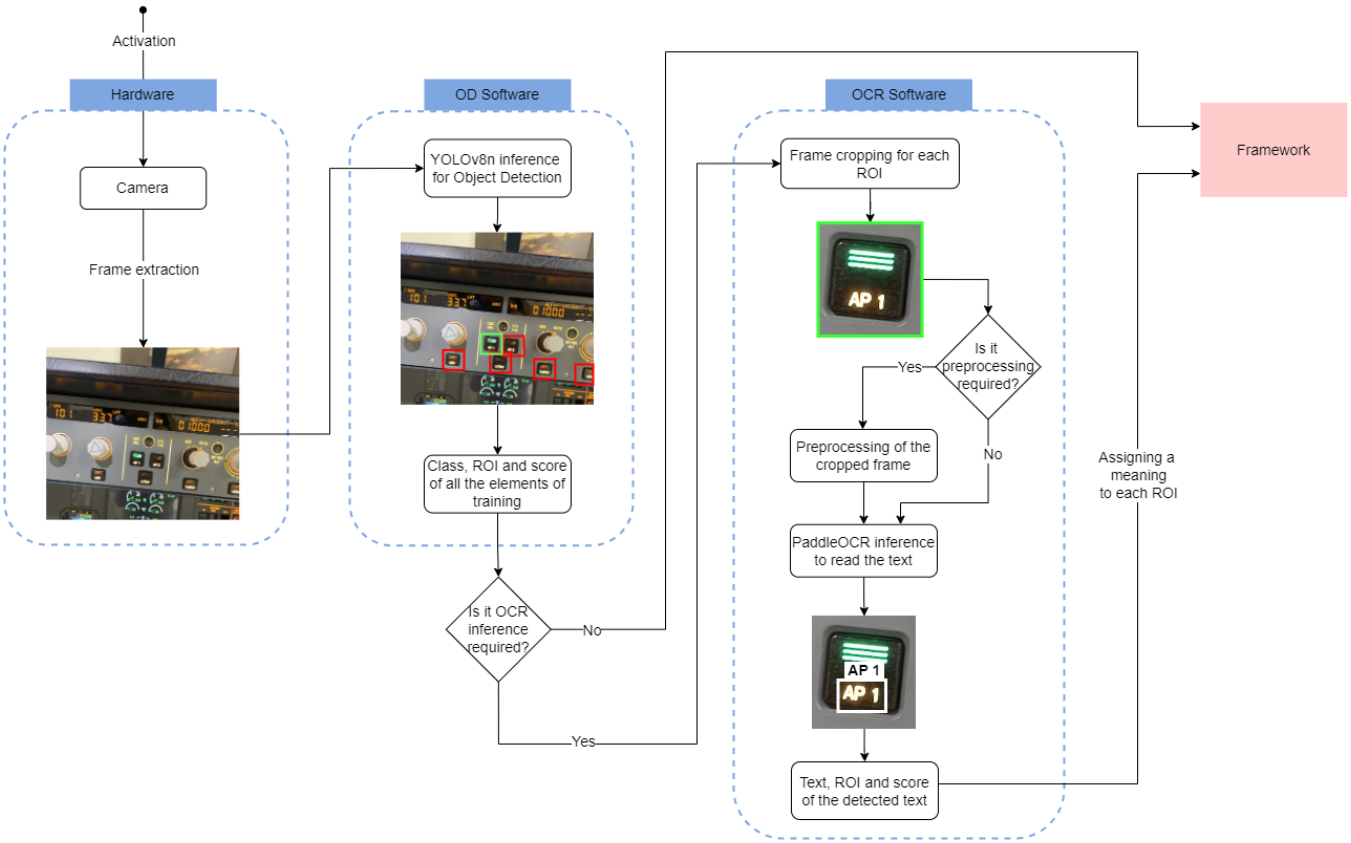

Figure 4. CV software flowchart. Credits: TXT E-Tech.

often repetitive and monotonous, making it a tedious but necessary part of the development process. The particular meticulous attention to detail required in unit testing ensures however the overall quality and functionality of the systems in further phases (system integration and flight tests). Taking into account unit tests on systems such as MCDU for civil aircrafts or UFCP in military aircrafts, it is evident that engineers face a significant workload due to the numerous visual inspections required. Automating the visual check process would significantly reduce the time testing engineers spend on these operations. A practical example is here provided on an A320 cockpit simulator, a standard civil aircraft, taking into account a possible unit test on MCDU SUT in Figure 5.

