ÇANKAYA UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

PROJECT REPORT

CENG 408

INOVATİVE SYSTEM AND DEVELOPMENT II

AI BASED COCKPIT SYBOLOGY IDENTIFICATION SYSTEM

TEAM ID:2021-17

İsmail Emre CANITEZ 201711009

Mustafa Tahir BİNGÖL 201711007

Nisa BÜYÜKNALBANT 201811015

Seyit Semih YİĞİTARSLAN 201726081

Ümit KUMAŞ 201726047

ADVISOR:

GÜL TOKDEMİR

CONTENTS

Abstract

This document covers the Project Plan, Literature Review, Software Specification Requirements and the Software Design Document of the project “ARTIFICIAL INTELLIGENCE BASED COCKPIT SYMBOLOGY IDENTIFICATION SYSTEM”. In this project we aim to create an Artificial Intelligence module to automate the test process of the cockpit panel. We expect the module to identify and label the shown symbols on the panel and calculate an error percentage by comparing the expected value and the identified value. We believe automating this process is crucial because of the high sensitive and the importance of the shown values on the panel; especially in industries like aerospace.

1. Introduction

Panels of aircraft cockpits contain multiple types of sensors. Situational and navigational information and the current state of the sensors are all displayed on the display devices of the cockpit. The values shown on these panels all have high importance and sensitivity. Reading a value even slightly wrong can cause big problems going onwards. And yet the test processes of these devices are done manually. Most of the time human eye is not enough to detect small inconveniences. Automating this process would be beneficial for both the accuracies of panels as well as the time being spent on the test processes. The artificial module to be developed will process the image of the given panel and label the sensor areas. After that it will detect the shown value of the current state of the panel and assign a value to the given label of the specific sensor. Comparing the results we expect and the given results, an error percentage will be calculated. After training the module this way we will be able to have an idea on the accuracy of the any given panel.

2. Literature Review

2.1 Abstract

Safety plays a big role in industries like Aerospace. That is why the role of verification and validation are also as important. Display panels of the cockpit consist of multiple objects (symbols shown in the panel). Manual verification and tests are error-prone and time-consuming activities. Error rate calculations of the shown symbols and the characters on the cockpit panel are done manually. We aim to reduce the existing error rate by developing an AI Module for symbology and character identification (Optical Character Recognition, OCR) then integrate this module into the testing process so that the unnoticed errors and the sensitive values which can not be detected by the human eye could be detected and the needed cautions can be taken beforehand. Our approach includes the study on deep learning-based graphics symbol detection. We aim to reduce the error rate of identification of symbols in our project.

2.2 Introduction

Aircrafts contain many types of sensors. Situational and navigational information and the current state of the sensors are all displayed on the display devices, panels, of the cockpit. Depending on these symbols many actions are taken and decisions get made, due to that, the correctness and the accuracy of the shown symbols are highly important. The tests for these display screens, should have a wide range and should be dependent on sensitive parameters in order to minimize the error rate.
When the testing, identification and the validations of the shown symbols are done manually; the process gets complicated, time consuming and prone to have a higher error rate. The sensitivity of the values are critical and the symbols move and change rapidly due to the nature of the aircrafts. The human eye cannot detect multiple responses and might miss to detect an existing error/issue. Object detection could be highly beneficial in these cases.
An AI Module that identifies the symbols and characters then labels them will be developed for this project. Detected symbols will be open to be identified, classified and be labeled. In order to identify the symbols and the characters, we plan to benefit from the algorithms such as R-CNN[1], Fast R-CNN[2], Faster R-CNN[3], SSD[4], YOLO[5] and techniques of Optical Character Recognition (OCR).

2.3 Related Works

There are only a few works done/ongoing projects on the topic of identification and verification of the symbols which are displayed on the display devices, which we believe to be an important matter. We were able to examine the works which included similar technologies as the ones we mean to use. We were able to find a work which is highly similar to our project which is called “Cockpit Display Graphics Symbol Detection for Software Verification Using Deep Learning” [7]. This project has a similar goal but mainly focuses on the dynamic changes of the symbols and to have a higher catch-rate of the changing objects. Authors of this project touched upon algorithms such as; R-CNN [1], Fast R-CNN [2] and Faster R-CNN [3] for identification, classification and the object detection of dynamic symbols with a region proposal method. They suggest using SSD [4] and YOLO [5] for regression based methods which are more suitable for their work, due to these methods being applicable to a real-time environment. Authors of [7] proposed that SSD produces a higher accuracy over other algorithms with faster processing speed and in synchronization with the cockpit display device refresh rate. A dataset named “VOC 2007” was used for this project. Results were as such:

PERFORMANCE OF OBJECT DETECTION ALGORITHMS ON VOC 2007 DATASET

Object Detection Algorithm	*mAP*	FPS
Fast R-CNN	70.0	0.5
Faster R-CNN	73.2	7
SSD300	74.3	46
SSD500	76.8	19
YOLO	63.4	45
YOLOv2 (352x352)	73.7	81
YOLOv2 (544x544)	78.6	40

Figure 1: Performance of Object Detection Algorithms

Other than this project, we were not able to find projects which were this similar to ours. So we also examined the algorithms that can be applicable to our project and other projects that used these algorithms. We examined the document "Traffic Sign Detection Method of Improved SSD Based on Deep Learning" [8] in order to have a better grasp on SSD. They touch upon how the deep learning is a richer and a more capable method over hand-crafted designs including the features SIFT[9], Haar[10], HOG[11] and the classifiers SVM[12] and Adaboost[13]. Deep learning methods that identify 5 traffic signs can be categorized in two groups. First being the "Region-based Recommendation Model" and the other being the "Regression-based Recommendation Model". Authors explain the Region-based Recommendation Model as, "The region-based recommendation model firstly selects the recommended region, and performs feature extraction and classification on the recommended region, such as R-CNN, Fast R-CNN, Faster R-CNN, SPP-NET and so on.".[8]. Authors suggest that the advantage of the Region-based model is the higher accuracy that it provides while the disadvantage being the low success rate of real-time performance. Authors explain the Regression-based Recommendation Model as, "The regression-based model is used to detect the relationship between the default box, the ground truth box, and the predicted box, such as YOLO and SSD." [8]. They conclude that the SSD algorithm has a higher accuracy and real-time performance over YOLO, but the SSD fails to successfully detect small objects. Due to that, Authors aim to develop an improved version of the SSD algorithm. According to the feature of traffic signs, they adjusted the feature map scale and aspect ratio based on the SSD model, removing the high-layer abstract feature maps. Their dataset consists of 1532 images which include every weather condition (extreme ones as well).

1314 of these images belong to the training set and the rest belongs to the test set. There are a total of 2741 traffic signs in these pictures (turn right, turn left, no parking ,etc.).

Success rates of the project are given as:

TABLE DETECTION COMPARISON OF IMPROVED SSD AND CLASSIC SSD

Traffic signs	Training set	Test set	Classical SSD(AP)	Improved SSD(AP)
No left turn	93	11	0.7553	0.7871
No parking	111	16	0.3740	0.6232
Keep right	197	26	0.7810	0.8983
No motor vehicles	257	44	0.4564	0.6592
Speed limit 20km/h	842	144	0.7731	0.7751
No horn	853	147	0.7883	0.7738
total	2353	388	0.6547	0.7528

Figure 2:Comparison of Improved SSD and Classic SSD

We examined the document “An End-to-End Deep Neural Architecture For Optical Character Verification and Recognition In Retail Food Packaging” [14]. Authors suggests that an end-to-end deep neural network architecture for recognition of the use by date in food package photos. Authors also claim that the architecture consists of two networks. Optical Character Verification is made by the first network for OCR. Pre-processing operations and selection of candidate photos are made by the first network. This network acts as a filter. Images that are

pre-processed and selected by the first network, are used by the second network. Second network consists of Fully Convolutional Neural Network (FCN) which helps to recognize the use by date. Fully Convolutional Neural Network (FCN) is used to process the images locally. Multiple levels of attributes which are extracted from the image, are used to localise the use by date. Character recognition is made by the MSER algorithm. Stroke Width Transform (SWT) [15] and Maximally Stable Extremal Regions (MSER) [16] techniques are used to increase the detection of success. Dropout[17], data augmentation [18], and transfer learning (TL) [19] can be used to decrease overfitting.
Authors used two datasets that were taken from a food company. Datasets include 1404 and 6739 images with different colors and context. The images are categorized into five categories which are complete dates, missing day, missing month, no date and unreadable.

