Institute for Information Systems

Bachelor's Thesis
Course of Studies Computer Science

Designing a Conceptual Model and Data Extraction Pipeline for Flight Test Cards

Martin von Depka Prondzinski

Matrikel 5147782

February 12, 2025

Institute for Information Systems
Prof. Dr. Wolf-Tilo Balke
Institute for Application Security
Prof. Dr. Martin Johns

Supervisors:
Dr. Hermann Kroll
Dipl.Ing. Christina Pätzold

Statement of Originality

This thesis has been performed independently with the support of my supervisor/s. To the best of the author's knowledge, this thesis contains no material previously published or written by another person except where due reference is made in the text.

Braunschweig, February 12, 2025

M. v. Depka P.

Abstract

The goal of this bachelor's thesis is to design a conceptual model and a data extraction pipeline for flight test cards, which are protocols created as Word files and filled with results and remarks by the flight test engineer. The concept is necessary because, in the current DLR workflow, flight test cards are stored as PDF files, which is not an optimal solution for matching and comparing flight test card data. This creates the need for a database to ensure suitable accessibility and efficient retrieval of flight test card data. The solution is tailored to the needs of DLR, the German aerospace research and technology center, particularly for all institutes involved in flight test analysis. This is because DLR is actively developing and researching new technologies in the field of aeronautics, and understanding these flight tests requires data from flight test cards. To achieve this, various approaches for extracting data from flight test cards will be explored, including different OCR techniques and table detection methods. The most suitable approach was selected and implemented into a pipeline to automatically extract data from the flight test cards. For efficient storage and querying of the extracted data, a data model was designed to link the flight test card information with the recorded sensor data and the additional information of the flight test, enabling better analysis of the maneuvers performed at DLR. Therefore, the content of the flight test cards was analyzed, along with methods for storing this data, to develop a suitable database model that meets the specific requirements derived from the flight test card content. This addresses the need for flight test cards to fully understand the executed maneuvers. The results of this thesis demonstrate that it is possible to automatically extract data from flight test cards, offering a clear advantage over the current solution. However, as a conceptual model, it also highlights several areas for improvement before the system can be deployed in production. This thesis serves as a foundation for further development and enhancements, with the ultimate goal of integrating the system into DLR's operational workflow for flight test cards.

ifis
Institut für Informationssysteme
Technische Universität Braunschweig

Task Description (Aufgabenstellung)

zur Bachelorarbeit ***„Designing a Conceptual Model and Data Extraction Pipeline for Flight Test Cards“*** *von* ***Martin von Depka Prondzinski*** *(Matrikelnummer 5147782).*

The “Deutsches Zentrum für Luft- und Raumfahrt” (DLR) Flight Experiments Facility operates the largest fleet of civil research aircraft in Europe. All these aircraft are equipped with a large set of permanent sensors. These sensors serve as additional instrumentation for the researching of flight physical effects. All measurement data, as well as a significant number of the parameters from the aircraft's bus system, are continuously recorded using data measurement system. To ensure reusability the maneuvers performed during each flight must be labeled and should be, at best, be stored in an efficient data model. This data model should allow users to search for specific maneuvers and provide an overview of the test procedures.

However, flight cards and their relevant data are filled manually, so extracting information from texts becomes necessary. At the moment, the flight cards are stored as PDF files, with the file name being the only link between the PDF and the flight data. In this thesis, it is essential to create a pipeline for the automated extraction of this data and to develop a data model for the data extracted. The data should be extracted by automated extraction from PDFs via optical character recognition and table extraction. A database must then be created, along with a corresponding data model that can store various maneuvers. The data model should be tailored to meet the DLR's needs, to better reuse and enhance compatibility to connect maneuvers with the timeseries. In this bachelor thesis a concept must be developed for the potential integration of this solution into DLR's workflow and systems, with a focus on one research aircraft as a case study for implementation with two different types of flight test cards.