Firstly, a high-resolution RGB camera must located about perpendicularly to the SUT, enabling Computer Vision to perform inference checks. The a2A4508-6gcBAS Basler ace 2 R, chosen for this aim, is a high-performance industrial camera designed for demanding image processing tasks. Its compact dimensions ($48.9 \times 29 \times 29$ mm without lens mount and connectors) and lightweight (< 100 g) make it suitable for space-constrained applications like cockpits. Paired with the premium Basler C11-3520-12M-P C-mount lens, optimized for high contrast and low distortion, this setup is ideal for precise imaging applications. The combination ensures high resolution and clarity, crucial for developing CV software aimed at automating unit tests, and providing accurate detection and

Figure 5. A320's Multi-Function Control and Display Unit. Credits: TXT E-Tech.

analysis of small components or text on screens. Given the high resolution of the camera, this latter can be positioned also at a considerable distance from the object. By using its RoI functionalities, it is then possible to crop the image around the SUT of interest, or around specific parts of the SUT of interest. The main technical specifications are reported in Table 3.

Table 3. Basler camera and lens specifications (Ref. 15).

Camera Component	Specifications
Resolution	18.4 MP
Pixel size	$2.5 \times 2.5 \mu$m
Frame rate	up to 6.3 fps
Lens Component	**Specifications**
Fixed focal length	35 mm
Aperture range	from F2.0 to F16
Resolution	up to 12 MP

Once the camera is properly set, testing engineers, or potentially a robotic arm, can perform the necessary gesture sequences to initiate the visual tests. The CV system can support the engineers by automating the visual checks. A possible unit test on MCDU is reported in Table 4, with a focus on step 3 shown in Figure 6.

Table 4. Unit test on MCDU

#Step	Gesture	Visual check
1	Press BRT button	Check RDY light is turning OFF Check MCDU MENU appears on the display
2	Press DATA button	Check DATA INDEX 1/2 appears on the display as page title Check GPS (small font) <MONITOR (large font) appears in correspondence of LSK3
3	Press LSK3 button	Check both GPS are showing the same information Check both GPS are showing correct information about the aircraft position

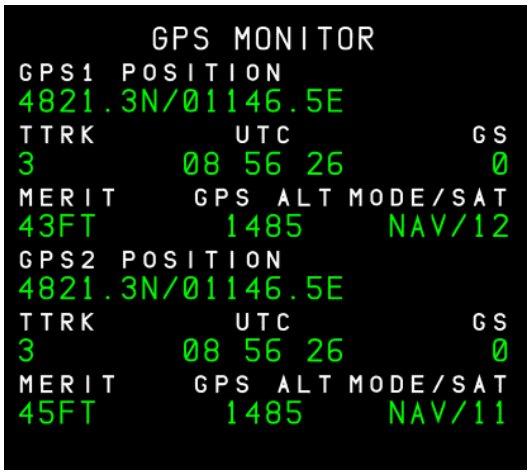

Figure 6. GPS monitor. Credits: FlyByWireSim.

To automatically perform step 1 of the visual test sequence YOLOv8n model has been fine-tuned to recognize the indicator light off for the signals in the upper part of the MCDU. In addition, Paddle has been intensely used to capture all the texts that must be recognized (steps 1, 2, 3). For the checks corresponding to step 3, the information about the position of the texts is crucial. For each text extracted by the Paddle model, there must be a correspondence between the upper part of the display and the lower one, except for some numbers (i.e. "GPS1", "GPS2", "NAV/12", "NAV/11"). Further algorithms of comparison have been customized for this task. In addition, the color extraction functionality explained in the "Object Detection and OCR Implementation for Real-Time Cockpit Analysis" paragraph is crucial to perform a complete check.

The performances obtained by the use of PaddleOCR are overall satisfactory. To evaluate the algorithm's goodness, a study on the Character Error Rate has been conducted. CER calculation (Equation 1) is based on the Levenshtein distance concept, by measuring the minimum number of operations required on the characters T to transform the detected text into the real one. In this formula, C corresponds to the total number of characters involved.

$$\text{CER} = \frac{T}{T+C} \times 100 \tag{1}$$

The lower the CER, the higher the accuracy, and the better the OCR performances without the use of correcting dictionaries. In the context of this test for the MCDU, in a controlled environment with a fixed position of the camera pointing at the display, without changing the condition of luminosity, and with few texts that have to be recognized, the CER reached is $\sim 3\%$, with an accuracy slightly below 97%. The most common errors are missing spaces, which is the main reason why the word error rate has not been considered to evaluate the model's performance. The accuracy reaches virtually the outstanding result of 100% after the application of the correction algorithm by the use of a known dictionary containing all the possible texts displayed.