Accuracy of the project are given as:

Complete vs.	Dataset	Images	Accuracy %
Unreadable	1	645 vs 645	90.1
	2	2847 vs 2847	96.8
Partial/No Date	1	645 vs 444	89.3
	2	2954 vs 2954	95.9

Figure 3: Global Based Experiment Result With Network 1.

	Tested Clear Images	Accuracy %
Dataset 1	240	98
Dataset 2	482	97.10

Figure 4: Local Based Experiment Results With Network 2.

Another work we thought might concern ours is “Optical Character Recognition for Alphanumerical Character Verification in Video Frames” [20]. In this work they aim to identify and recognize texts from images via OCR method. Authors touch upon two different parts of the project being “Character Mapping” and “Character Verification”. OCR and OTE methods were used for this project. They followed 3 main steps in order to identify the text, the steps being; segregating the videos (images) into frames, detecting the frames that included text within and finally identifying and extracting the text. The frames which did not include a text were erased due to efficiency and a gain in performance. Authors used “Motion Analysis Method” to detect the frames which did not include text within. The frames with a text were labeled as “Character Candidate Region.” The detected frames were in RGB but due to a gain in efficiency they were converted into a gray-scaled frame. The algorithm which took a video as an input gave an editable word file as an output at the end. Authors claim that the OCR methods other than this document are limited with different font-style and font-sizes. They also claim that these methods are only capable of processing stable images (nomotion), so the work [20] aims to develop a method which can process/identify nonstable images as well.

The table below shows the success rates of a test-run that was done with 10 test subjects:

Sample images (with scene text and superimposed text)	Number of characters in an image	Total number of characters detected, recognized, and extracted			
		Text detection phase	Detection percentage (%)	Text recognition and extraction phase	Recognition and extraction percentage (%)
10.avi	14	14	100	12	85.71
5.avi	93	90	96.77	84	90.32
mg1.avi	41	41	100	40	97.56
s2.avi	110	110	100	101	91.81
2.avi	11	11	100	11	100
sample.avi	62	61	98.38	61	98.38
blank.jpg	10	10	100	10	100
txt.jpg	18	18	100	16	88.88
s3.jpg	61	60	98.36	54	88.52
6.avi	24	24	100	20	83.33

The overall detection percentage of OCR-based system for 10 sample images is 99.35 %
The overall recognition and extraction percentage of OCR-based system for 10 sample images is 92.45 %

Figure 5: The Overall Detection Percentage of OCR-Based System

Another work which include similar technologies with ours is NN-based handwriting character definition system for optical character recognition OCR “Optical Character Recognition Technique Algorithms” [20]
In addition to optical character recognition, the handwriting character recognition system is also trying to provide a high accuracy value. In the article, systems consisting of existing technologies are examined, compared and systematically examined. In the article, the character recognition system is divided into offline character recognition systems and online character recognition systems. In the article, the character recognition system is divided into offline character recognition systems and online character recognition systems. In the online character recognition system, hand movements and movements at the end of the pen are examined at the moment when the character is written and these movements are trying to be defined.
The accuracy of the HCR system is said to be limited to 90%. In character recognition systems, some languages have been researched less because they are used less than local languages, so that more accuracy values are being taken with more work for other languages. In addition to these operations, image processing and pattern recognition play a very important role in handwriting character recognition. In addition, using statistical information in this system, the distribution of systems in the image is collected in three main headings as follows; Partitioning in regions, profile generation and projections, distances and crossing. In addition, structural features and the structure of the characters, as well as features such as the number of horizontal, vertical lines and the number of crossed points extracted from the geometric shapes of the characters, are used in this study. In addition, frequency domains like Discrete
Fourier, Transformation (DFT), Discrete Cosine Transformation (DCT), Discrete Wavelet, Transformation (DWT), Gabor filtering, and Walsh-Hadamard transformation etc.
Features such as features are used at a high or low level. For different languages and

different designs, different methods are used in this way. In addition, Matrix Matching, Fuzzy Logic, Feature Extraction, Structural Analysis, Neural Networks are used according to the OCR characteristics of various techniques used in design. In the Neural Network, MLP, which is the most common multi-layer neural network model, attracts attention. Here we are trying to get the desired output by using historical data. The components of the OCR system are as follows. First comes Location and Segmentation. Segmentation finds the components of an image, prints the regions of the document in which it is located, and extracts the created graphics from the shapes. Pre-processing, on the other hand, is the elimination of noise ions caused by scanning processes and the conversion of digitized characters into smoother ones.
Another one is Post processing, which are two types of post processing, one of them Grouping, the other is Error-detection and correction.
OCR is an abbreviation of the word optics for this system for the recognition of handwritten characters. The main purpose of the OCR system is to try to simulate human reading systems.
Moreover, the article describes the structure of the algorithm used in the neural network part of the proposed algorithm in the winter. As a result, 100% accuracy is achieved with OCR in the neural network proposed in the article.

2.4 Conclusion

Automated identification and verification of displayed symbols is a very promising and a needed field as it can provide a significant reduction in manual effort, needed time and error rate. The project we have proposed aims to detect symbols, label the state of the detected symbol and compare the label with the expected response to generate an error percentage with the studies of deep learning and image processing. In the scope of this project, we will mainly focus on the detection and the identification of symbols first. There are many technologies and projects that were done by using these technologies as touched upon in this literature review. We believe removing the limitations of manual systems and having an automated identification/verification system, especially when the values are as sensitive as in aircrafts, is crucial. We aim to contribute to this automation process with our project.

3. Software Requirements Specification

3.1 Introduction

3.1.1 Purpose

The purpose of this document is to present a detailed description of the Artificial Intelligence Based Cockpit Symbology Identification System, which we are willing to develop. We intend this document to explain the overall aspects of the project, explain the features that are going to be used in the system and what the system will do. This document is intended for both the stakeholders and the developers of the system and will be proposed to the Cankaya University.

3.1.2 Scope

This software system will be an Artificial Intelligence based identification system that aims to detect, identify and label the symbols on the given panel. This system will be designed to be used in the test processes that are done on the cockpit display screens. By detecting minor changes that the human eye can not detect and calculating an error rate. More specifically, the Artificial Intelligence module will compare the wanted symbol depending on the given test case with the symbol that is being shown on the panel after identifying and labeling it. Then it will produce an error rate depending on the overall accuracy. We aim to have a significantly less error percentage due to given values on the cockpit panels being very sensitive and the possibility of the human eye not detecting minor changes being higher. Our purpose is to automate the testing process. The system will contain a database of various panels and will identify the parameters as Airspeed Indicator, Altimeter, Vertical Speed Indicator and etc.

3.1.3 Definitions

Software Requirements Specification --> A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document.

OCR --> Optical Character Recognition

AUROC --> Area Under the Receiver Operating Characteristics

3.1.4 Overview of Document

Software Requirements Specification document consists of 3 chapters. The first chapter is for the introduction and the main aspects of the project.

The second chapter, contains a general description of the project. The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product

3.2 Overall Description

3.2.1 Product Perspective

In this project, it is planned to define the shapes and characters shown on the Primary Flight Display (PFD) screens located in the cockpit. On the other hand, cockpit images are taken through a simulator and thus tested and the main goal is tried to be achieved.

3.2.1.1 Development Methodology

We are considering using scrum to develop the program we are going to do. In this way, while developing our project day by day, we will also improve the process from the test stage to the construction stage.

3.2.2 User Characteristic

There will be one user characteristic in the program we will use. When the engineer in the test unit who will use the program opens the program, the artificial intelligence model that will be made for the design in the cockpit will be run.

3.3 Requirements Specifications

3.3.1 External Interface Requirements

3.3.1.1 User Interfaces

Since this a cockpit symbology identification system to test the system, it should be observed that data on the displays whether are working properly or not. The system provides vital information like airspeed, attitude, slip/skid, altimeter, vertical speed, horizontal situation and turn rate. Any undesirable error on the display should be indicated.

3.3.1.2 Hardware Interfaces

Not applicable

3.3.1.3 Software Interfaces

Programming language python is mainly used for this project. Falcon Bms 4.35 and Microsoft Flight Simulator will be used to collect the dataset. Falcon Bms 4.35 and Microsoft Flight Simulator require windows 7 or above.