The tasks to be processed are:

- Requirements analysis and design of a robust conceptual model
- Develop an automated extraction pipeline to read flight test cards from PDF documents and store the extracted data in the designed model
- Evaluate the effectiveness of both the conceptual model and the extraction pipeline using a case study based on two flight test cards

Date: 12.11.2024

(Prof. Dr. Wolf-Tilo Balke)

M. v. Depka Prondzinski

(Martin von Depka Prondzinski)

Contents

List of Figures

List of Tables

1 Introduction

Big data and artificial intelligence have become major buzzwords in computer science. However, even people without a technical background are likely familiar with artificial intelligence or use it daily. The German Aerospace Center (DLR) generates a significant amount of data through its research aircraft operations. Like many other industries, DLR needs to annotate this data to understand the procedures performed. Annotating the data makes it understandable and reusable for researchers which were involved in the project to find easily their maneuvers and perform their analysis on the data as well as evaluating if there any more maneuvers needed for the project success. But especially these researchers who were not involved in the original project get a better opportunity to understand the data and use these data for their research cases. This allows them to locate and review specific maneuvers from past flights, potentially avoiding the need for new flight tests, which save money and reduces the environmental impact of research. Another reason for storing this data is to make published scientific research on these data more understandable by annotating the data and allowing to give background information for a fundamental scientific discussion. This would be done by providing provenience of the flight test data by the data from the flight test card. Our goal is to extract data from flight test cards documents created for each flight that list all the maneuvers performed and develop an optimized data model to store this information in a database. To achieve this, it must build a pipeline that automates these tasks. One of the key components of this pipeline is the database and its data model, designed to efficiently store information from flight test cards. To achieve this, the content of the flight test cards was analyzed to identify relevant data for this case study, forming the basis for building an appropriate data model. Another crucial component is the extraction process, which involves using Optical Character Recognition (OCR) to extract data from PDF files and employing table detection to identify and extract relevant maneuver descriptions. Optical character recognition is the process of converting images or documents containing handwritten or printed text into machine-readable text. For this purpose, various tools exist, which are investigated to determine the best and most suitable one for DLR's flight test cards. For DLR's needs, it requires an OCR tool that performs well with handwritten data, as flight test cards are often filled out by hand during maneuvers. Table detection, or recognition, involves automatically identifying and extracting tables from unstructured documents. This is essential because maneuver is stored in tables, though the schema may vary depending on the flight test engineer. Detecting and extracting this data accurately is crucial for the project's success. It should be explored how similar projects have integrated these steps into a pipeline to ensure that the data is extracted, stored correctly, and easily searchable. An important aspect will be matching the extracted data with the corresponding flight and storing this

information in a way that allows for efficient search and retrieval.

Figure 1.1: DLR reasearch aircrafts

2 Background

In this chapter, the technical and structural background of the input data and the technologies used will be introduced and discussed. This provides an overview of the resources available for the design of the model and clarifies their necessity. To begin, flight test cards, the primary data source, will be introduced, including an explanation of their purpose and the structure of the data they contain. Following this, the flight data storage system of the DLR will be presented, as it is essential to connect the data from the flight test cards with the corresponding flight data. This connection is crucial because the maneuvers described in the flight test cards must be identified in the flight data using the start and end times of each maneuver. Next, possible data models will be discussed to identify a suitable structure for storing the flight test card data. Finally, modules for extracting data from PDFs will be reviewed to explore potential solutions and determine the most appropriate approach.

2.1 Flight Test Cards

This section provides an overview of the purpose and basic structure of flight test cards to clarify why this data is needed and the foundation on which the parsing process is performed. A detailed analysis of the content of the flight test cards specific to the case study will be presented in Chapter 3.