System Integration Test

System Integration Tests are similar to Unit Tests, but they integrate simultaneously different SUT. The difficulty increases for OD since the models must be trained to more classes of interest, and OCR will be used to detect different texts, numbers, letters, and symbols. For this aim, the CV system can be composed of a changing number of cameras, depending on the test environment. High-resolution RGB cameras (a2A4508-6gcBAS Basler ace 2 R camera) are used to point at specific modules during the system integration tests, performing inference on the framed element. As a supplement, a "mobile" camera can be installed on the robotic arm used to automate the gestures. The stereoscopic RealSense Intel D435i has been chosen for this purpose, whose main technical specifications are reported in Table 5.

The RGB frames are overlapped on the depth output through a pixel under-sampling. The 3D functionality is crucial to map the cockpit space at runtime, visualizing the environment as a point cloud defining the depth between each component and the robot's TCP. Moreover, by a superposition of z pixels with RGB pixels, it is possible to perform CV inference with fine-tuned models for OD and to know the coordinates on the 3D space of each item of interest. Each SUT can be recognized, while obstacles can be mapped in the space and detected as

Table 5. RealSense camera specifications (Ref. 16).

Depth Component	Specifications
Resolution	up to 1 MP
Field of view	$87^\circ \times 58^\circ$
Frame rate	up to 90 fps
RGB Component	**Specifications**
Resolution	up to 2 MP
Field of view	$69^\circ \times 42^\circ$
Frame rate	up to 30 fps

elements to avoid. Going into detail, each component of different SUT such as buttons, switches, and levers is recognized and mapped in spherical coordinates, with the robot base as the system's center. Further algorithms are yet taken into account to optimize the robotic gestures and make them updated in real-time (Ref. 17). ARTO's CV system has been already implemented in new prototype simulator environments for both aircrafts and helicopters. For research purposes, the results here presented have been obtained in an A320 simulator cockpit, in Figure 7.

Figure 7. Arto system on A320 cockpit. Credits: TXT E-tech

To show the capabilities of the developed system in a system integration test a well-known procedure by pilots and engineers has been chosen: the landing procedure with autopilot engaged system. This not only demonstrates the versatility and adaptability of the ARTO CV system but also provides a detailed examination of its application in a different aircraft environment, showcasing the system's ability to manage and coordinate complex operations effectively. This comprehensive assessment highlights the ARTO's CV system's potential for enhancing flight safety and efficiency through rigorous system integration testing. The CV checks during the landing procedure consist of 10 passages, 6 performed via Object Detection and 4 via OCR, shown in Table 6. To perform the landing, the fixed camera is pointed at the PFD instrument.

To perform the OD checks, the YOLOv8n model has been

Table 6. CV procedure steps.

#Step	Visual Check	OD/OCR	Camera Type
1	Check if the distance to the airstrip on the MCDU is equal or below 21 nm	OCR	Mobile
2	Verify if the AUTO BRK MED button is on	OD	Mobile
3	Check if the speed has reached the correct value on PFD to put the Flaps lever at 2	OCR	Fixed
4	Verify if the LS button is ON on the EFIS panel	OD $\rightarrow$ OCR	Mobile
5	Verify if the APPR button is ON on the FCU panel	OD $\rightarrow$ OCR	Mobile
6	Verify if the GS pink rhombus in one bar above the reference on PFD before put the Flaps lever to 3	OD	Fixed
7	Verify if G/S is engaged on PFD	OCR	Fixed
8	Verify if the LDG gear lights are GREEN	OD	Mobile
9	Verify if the TERR button near the ND is ON	OD	Mobile
10	Verify if the altitude has reached 200 ft on PFD	OCR	Fixed

fine-tuned to obtain two different sets of weights, one for the mobile camera, observing during the landing procedure different mechanical modules (MCDU, Landing Gear Panel, EFIS, FCU) and one for the fixed camera, constantly pointing at the Primary Flight Display. The classes of interest to perform OD are reported in Table 7. A set of $\sim$ 1800 photos has been collected for training, validating, and testing the mobile camera model, while a set of $\sim$ 600 photos has been sufficient for the fixed camera model. Both the dataset have been split for this purpose in 70% for training set, 20% for validation set, and 10% for test set. The instances in each set are however not balanced: for the fixed camera model, PFD was displayed in each photo, so these instances are doubled compared to GS close and GS far. However, for the fixed camera model, detecting the PFD is not currently necessary, but is performed in anticipation of potential future requirements. Generic button OFF instances are several more than generic button ON because on the EFIS and FCU panels, most of the buttons are mutually exclusive in the ON state. The labeling of the images has been performed by using Computer Vision Annotation Tool (CVAT, (Ref. 18)).