3.3.2 Functional Requirements (Use Cases)

Use Case Name	Identify Panel
Related Use Cases	Label the Symbols
Trigger	The panel that contains the symbols to be detected is given to the module.
Precondition	The given panel's symbology placements are the same as the training data.
Basic Path	1. The User gives the module an image of the panel in a specific state. 2. The system identifies the panel. 3. The system identifies the places of the symbols that include the values to be labeled.
Alternative Paths	In step 2, if the User has given the exact same panel but with different values, the system will not identify every other symbol one by one again.
Postcondition	Places of the symbols are specified by the system, so the system is ready to label the symbols in the panel.
Exception Paths	The panel might not fit the standards of the trained AI.
Other	Localization and dividing the panel into regions are done by the system during this use case.

Figure 6: Identify Panel Use Case

Use Case Name	Label The Symbols
Related Use Cases	Identify Panel, Assign Values
Trigger	The panel that contains the symbols to be detected is given to the module.
Precondition	The system divided the panel into sections successfully.
Basic Path	This Use Case uses OCR to process the images in the given regions and labels them as "Airspeed Indicator, Attitude Indicator, Slip/Skid Indicator, Altimeter, Vertical Speed Indicator, Horizontal Situations Indicator and Turn Rate Indicator".
Alternative Paths	This use case might be skipped if the panel is the same, but the values shown on the labeled symbols are different, in other words the if the labels are not changed.
Postcondition	Symbols that hold the values are labeled and the values are ready to be read.
Exception Paths	The symbols might not fit the standards of the trained AI.
Other	None

Figure 7: Label The Symbols Use Case

Use Case Name	Assign Values
Related Use Cases	Label The Symbols, Compare
Trigger	The symbols are labeled successfully.
Precondition	The symbols are in a processable (readable) condition.
Basic Path	1. The system executes the OCR on each labeled symbol. 2. It detects the given value that is shown on the symbol and assigns and stores the read value. 3. The value gets assigned to that specific label.
Alternative Paths	If the value is the same as the previous state step 3 can be skipped.
Postcondition	All the labels are assigned a value.
Exception Paths	There were values in a bad condition for a reading (bad quality, etc.)
Other	These values are stored to be compared with the desired ones.

Figure 8: Assign Values Use Case

Use Case Name	Compare
Related Use Cases	Identify Panel, Assign Values, Generate Error Rate
Trigger	Values are assigned to the labels successfully.
Precondition	Values are in the wanted format so there is no type error during comparation (integer, double, float, etc.)
Basic Path	1. 100% accurate values that belong to the given panel to the use case 3.2.1.1 are stored separately by the module with the labels they belong to. 2. The assigned values that we gain from 3.2.1.3 are used by this module. 3. The values that have the same labeled gets compared with the presence of an error percentage. 4. The assigned values will be given a score by the module depending on the wanted values.
Alternative Paths	Compare use case can also be used separately or can be used multiple times per symbol identification.
Postcondition	Each value is signed a score (no null values allowed.)
Exception Paths	Data to be compared with is incorrect.
Other	None

Figure 9: Compare Use Case

Use Case Name	Generate Error Rate
Related Use Cases	Compare
Trigger	Scores are assigned to the assigned values.
Precondition	Every value is given a score.
Basic Path	A general performance of the AI can be obtained by this use case. Depending on the given scores this use case generates a general score for the AI and an error percentage using the gained accuracies.
Alternative Paths	Rather than a grayscale, a true-false (0,1) test can be applicated which would give a less accurate accuracy percentage but would be faster.
Postcondition	Generated percentages are shown.
Exception Paths	Calculation steps were flawed.
Other	In a dynamic model this use case can be improved to test the used hardware.

Figure 10: Generate Error Rate Use Case

3.3.3 Performance Requirements

In order to use the trained model, it is sufficient to use a modern computer. However, in order to complete the training process of the model quickly and smoothly, it is necessary to use a Google Colab working environment or a computer with the minimum system requirements written below.

OS: Windows 8/Ubuntu 14.04.5 LTS (64 bit is required for all operating systems.) CPU: Intel® Core ™ i7 7700/AMD Ryzen™ 7 2700 GPU: Nvidia Tesla K20 / Nvidia GeForce RTX 3080 RAM: 16GB DDR4 DISK: SSD is required

3.3.4 Software System Attributes

3.3.4.1 Security Requirements

The system will not be used without the permission of the relevant units of TAI. The training process of the model will be done in a closed system so that the model is not manipulated.

3.3.4.2 Usability Requirements

The system is designed to be used by people who are competent in this area. Users will be able to use the model we will create as a result of the project by integrating it into their own systems.

3.3.4.3 Reliability Requirements

Reliability is one of the key elements of identification systems. Therefore, artificial intelligence model must be trained and tested with various dataset. Additionally, model performance must be evaluated. We are going to use some performance metrics like accuracy, confusion matrix, precision, recall, f1-score and au-roc.

3.3.4.4 Maintainability Requirements

Artificial intelligence model must be open to changes and updates. In order to increase performance of the model, the training and test datasets can be updated.

3.3.4.5 Availability Requirements

Our model will work on all operating systems.

4. Software Design Document

4.1 Introduction

4.1.1. Purpose

This Software Design Document (SDD) provides the design details of the project “Artificial Intelligence Based Cockpit Symbology Identification System (AIBCSIS)”. Purpose of this document is serving as a guideline throughout development phase of the project for developers.

4.1.2. Project Scope

This software system will be an Artificial Intelligence based identification system that aims to detect, identify and label the symbols on the given panel. This system will be designed to be used in the test processes that are done on the cockpit display screens. Main purpose is to detect the minor changes that the human eye can not detect and calculating an error rate. More specifically, the Artificial Intelligence module will compare the wanted symbol depending on the given test case with the symbol that is being shown on the panel after identifying and labeling it. Then it will produce an error rate depending on the overall accuracy. We aim to have less error percentage due to given values on the cockpit panels being very sensitive and the possibility of the human eye not detecting minor changes being higher and to automate the testing process. The system will contain a database of various panels and will identify the parameters as Airspeed Indicator, Altimeter, Vertical Speed Indicator and etc.

4.1.3. Glossary

* SDD --> Software Design Description * AIBCSIS --> Artificial Intelligence Based Cockpit Symbology Identification System * OCR --> Optical Character Recognition

4.1.4. Overview

This Software Design Description (SDD) document provides necessary information about the project Artificial Intelligence Based Cockpit Symbology Identification System. This document includes the design approach, technologies to be used; class, activity and sequence diagrams and the interfaces of the project. These information is aimed to guide any programmer to understand our design and be an assistant in the development phase.

4.1.5. Audience

The main audience of this document includes the developers of this project and Çankaya University CENG407/CENG408 course management. Developers are

supposed to use this document in the development phase of the structure and design of each component.

4.2 Architecture Design

4.2.1. Design Approach

We plan to use the agile development method throughout our project. We chose scrum, which is an agile software development methodology, because it meets our demands. The Scrum method will give us the flexibility and speed we want. It will be possible for us to make some dynamic changes in line with the wishes of the company we work with within the framework of our project. As a team, we have shared all the work to be done in order to complete the project on time. In addition, we hold frequent meetings in accordance with the Scrum and ensure that the problems are resolved quickly.

4.2.2. Used Technologies

It is planned to develop the system using python on Google Colab infrastructure. Technologies such as SSD (Single Shot Detection), YOLO (You Only Look Once), Faster R-CNN, TensorFlow will be used for model training. We will work with up-to-date technologies that use the OCR method to extract information from the image. In order to provide user interaction of the system, the interface will be designed using python QT Designer. This interface will be developed to run on windows operating system.