2.1.1 Purpose

A flight test card is essential for the execution of a mission, this is reasoned in the case that the flight test card provides an overview about the mission. The flight test card consist of a subset of defined tests within the overarching flight test plan. As a result, the test team for each flight must develop a detailed mission plan. "The mission details are often documented in a set of flight test cards rather than a flight test plan" (Pavlock, 2013, p. 20 [9]). Flight test cards outline sequences of events organized in a logical, efficient, and safe manner to conduct the test. They contain key attributes, such as test aircraft identification, test card revision and card numbers, test objectives, descriptions of the aircraft and test point configurations, test maneuvers, and acceptance criteria for test points. Given the importance and quantity of information in a flight test card, "(...) each card should be kept clear, concise, and understandable so as not to cause confusion during the mission" (Pavlock, 2013, p. 21 [9]). The flight test card should be reviewed in the pre-mission briefing to ensure that all crew members have a unified understanding. This briefing allows the team to "(...) review the mission objective, scope, procedure, and requirements to ensure

that everyone executes the same test with the same expectations of roles, responsibilities, and outcomes" (Pavlock, 2013, p. 21-22 [9]). During the debrief, the flight test card plays an equally crucial role: "(...) Debriefs are a time to evaluate accomplishments and document anomalies regarding each mission" (Pavlock, 2013, p. 23 [9]). Flight test cards support discussions of what went well, areas for improvement, unexpected responses, and verification of test points. They serve as a record for remarks and comments on the test procedure and confirm that a test was completed, detailing the test's key attributes to ensure efficient, safe, and successful flight testing. "As with the pre-mission brief, a card-by-card review of the test cards must be performed to facilitate detailed discussions among all test team members who were involved with each test action that was performed" (Pavlock, 2013, p. 23 [9]). Reviewing the test cards may also reveal if further data analysis is necessary to ensure no undetected anomalies were overlooked in real time during the test. With the digitalization of test cards, debriefing can be conducted immediately after the flight test, using data directly from the measurement systems. This offers greater clarity regarding test points and flight test outcomes, as the data can be analyzed immediately. Furthermore, in later reviews, digitalized flight test cards enable analysis with a clear understanding of test goals, capturing all information noted during the flight without needing to consult crew members about specific issues or unusual effects encountered. All this information gathered from the flight test cards enhances the flexibility of flight reviewing and analysis.

2.1.2 Structure

In this part of the thesis, the structure of the flight test cards will be analyzed. The initial focus will be on the structure of the two projects that are used as case studies in this thesis.

Mission Information

First, the header section includes the aircraft on which the test is performed, the run, date, page, and the test category of the flight test. Next, the flight purpose is defined on the flight test card, which could specify a project flight (e.g., a data flight), a training flight, or an approval flight for modifications. The following section on the flight test card lists the flight crew, including the number of members, their names, and their roles, such as pilot in command (PIC) or leading flight test engineer (LTFE). The next items are loading and fuel information, followed by takeoff data. This section concludes with the signatures of the person who prepared the flight test cards and the person who reviewed them. All of this data is stored in boxes, which will be useful later if we need to reference this information. However, in the first step, we will focus on the maneuver data recorded on the flight test card. Before defining the maneuvers, the next page describes the objective of the test, references, aircraft information, limitations, and remarks. These sections can span multiple pages depending on the amount of data provided in each category.