Table 7. Classes of interest for the training of both models.

Mobile camera model	Occurrences	Fixed camera model	Occurrences
Landing Gear Lights ON	543	PFD Display	594
Generic button ON	662	GS close	297
Generic button OFF	2711	GS far	297
Autobrake button ON	583		
Terrain button ON	504		

The evaluation of the CV models on the test datasets yielded several significant insights. Both models demonstrate robust performance across various metrics, with high recall, precision, and F1 scores. The confusion matrices (Figures 8 and 9) highlight areas where each model excels and areas where further improvement is needed (especially for the mobile camera model in Figure 8). Improvements could be made to better distinguish the generic ON and OFF button classes (for Button OFF class, the model has a slightly lower precision, around 97.0%, and F1 score, of 98.5%). The low false positive rate suggests that both models are well-suited for practical

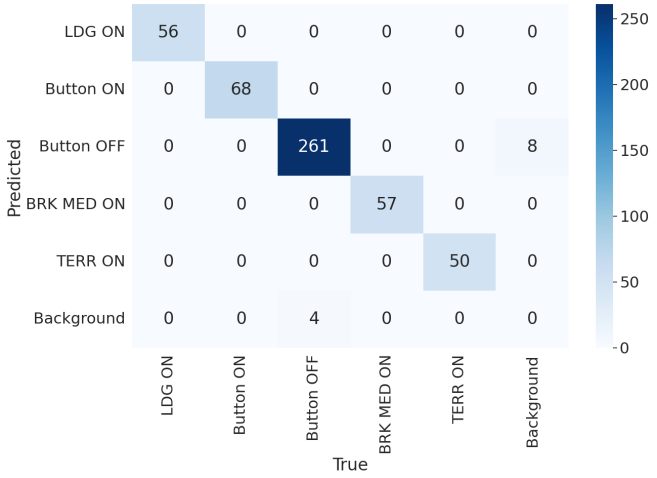

Figure 8. Confusion matrix on test set of mobile camera model for the 5 classes of interest.

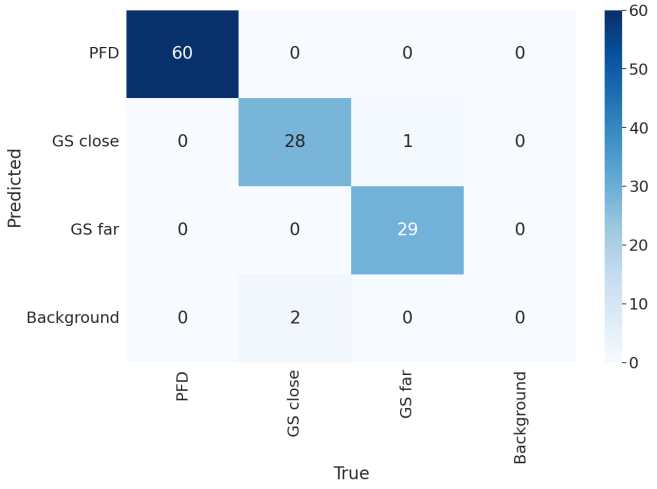

Figure 9. Confusion matrix on test set of fixed camera model for the 3 classes of interest.

applications in the aerospace test domain, where accurate and reliable real-time detection of specific features is critical.

To perform only OCR steps with the fixed or mobile cameras, a crop on the region of interest is done a priori in real-time framing, to isolate the frame areas where Paddle can do inference. Paddle model is then applied only on the cropped area to read numbers (steps 1, 10) or texts (steps 3, 7). Eventually, for the steps where OD and OCR are required one after the other, the process previously explained with the flowchart in Figure 4 has been used. As attending, compared to the performance of PaddleOCR observed in Unit Tests, the Character Error Rate (CER) increased from 3% to 10% during System Integration Tests. This sensitive difference is attributed to varying distances between the mobile camera and the SUT, along with differences in the brightness of the items being read and the movement of some items on the displays. To address these challenges, sharper solutions with RegEx library were implemented.

In all the discussed cases, the redundancy of the inference iterating on 8 frames per second for both cameras provides additional robustness for the algorithm. Eventually, each visual check in the test has a failure tolerance over a certain number of iterations. This is because the feature being examined may be detected by the computer vision algorithm during the system integration test only a few seconds after the test begins (i.e. step 3: check if speed has reached the right value on PFD).