4.2.3. Scenario View

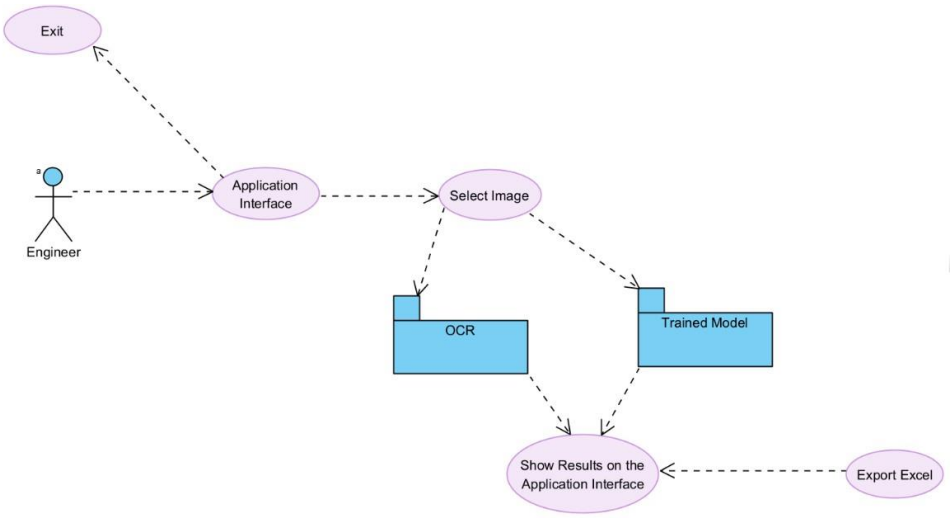


Figure 11: Scenario View

4.2.4. Activity Diagram

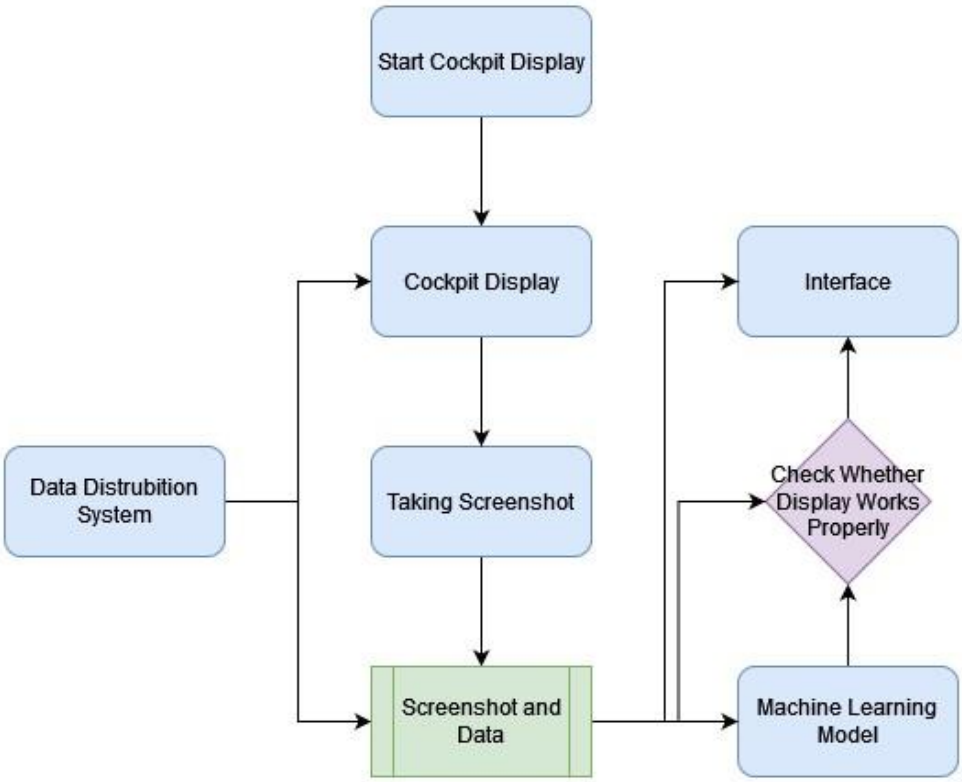


Figure 12: Activity Diagram

4.2.5. Sequence Diagram

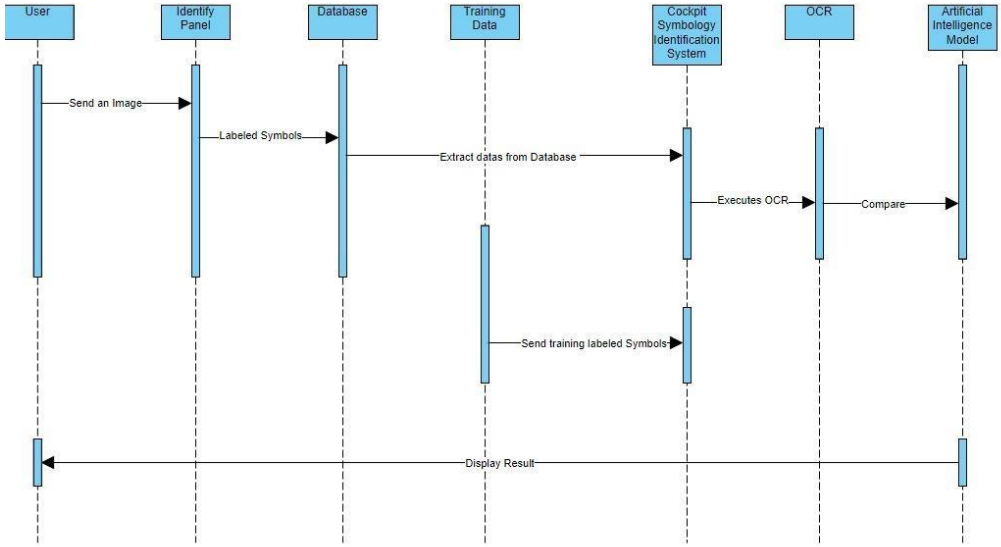


Figure 13 Sequence Diagram

4.3. User Interface Design

* Main Menu

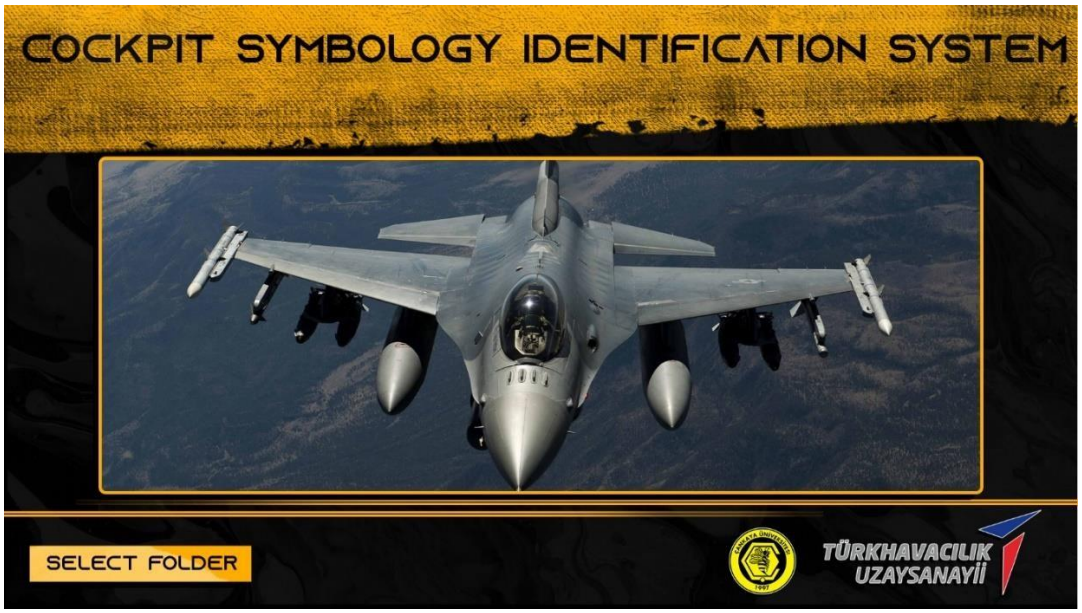


Figure 14: Main Menu

* Image Identification Screen

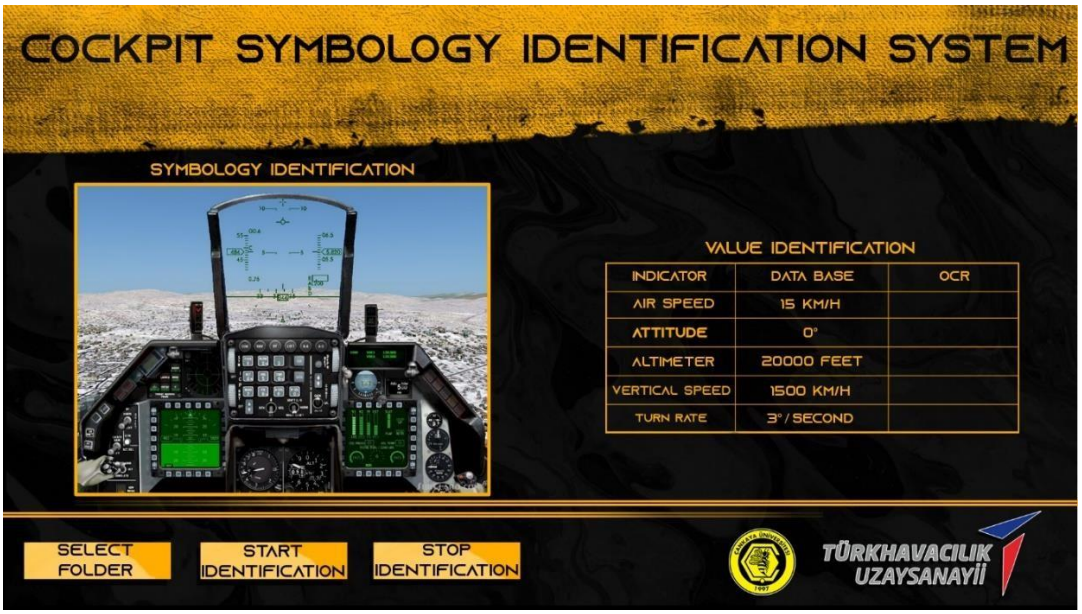


Figure 15: Image Identification Screen

* Results Screen

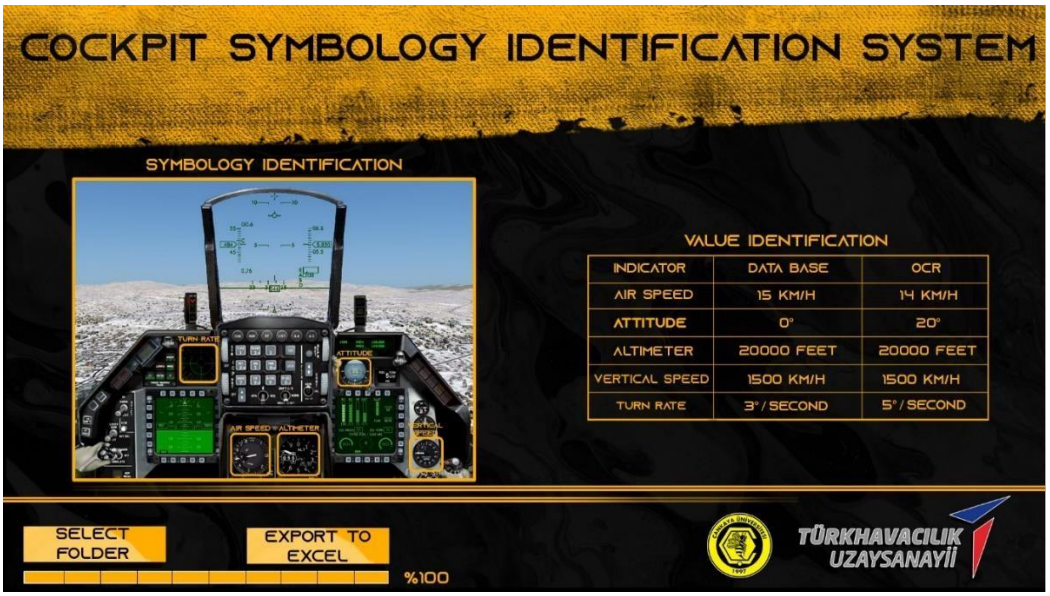


Figure 16: Results Screen

5. Work Plan

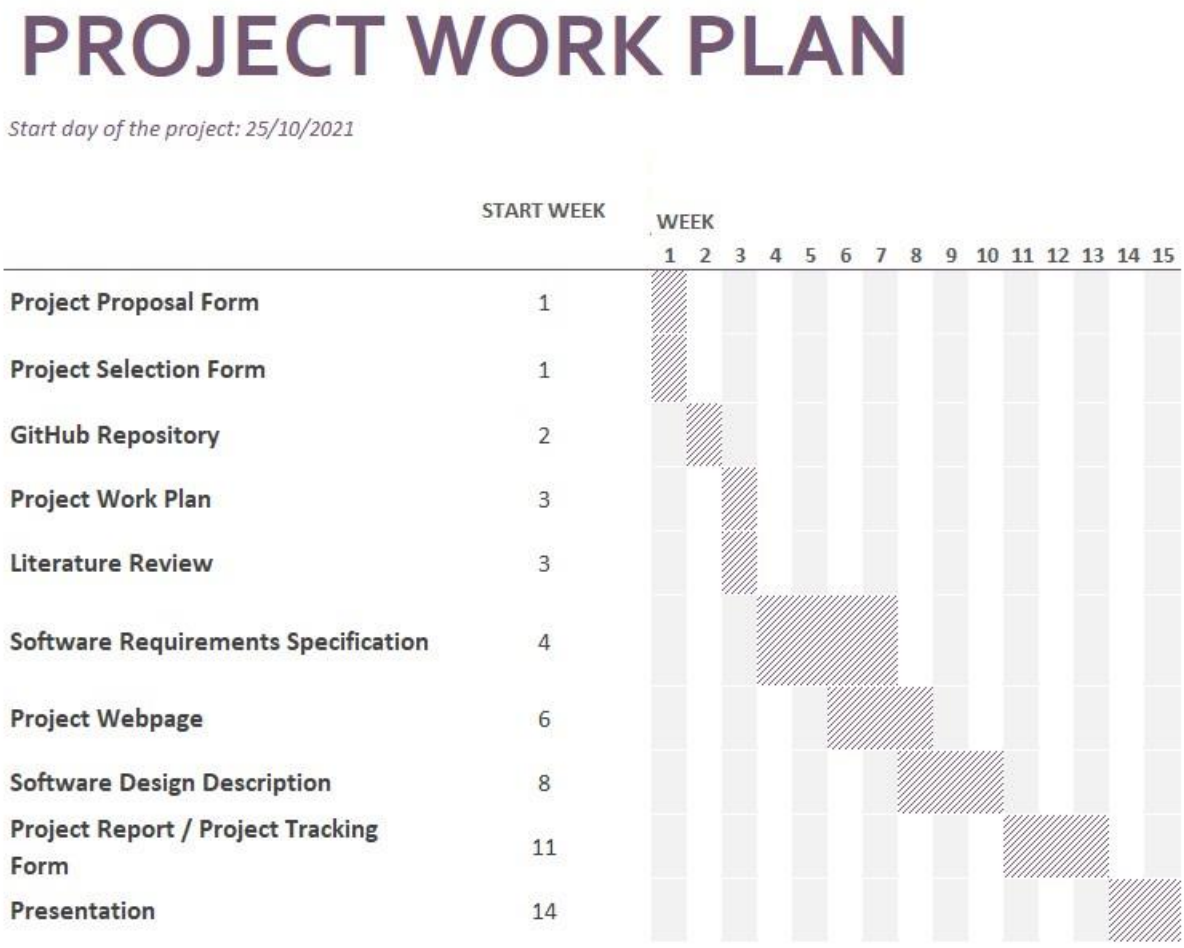


Figure 17: Project Work Plan

6. Conclusion & Discussion

Automated test process of the displayed symbols is a very promising and a needed field. It will help with the reduction of the manual effort and the error rate. Our first goal will be to identify and label the shown symbols on the panel. Considering the importance of the values that are shown on the panel of the aircrafts, we believe having an automated test process is highly crucial. We aim to contribute to this automation process with the Artificial Intelligence module that we will develop.

There are already ways to test the accuracy of the panels on cockpits. But they are done manually. The module that we will develop aims to automate the procedure and by that, decrease the error percentage that was caused by the human eye. We believe such technology will be highly beneficial for the Aerospace Industries and will increase the efficiency/time ratio of the test processes.