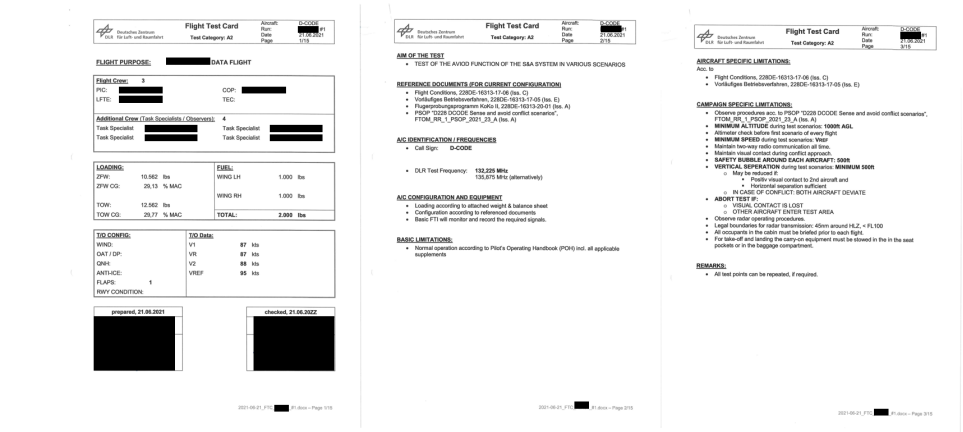


Figure 2.1: Flight test card Mission Information

Preflight information

Following this preflight information, the test program is outlined in the flight test cards. This may begin with preflight preparations on the aircraft, which are typically listed in bullet points, including machine-written text and tables or boxes to record data by hand. This part is followed by the takeoff task in the aircraft.

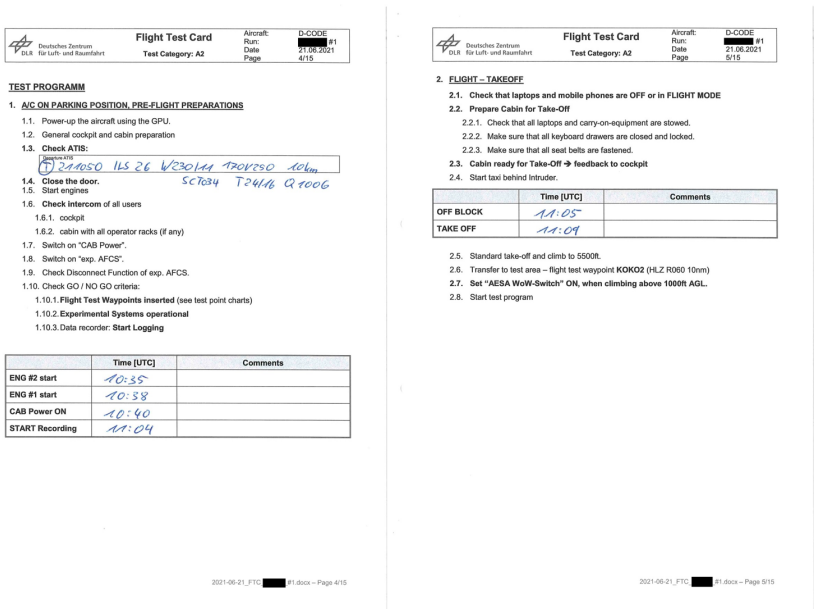


Figure 2.2: Flight test card pre-maneuver tasklist

Maneuver information

The test program section includes test points, each identified by a test point number and a name. For example, test points could be described with images and accompanying machine text outlining the procedure. Next, procedural details are organized in a table to be filled out during the flight test. In our case, a flight test point could also be described with a machine-written table containing the flight path of the test point. This means that, within the test description, there are mostly tables, which can be handwritten or machine-written, but they generally include images to illustrate the procedures, especially in the cards used for the case study.