From a qualitative point of view, the OD fine-tuning of the models and the OCR algorithm involved, plus the precautions taken to match the raw text detected with the correct words and numbers (see Object Detection and OCR Implementation for Real-Time Cockpit Analysis paragraph) led to great results. ARTO has the capabilities to perform correctly and in loop sequence, a real-time landing procedure with autopilot engaged with high levels of precision and reliability.

Post-flight Test Analysis

The integration of CV technologies also offers significant benefits in post-flight test analysis. Computer Vision can be extraordinarily supportive of automating the detection of various flight parameters, structural conditions, and system behaviors, enhancing accuracy and efficiency, in the study of on-board recorded videos. For this aim, CV software has been developed to detect flight test "inconveniences" such as warning flags, and error messages due to aircraft particular flight conditions captured in video recorded inside new prototypes, and return their track in time. The cockpit taken in exam mainly consists of three wide-format touchscreens. A fixed camera, set above the pilot's seat, records simultaneously the Pilot's PFD and the central Multifunction Display. Since the information on the Copilot's PFD is redundant, neither is it considered nor recorded. For this type of analysis, the inference is not performed in real-time; instead, the algorithm processes video footage that has been previously recorded during flight tests or ground tests.

YOLOv8n model has been fine-tuned to recognize 16 classes of interest, reported in Table 8 with their instances. The dataset used for training (70%), validation (20%), and test (10%) has been obtained from more than 5 days of recorded flight tests, ground tests, and simulator tests, from which $\sim$ 4700 screenshots have been extracted, capturing the items of interest. Since all the classes to be recognized are elements that appear on the screen, also in this case the instances in the dataset are not balanced: there is a markedly higher number of occurrences for the display ON compared to the other classes, some items have a poor statistics (i.e. Alert Message 5). The labeling of the images has been performed by using Computer Vision Annotation Tool (CVAT, (Ref. 18)).

The evaluation of the CV algorithm on the test dataset yielded significant results also for this more complex case. Firstly, the imbalance of the dataset is well handled by the model. The confusion matrix (Figure 10) is heavily dominated by the diagonal values, indicating that the model has correctly clas-

Table 8. Classes of interest for the dataset of flight test inconvenience model.

Classes FTI model	Occurrences	OD/OD→OCR
Display OFF	387	OD
Display ON	8994	OD
CAS open	1770	OD→OCR
CAS closed	2845	OD→OCR
CAS empty	1680	OD
Red flag	2854	OD→OCR
Yellow flag	1405	OD→OCR
Alert Message 1	377	OD
Alert Message 2	394	OD
Alert Message 3	282	OD
Alert Message 4	232	OD
Alert Message 5	18	OD
Alert Message 6	461	OD
Alert Message 7	44	OD
Fail 1	670	OD
Fail 2	488	OD

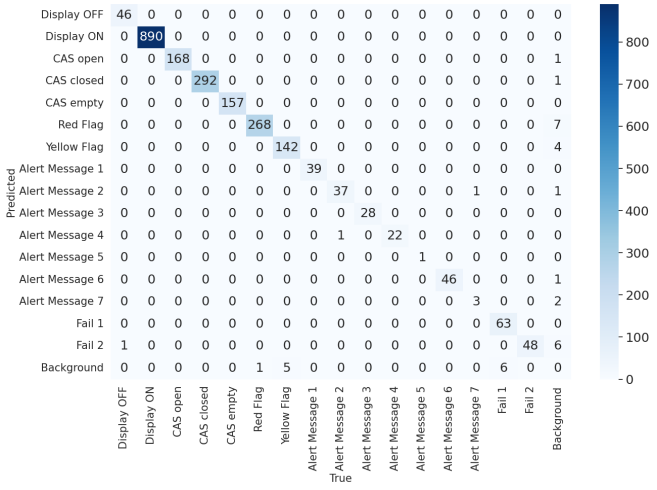

Figure 10. Confusion matrix on the test set for the 16 classes of interest for FTI model.

sified most instances for each class. There are very few misclassifications, suggesting the high overall accuracy and robustness of the model. For instance, "Alert message 2" was misclassified as "Alert message 4" once, and "Display OFF" as "Fail 2" once. These misclassifications are minimal compared to the total number of predictions and do not impact the overall good model's performance.