7. References

[1] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2014.81

[2] Girshick, R. (2015). Fast R-CNN. 2015 IEEE International Conference on Computer Vision (ICCV). https://doi.org/10.1109/iccv.2015.169

[3] Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149. https://doi.org/10.1109/tpami.2016.2577031

[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. Computer Vision – ECCV 2016, 21–37. https://doi.org/10.1007/978-3-31946448-0_2

[5] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr.2016.91

[6] Pal, D., Alladi, A., Pothireddy, Y., & Koilpillai, G. (2020). Cockpit display graphics symbol detection for software verification using Deep Learning. 2020 International Conference on Data Science and Engineering (ICDSE). https://doi.org/10.1109/icdse50459.2020.9310145

[7] Dongtao, Z., Jie, C., Xing, Y., Hui, S., & Liangliang, S. (2018). Traffic sign detection method of improved SSD based on Deep Learning. 2018 IEEE 4th International Conference on Computer and Communications (ICCC). https://doi.org/10.1109/compcomm.2018.8780999

[8] Wang, R., Xie, G., Chen, J., Ma, X., & Yu, Z. (2014). Traffic sign recognition based on kernel sparse representation. 2014 International Conference on Audio, Language and Image Processing. https://doi.org/10.1109/icalip.2014.7009821

[9] Tang, Y., Zhang, C., Gu, R., Li, P., & Yang, B. (2015). Vehicle detection and recognition for Intelligent Traffic Surveillance System. Multimedia Tools and Applications, 76(4), 5817–5832. https://doi.org/10.1007/s11042-015-2520-x

[10] Zaklouta, F., & Stanciulescu, B. (2014). Real-time traffic sign recognition in three stages. Robotics and Autonomous Systems, 62(1), 16–24. https://doi.org/10.1016/j.robot.2012.07.019

[11] Yu, B., Wang, Y. T., Yao, J. B., & Wang, J. Y. (2016). A comparison of the performance of Ann and SVM for the prediction of traffic accident duration. Neural Network World, 26(3), 271–287. https://doi.org/10.14311/nnw.2016.26.015

[12] Wang, M., Guo, L., & Chen, W.-Y. (2017). Blink detection using Adaboost and contour circle for fatigue recognition. Computers & Electrical Engineering, 58, 502–512. https://doi.org/10.1016/j.compeleceng.2016.09.008

[13] De Sousa Ribeiro, F., Gong, L., Caliva, F., Swainson, M., Gudmundsson, K., Yu, M., Leontidis, G., Ye, X., & Kollias, S. (2018). An end-to-end deep neural architecture for optical character verification and recognition in retail food packaging. 2018 25th IEEE International Conference on Image Processing (ICIP). https://doi.org/10.1109/icip.2018.8451555

[14] Epshtein, B., Ofek, E., & Wexler, Y. (2010). Detecting text in natural scenes with stroke width transform. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2010.5540041

[15] Chen, H., Tsai, S. S., Schroth, G., Chen, D. M., Grzeszczuk, R., & Girod, B. (2011). Robust text detection in natural images with edge-enhanced maximally stable extremal regions. 2011 18th IEEE International Conference on Image Processing. https://doi.org/10.1109/icip.2011.6116200

[16] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

[17] Tanner, M. A., & Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. Journal of the American Statistical Association, 82(398), 528–540. https://doi.org/10.1080/01621459.1987.10478458

[18] Venkateswara, H., Chakraborty, S., & Panchanathan, S. (2017). Deep-learning systems for domain adaptation in Computer Vision: Learning Transferable Feature Representations. IEEE Signal Processing Magazine, 34(6), 117–129. https://doi.org/10.1109/msp.2017.2740460

[19] Shetty, S., Devadiga, A. S., Chakkaravarthy, S. S., Varun Kumar, K. A., Kamalanaban, E., & Visu, P. (2014). Optical character recognition for alphanumerical character verification in video frames. Advances in Intelligent Systems and Computing, 81–87. https://doi.org/10.1007/978-81-322-21265_10

[20] Rao, N. V., Sastry, A. S. C. S., Chakravarthy, A. S. N., & Kalyanchakravarthi, P. (2016). OPTICAL CHARACTER RECOGNITION TECHNIQUE ALGORITHMS. Journal of Theoretical & Applied Information Technology, 83(2).

[21] IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1998 , vol., no., pp.1-40, 20 Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

[22] Leffingwell, D., & Widrig, D. (2003). Managing software requirements: A use case approach. Addison-Wesley.

[23] https://www.alibabacloud.com/blog/how-to-create-an-effective-technical-architecturaldiagram_596100

8. Test Plan

8.1 Introduction

8.1.1 Version Control

Version No	Description of Changes	Date
1.0	First Version	April 1, 2022

8.1.2 Overview

Artificial intelligence-based cockpit symbology identification system will be tested in terms of accuracy of the model with using different algorithms.

8.1.3 Scope

This test plan document includes different test cases about artificial intelligence-based cockpit symbology identification system.

8.1.4 Terminology

Acronym	Definition
SRS	Software Requirement Specification
SDD	Software Design Document
OCR	Optical Character Recognition
GUI	Graphical User Interface
PFD	Primary Flight Display

8.2 FEATURES TO BE TESTED

This section lists and gives a brief description of all the major features to be tested. For each major feature there will be a Test Design Specification added at the end of this document.

8.2.1 Artificial Intelligence Model (AIM)

Artificial intelligence-based cockpit symbology identification system can verify indicator on the PFD screen.

8.2.2 Graphical User Interface (GUI)

Artificial intelligence-based cockpit symbology identification system has a GUI that includes buttons, labels and tables.

8.2.3 Optical Character Recognition (OCR)

Artificial intelligence-based cockpit symbology identification system can read indicator data on the PFD with using OCR.

8.3 FEATURES NOT TO BE TESTED

8.3.1Testing For Original Data (TOD)

Data that obtained by OCR will not be compared with original data by Artificial intelligence-based cockpit symbology identification system.

8.4 ITEM PASS/FAIL CRITERIA

8.4.1 Exit Criteria

- 100% of the test cases are executed
- 90% of the test cases passed
- All High and Medium Priority test cases passed

8.5 REFERENCES

[1] CENG408_GROUP17_SRS, JANUARY 7, 2022. Available:

https://github.com/CankayaUniversity/ceng-407-408-2021-2022-Artificial-Intelligence-Based-Cockpit-Symbology-Identification-System/wiki/Software-Design-Document

[2] CENG408_GROUP17_SDD, JANUARY 7, 2022. Available:

https://github.com/CankayaUniversity/ceng-407-408-2021-2022-Artificial-Intelligence-Based-Cockpit-Symbology-Identification-System/wiki/Software-Design-Document

8.6 TEST DESIGN SPECIFICATIONS

8.6.1 Artificial Intelligence Model(AIM)

8.6.1.1 Subfeatures to be tested

8.6.1.1.1 Identification of the Indicators (AIM.IOI)

We must identify the different indicators on the PFD.

8.6.1.1.2 Location of Indicator (AIM.LOI)

We must detect the location of indicators on the PFD.

8.6.1.1.3 Prediction of Airplane (AIM.POA)

We must predict the name of airplane.

8.6.1.2 Test Cases

TC ID	Requirements	Priority	Scenario Description
AOM.IOI	2.1	H	Check artificial intelligence model whether identify indicators on the PFD or not
AIM.LOI	2.1	H	Check whether location of indicators are correct position or not
AIM.POA	2.1	H	Check whether name of aircraft is correct or not

8.6.2 Graphical User Interface (GUI)

8.6.2.1 Subfeatures to be tested

8.6.2.1.1 Test of Buttons (GUI.TB)

In this module, buttons of the user interface will be tested.

8.6.2.1.2 Test of Tables (GUI.TT)

The user interface has some tables to show data from OCR and to show original data. At this point, these tables will be tested.,

8.6.2.2 Test Cases

TC ID	Requirements	Priority	Scenario Description
GUI.TB	2.2	H	It will be tested if the buttons in the user interface work correctly.
GUI.TT	2.2	H	The user interface has some tables to compare data that the real data and read by the OCR. Filling these tables with correct data will be tested.

8.6.3 Optical Character Recognition (OCR)

8.6.3.1 Subfeatures to be tested

8.6.3.1.1 Control of the OCR Model Output (OCR.COMO)

We must ensure that ORC gives the valid values. For example, speed cannot be bigger than 3000 km/h.

8.6.3.2 Test Cases

TC ID	Requirements	Priority	Scenario Description
OCR.COMO	2.3	H	It is tested that OCR gives the appropriate outputs

8.7 Detailed Test Cases

8.7.1 Identification of the Indicators (AOM.IOI)

TC_ID	AOM.IOI
Purpose	Obtain higher accuracy with using different algorithms
Requirements	2.1
Priority	High.
Estimated Time Needed	20 Minutes
Dependency	Models should be already created.
Setup	A computer must be provided.
Procedure	[A01] Test data with different algorithms.
	[A02] Calculate accuracy and evaluate metrics.
	[A03] Analyze results.
	[A04] Find algorithm that given higher accuracy.
Cleanup	Exit

8.7.2 Location of Indicator (AIM.LOI)

TC_ID	AIM.LOI
Purpose	Test that the location of the indicators are correct position or not.
Requirements	2.1
Priority	High.
Estimated Time Needed	3 Minutes
Dependency	Models should be already created.
Setup	A computer must be provided.
Procedure	[A01] Test data with different algorithms.
	[A02] Calculate accuracy of the location, and evaluate metrics.
	[A03] Analyze results.
	[A04] Find algorithm that given higher accuracy
Cleanup	Exit

8.7.3 Prediction of Airplane (AIM.POA)

TC_ID	AIM.POA
Purpose	Test that the name of airplane which is predicted by model is correct or not
Requirements	2.1
Priority	High.
Estimated Time Needed	1 Minutes
Dependency	Models should be already created.
Setup	A computer must be provided.
Procedure	[A01] Test prediction with different algorithms.
	[A02] Test model with different airplanes.
	[A03] Analyze results.
Cleanup	Exit

8.7.4 Test of Buttons (GUI.TB)

TC_ID	GUI.TB
Purpose	Detect Buttons are work or not.
Requirements	2.2
Priority	High.
Estimated Time Needed	10 Seconds
Dependency	GUI should be already created.
Setup	A computer must be provided.
Procedure	[A01] Check the positions of the buttons.
	[A02] Verify that the buttons are clickable.
	[A03] Test that the buttons function correctly.
	[A04] If the button functions work fine, complete the test.
Cleanup	Exit

8.7.5 Test of Tables (GUI.TT)

TC_ID	GUI.TT
Purpose	Detect Tables are correct or not.
Requirements	2.2
Priority	High.
Estimated Time Needed	10 Seconds
Dependency	GUI should be already created.
Setup	A computer must be provided.
Procedure	[A01] Find tables.
	[A02] Compare the data in the database with the data in the table.
	[A03] Complete the test according to the compare result.
Cleanup	Exit

8.7.6 Control of the OCR Model Output (OCR.COMO)

TC_ID	OCR.COMO
Purpose	Check that OCR returns valid values.
Requirements	2.2
Priority	High.
Estimated Time Needed	3 Minutes
Dependency	Model and test are applied already
Setup	Test dataset is applied to prepared model and valid cases are prepared for all indicators.
Procedure	[A01] Obtain values with using OCR method.
	[A02] Store the all the values for all given test dataset.
	[A03] Check that value is valid or not.
	[A04] Analyze results.
Cleanup	Exit

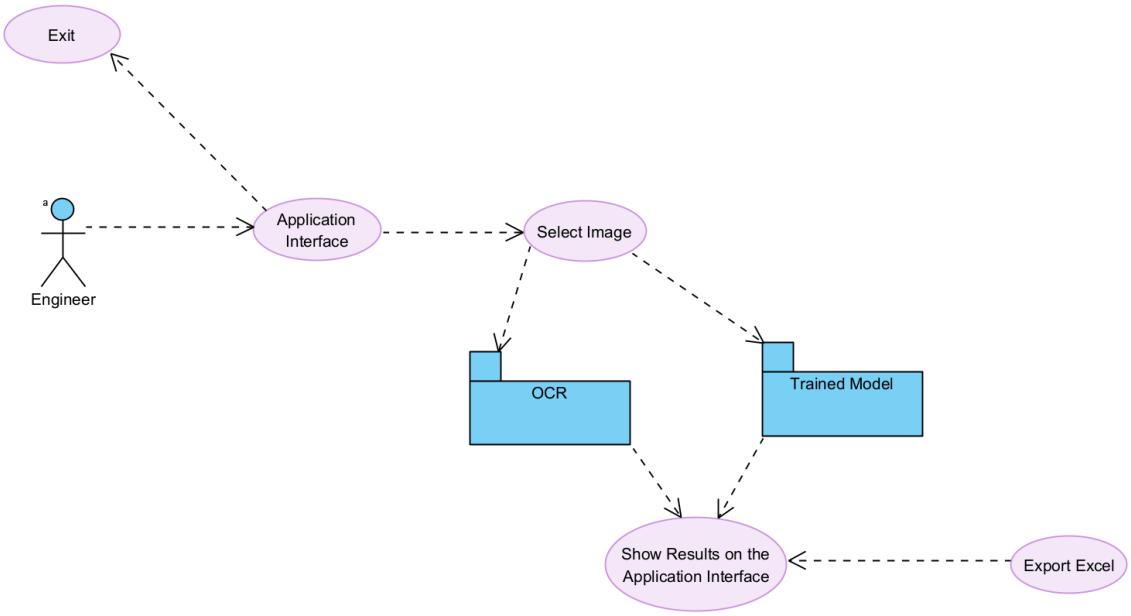
9. Test Results

Test Case ID	Priority	Tester	Date	Result	Comment
AOM.IOI	High	Tahir Bingöl	06.05.2022	Passed	The AI identifies the indicators clearly.
AIM.LOI	High	Nisa Büyüknalbant	08.05.2022	Passed	The indicators' positions are correct.
AIM.POA	High	Tahir Bingöl	24.05.2022	Passed	The names of the planes that were tested were all guessed accurately.
GUI.TB	High	Ümit Kumaş	19.04.2022	Passed	The button works properly.
GUI.TT	High	Emre Canıtez	21.05.2022	Passed	The tables show the data precisely.
OCR.COMO	High	Seyit S. Yiğitarslan	09.05.2022	Partially Passed	The OCR gives proper values.

10. User Manuel

User Manual For “Artificial Intelligence Based Cockpit Symbology Identification System”

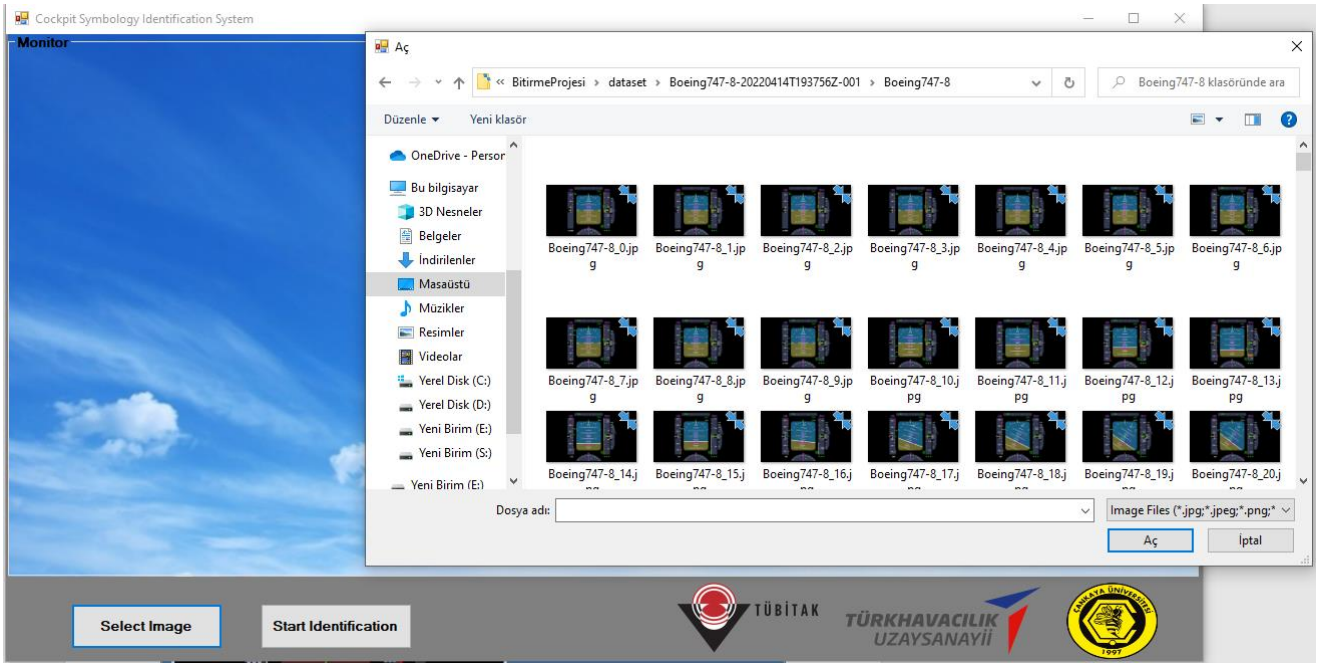
The identification system takes a still image of the PFD (Primary Flight Display) and gives it to the model that has been pre-trained with an AI algorithm as an input. The model detects the locations of the indicators and identifies them (Airspeed, Attitude, Altimeter, Vertical Speed and Horizontal Speed). Afterward, the values given in these indicators get displayed on the screen with the help of OCR.