The developed software, provided with a Graphical User Interface, allows users to select a video to analyze. The software processes the chosen video iterating over the frames. YOLOv8n performs inference on each of them, to detect the discussed items. A first check is made on the number of recognized screens, in the ON or OFF state, always requiring the total to be two; otherwise, the frame is discarded. This ensures accuracy when the view might be obstructed, such as by the pilot's head or variations in ambient brightness. In these cases, there would be no point in continuing with the further algorithm steps. If two screens are correctly detected, the information about the presence of the other classes of interest is saved on variables. Since the camera used in this context records 30 frames per second, every 30 frames the global information collected is condensed in a single output: if a feature (i.e. "Alert Message 1") has been detected in $\geq 30\%$ of the analyzed frames, it is considered present during the analyzed second. This information is saved with an associated confidence score equal to the fraction of frames in which the item has been detected over 30 frames per second. This way, the software can provide a more comprehensive and accurate representation of the observed events, reducing noise and potential errors associated with analyzing individual frames.

A crucial part of this algorithm is then represented by the application of PaddleOCR on some items recognized via the YOLOv8n model. In particular, on CAS open, CAS closed, Red Flag, and Yellow Flag classes, as reported in Table 8. The region containing the CAS messages can be empty, if no error messages are displayed, or is identified as CAS closed if at least one text message is present. Warning or error messages are displayed in color code (red, yellow, cyan, and white) depending on the gravity of the condition. The CAS closed can be expanded by the pilot to let the crew read more messages. In this situation, a CAS open item is displayed on the screen beside the CAS closed region. If OD inference detects CAS open or closed items, their RoI is cropped and given as an input frame to the PaddleOCR model as discussed in previous paragraphs. Paddle returns the RoI of each detected message and the raw read text. Thanks to the algorithms of color extraction, red, yellow, cyan, and white are ranked based on their probability of matching for each detected text and saved on a variable (i.e. "Fnglne Fall" is associated with the dictionary list [yellow, white, cyan, red] in this order). The lists of all the CAS messages that can be displayed divided by color are used in order of probability as a dictionary to match the correct texts and track their presence on the video timeline. In particular, the first message in color rank plus alphanumeric order that reaches the threshold of similarity $\geq 70\%$ is selected as the one that appeared on the display. In addition, given the experimented difficulty of reading red messages in low light environment conditions, even for human eyes, despite their primary importance for gravity color code, the following strategy is applied. If OD detects a CAS closed, color extraction detects a probable red CAS message, but OCR is not able to match the found text with any red text in the dictionary, the information is saved anyway, even if there is no match with a specific text. The output is collected in a separate variable called "detected but not recognized", giving the testing engineers the suggestion about the presence of a major error in a specific time window. For the other colors, as long as there is good visibility inside the cockpit, this problem has not been reported for the greater brightness of these texts. Note that CAS empty class is defined only to speed up the computational time since no inference with Paddle is applied to this region. For Yellow and Red Flag classes the concept is the same applied for CAS windows. If OD detects a Yellow or Red Flag in the inference of a frame, the RoI is

used to crop the frame around the object and OCR is used to read its contents. If there is no match with a specific text, the output is collected in a separate variable called "detected but not recognized".

To validate the results obtained with the developed CV algorithm, flight data extracted from the aircraft systems during flight tests recorded were used as True occurrences. In particular, the test has been conducted on the detected CAS messages to study the robustness of the character recognition algorithm (Paddle + matching algorithm). In this context, applying the matching algorithm is essential due to the high noise level in the frames recorded. The CER value is strongly correlated with the brightness inside the cockpit and can reach $\sim 40\%$ due to bad luminosity conditions. In Figure 11 the confusion matrix for a long in-flight video is reported as an example. In the discussed experimental setting, a higher tolerance for False Positives, which are however contained (1653), over False Negatives (1043), have been prioritized. This choice reflects a cautious strategy of giving more importance to tracking potential errors during flight tests that might not actually be present, rather than risking the oversight of critical flight events. The True Negative occurrences (107712), which are usually not discussed in OCR algorithm performances, are here composed as follows: if in a few seconds the OCR algorithm finds "Cabin Call" message displayed in white, but the flight data have not registered its presence. In those seconds "Cabin Call" represents a False Positive, but all the other seconds where "Cabin Call" has not been detected while not present, are counted as True Negative. That is the reason why the lower right square results so largely populated.