The interface has been developed with the Windows Forms Application using C#. To give a still image from a certain point of a flight, the user first needs to open the interface. The user will see the given interface after running the application:

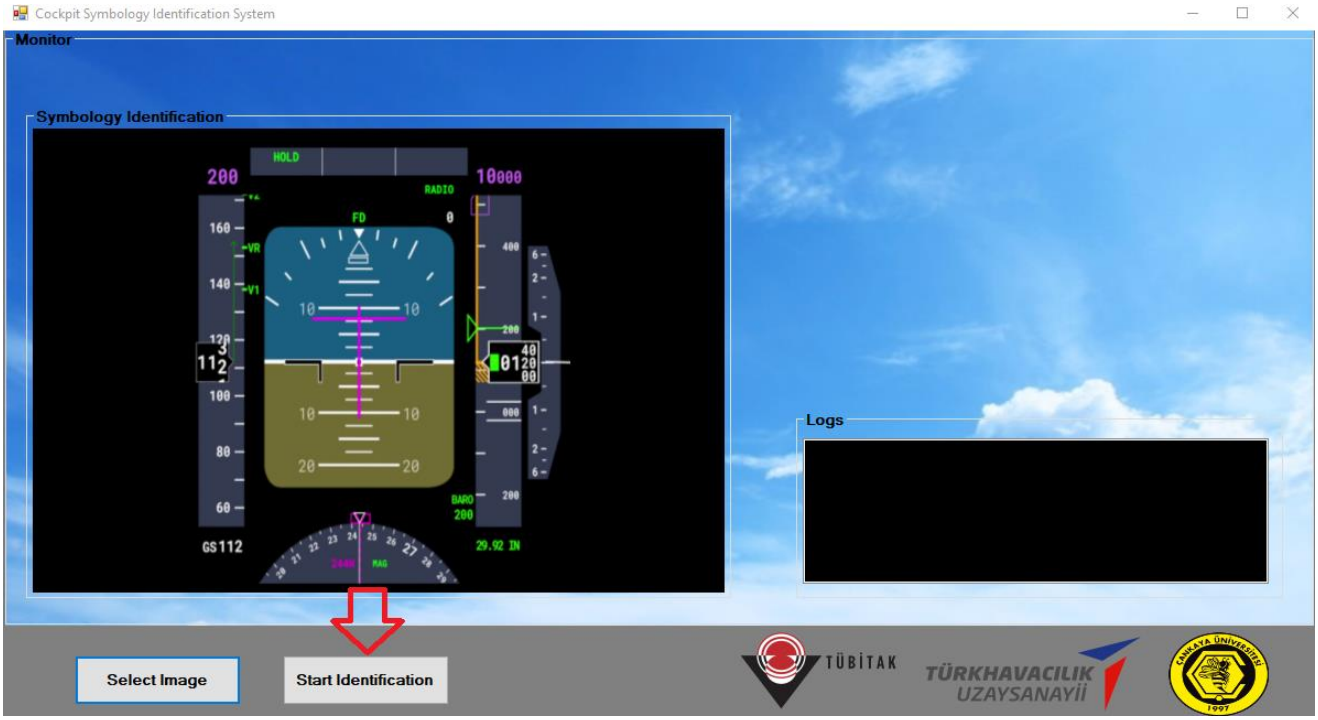
After opening the interface, to upload an image of the PFD the user needs to press the “Select Image” button:

After pressing the button, the file explorer will be opened as such. The user will be expected to select an image afterward:

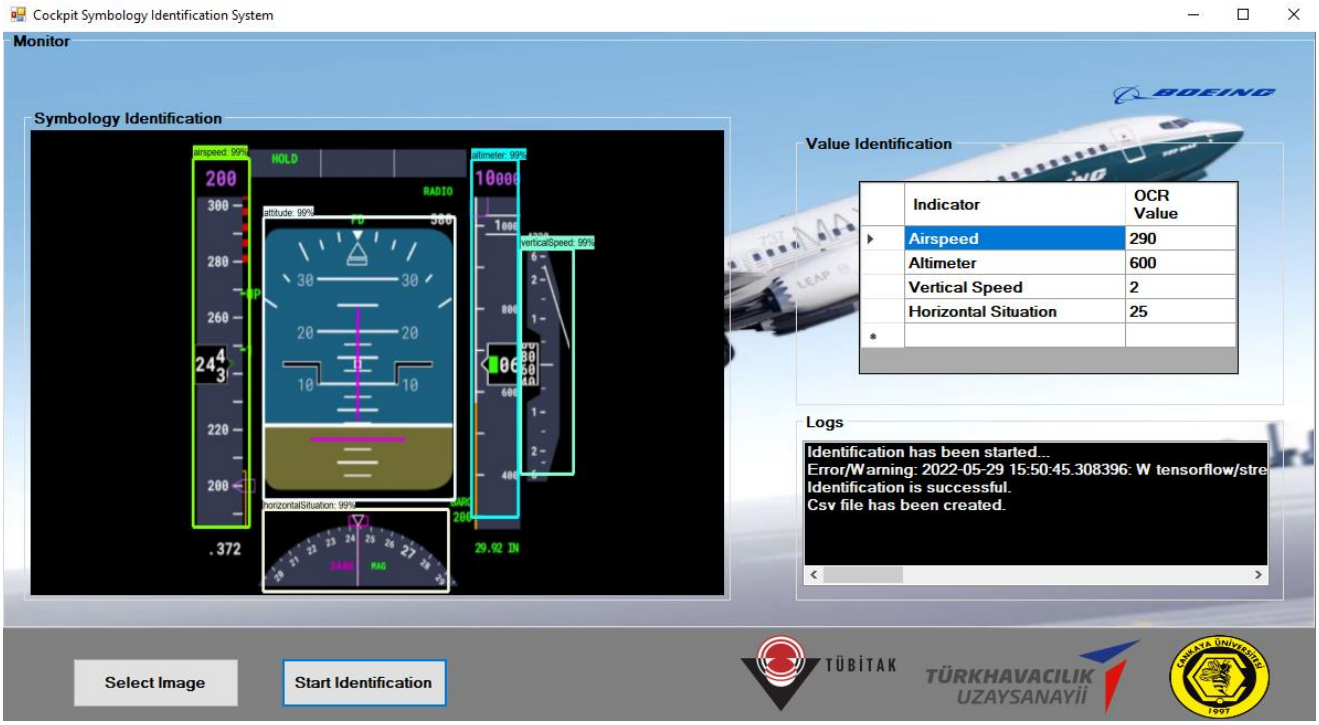


After selecting the image, the user will see the following interface:

To start the identification and send the image to the model, the user needs to press the “Start Identification” button:



After the user presses the button, the image will be sent to the model. The model will identify the locations of the indicators. These locations will be passed to OCR so that the values of the indicators can be read. When the identification gets completed successfully, the interface is expected to show the following interface to the user which includes the locations of the indicators, the accuracy percentages of these locations that have been guessed by the model, and finally the OCR values.



11. Conclusion

As mentioned before, values that are shown on the indicators of a PFD are highly sensitive, and important when it comes down to decision making, necessary calculations and so on. Which makes the test processes of these panels crucial depending on the importance of the accuracy. Knowing that having manual ways to test these panels, which how the tests are done today, is inefficient both time and money-wise, we have aimed to contribute to the automation process of these panels with this project.

We first developed an interface which lets the user give any state of a PFD as an input. Then we sent this input to the various models we have trained using different algorithms (Faster R-CNN, ..., ...) to find an optimum one to use. With the models, we have detected the locations of the indicators and then we have used the OCR technology to read the values that are displayed on the indicators. We gave these values as outputs and let the user export the values as an excel file.

The models we have trained and the application we have developed may not be applicable to real life situations which could require an input stream support. But we believe it is a fulfilling and a necessary step taken for the wanted renovation.