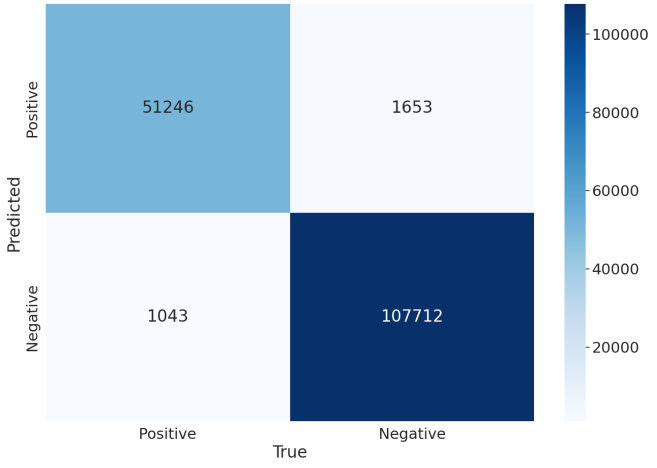

Figure 11. Confusion matrix on the test set for the CAS messages detection for FTI model.

With an accuracy of 98.3%, the model performs exceptionally well in correctly identifying CAS messages when they are displayed and accurately recognizing when they are not. The high precision of 96.8% and recall of 98.0% indicate that the algorithm is highly effective at correctly identifying both the presence and absence of CAS messages while minimizing False Positives and False Negatives. Eventually, the F1 score of 97.4% reflects a well-balanced performance between precision and recall. Overall, the algorithm demonstrates excellent performance across all key metrics, making it highly reliable for accurately monitoring CAS messages in this context. Moreover, with a precision error far $\leq 5\%$, the implemented algorithm has an error threshold that goes below the human ones due to the attention threshold dropping after hours of repetitive work.

CONCLUSIONS

The AI-Driven Computer Vision software has been correctly implemented and tested in three different cases: unit tests, system integration tests, and post-flight data analysis. The results obtained are here summarized:

1. Unit testing: the best performances have been obtained in this phase, with an OCR accuracy pre-dictionary correction of 97%, and after correction of nearly 100%. This is due to the simpler conditions of the tests: a more static test environment, only one system under test in analysis, shorter dictionaries of matching, and fewer classes of training for object detection. The emerged limits are surely the necessity of using a matching dictionary, still fundamental in this first approach without OCR fine-tuning. Without the matching algorithm, the CER error would overtake 3% in this context, and worsen brutally in the system integration tests and post-flight analysis. On this limitations are based further developments.

2. System Integration Testing: in this testing phase the difficulty level increases because two or more systems under test are involved during the testing phase. There are more visual objects and features to be recognized by OD model and different texts that have to be identified by OCR model, increasing the length of the dictionary for the matching algorithm. On the other hand, the experimental setup is still in a laboratory environment, where the luminosity conditions are stable, and the variability of the data is contained. This leads to great performances: for most of the classes of training the model achieves perfect precision, recall, accuracy, and F1 scores. Only for Button OFF class, the model has a slightly lower precision and F1 score (around 97.0% precision and 98.5% F1 score), but it does not significantly detract from the overall reliability of the model. In the System Integration Test, further developments will provide a generic OD model tuned to the recognition of most of the mechanical elements of SUT and the main symbologies appearing on the displays.

3. Post-flight Data Analysis: in this context, the number of classes to be detected is further increased and the elements appearing in the same frame at the same time are much more. The flight tests have been recorded inside real aircraft, so the environment is less stable: the frame changes rapidly luminosity, and the ambient conditions can be really noisy, too dark, or too bright (OCR's

CER can brutally increase without dictionary corrections). The matching via dictionary is crucial to reduce the error rate, in this context a precision slightly below 97% is reached thanks to this expedient. With this kind of variable data for long videos, automating the recognition of items such as screens OFF, alerts, warning messages, and flags can cut down the analysis time. This latter is today equal to the length of the video (a video of 1 hour is analyzed in approximately $\sim$ 1 hour), while an operator actually takes three or four times as long due to the high number of information simultaneously displayed. With this CV software, during the automating analysis timeframe, the operators are not involved in the task, and can just monitor the work status while performing other duties. Despite the results achieved, the goal will be to further reduce the computational times with techniques yet taken under exam, and improve the accuracy level (98.3%) cutting down false positive and negative occurrences. In further development of the CV software, the possibility to apply cuts a priori in some frame portions to improve the precision and reduce the computational time will be taken into account. However, the position of the camera must remain fixed, pointing for all the video timelines the same SUT.

Further applications of the AI-driven computer vision software for aeronautical testing are today's cases of study. For instance, it is planned to apply it in the future also in the field of defect inspection, and for maintenance provisions in aeronautical and aerospace industry processes. In the future versions of the CV software the focus will be on the building of an even more reliable and robust system, generically trained to the recognition of all the main components in SUT inside different cockpit models. OCR model, which has already been demonstrated to perform brilliantly, will be further fine-tuned for the recognition of text appearing on the main displays, trying to obtain a more reliable algorithm.

Author contact:
Eleonora Barbano - eleonora.barbano@txtgroup.com
Valentina Maffei - valentina.maffei@txtgroup.com
David Frisini - david.frisini@txtgroup.com

REFERENCES

1. Sultana, S., Pramanik, A., Rahman, M,. S., "Lung Disease Classification Using Deep Learning Models from Chest X-ray Images", 3rd International Conference on Intelligent Communication and Computational Techniques (2023) pp. 1-7, DOI: 10.1109/ICCT56969.2023.10075968

2. Huilin, C., Qiyu, S, Fangfei, L., Yang, T., "Computer vision tasks for intelligent aerospace missions: An overview", *arXiv e-prints* (2024), DOI:10.48550/arXiv.2407.06513

3. Kim, Y. and Li, Y., "Human Activity Classification With Transmission and Reflection Coefficients of On-Body Antennas Through Deep Convolutional Neural Networks", *IEEE Transactions on Antennas and Propagation*, vol. 65, no. 5, pp.2764-2768, (2017), DOI: 10.1109/TAP.2017.2677918

4. Redmon, J., Kumar Divvala, S., Girshick, R.B., Farhadi, A., "You Only Look Once: Unified, Real-Time Object Detection", IEEE Conference on Computer Vision and Pattern Recognition (2016), pp.779-788.

5. Sohan, M., Sai Ram, T., Rami Reddy, "A Review on YOLOv8 and Its Advancements", Data Intelligence and Cognitive Informatics (ICDICI 2023). Algorithms for Intelligent Systems. Springer. DOI:10.1007/978-981-99-7962-2_39.

6. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R., "Intriguing properties of neural networks", arXiv preprint arXiv:1405.0312, 2014. DOI:10.48550/arXiv.1405.0312.

7. Hoffstaetter, S., (2023, Version 0.3.13) Python Tesseract. GitHub: https://github.com/madmaze/pytesseract?tab=readme-ov-file.

8. PaddleOCR. Github: https://github.com/PaddlePaddle/PaddleOCR/blob/main/README_en.md.

9. Kittinaradorn, R. (2023, Version 1.7.9) Easy OCR. GitHub: https://github.com/JaidedAI/EasyOCR.

10. Morales, F., (2020, Version 0.8.4) Keras OCR. GitHub: https://github.com/faustomorales/keras-ocr.

11. Fuzzywuzzy library. Adam, C., https://pypi.org/project/fuzzywuzzy/

12. Levenshtein library. Bachmann, M., https://pypi.org/project/Levenshtein/

13. Frisini, D., Taumaturgo, V., Giulianini, G., Lucini Paioni, C., Morlacchi, G., Poltronieri, F., "Technology concept of an automated system for integration testing", *CEAS Aeronaut J* (2024), DOI: 10.1007/s13272-023-00709-3.

14. Frisini, D., Giulianini. G., Romano, M., Zonzini, N., Rinaldi, G., Taumaturgo, V., "Evaluation of an Automatic System for Cockpit Integration Testing", Vertical Flight Society 79th Annual Forum & Technology Display, (2023), pp. 1-22, DOI:10.4050/F-0079-2023-18012.

15. Basler Documentation on the official site: https://docs.baslerweb.com/a2a4508-6gcbas https://docs.baslerweb.com/c11-3520-12m-p

16. Realsense Documentation on the official site: https://www.intelrealsense.com/depth-camera-d435i/

17. Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Gonzalez Arenas, M., Gopalakrishnan, K., Han, K., Hausman, K., Herzog, A., Hsu, J., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, L., Lee, T.W.E., Levine, S., Lu, Y., Michalewski, H., Mordatch, I., Pertsch, K., Rao, K., Reymann, K., Ryoo, M., Salazar, G., Sanketi, P., Sermanet, P., Singh, J., Singh, A., Soricut, R., Tran, H., Vanhoucke, V., Vuong, Q., Wahid, A., Welker, S., Wohlhart, P., Wu, J., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., Zitkovich, B., "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control", *arXiv preprint*, arXiv:2307.15818, (2023)

18. CVAT.ai Corporation, "Computer Vision Annotation Tool, https://github.com/cvat-ai/cvat, DOI: 10.5281/zenodo.4009388, version 2.8.2