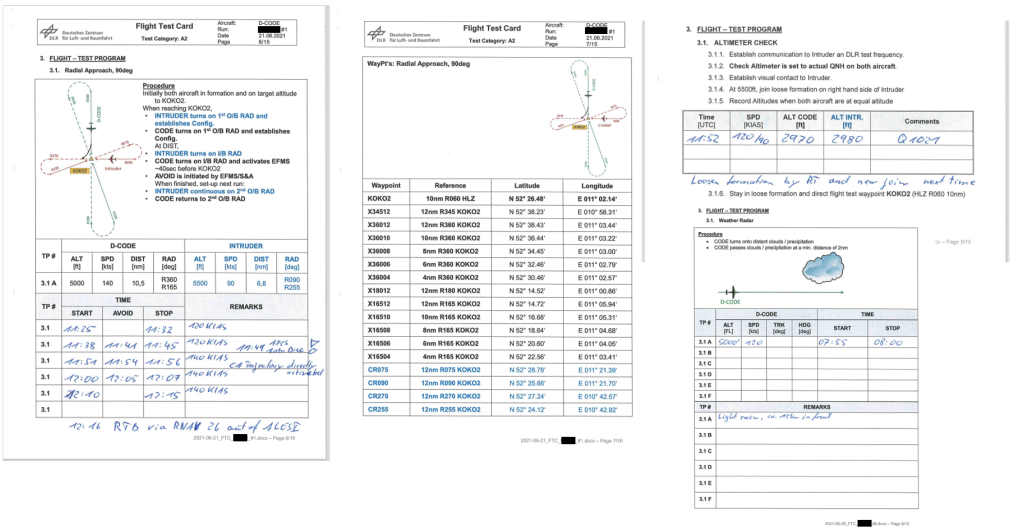


Figure 2.3: Flight test card maneuver

After flight information

After all these test procedures, there is a section on approach and landing. Similar to the pre-flight preparations, this section is structured with bullet points of machine-written text and contains boxes with labels and tables with key points to be filled out by hand during the flight. In the final sections, there may be machine-written tables showing variations that could be used during flight to determine relationships between attribute values. Lastly, there is a page dedicated to weight and balance. This page includes a table with weight and balance data as well as a diagram showing the center of gravity. For the purposes of this thesis, it is helpful that each handwritten entry is placed in boxes and tables, as this makes structural recognition of the data easier. Initially, the search will focus on the boxes containing maneuver information.

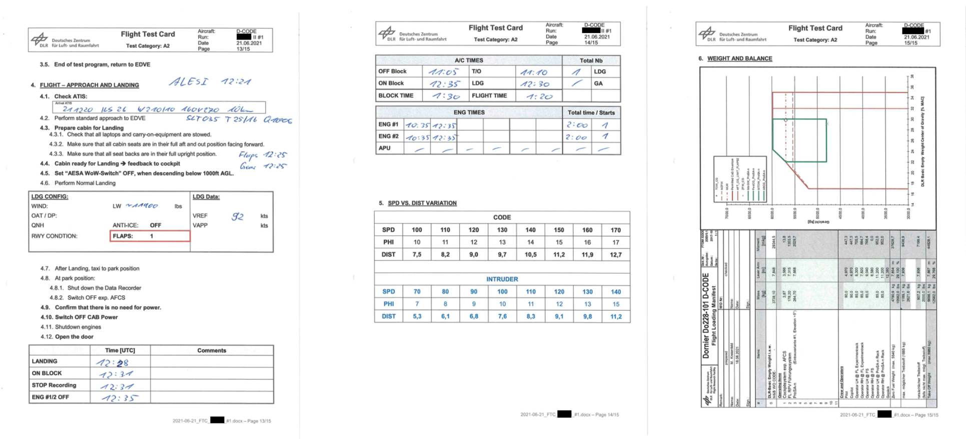


Figure 2.4: Flight test card after flight list

2.2 twinstash - Storage and application hub for mesasurement data

This section explains the purpose of twinstash, along with its structure and technical design. This information is essential because, in this thesis, the flight data must be linked to the data from the flight test cards. Establishing this connection enhances the comprehensibility of the flight test data by incorporating information about the performed maneuvers.

2.2.1 Purpose

The twinstash is a “digital twins storage and application service hub” developed by the German Aerospace Center (DLR). The primary goal of this application is managing large volumes of herogeneous flight research data, preserving the context of this data over long periods of time and ensuring the reuse and availability of the data. Currently, the project aims to “provide a basis for Digital Twins of the DLR research aircraft” (Haufe et al., 2022, p.1 [6]). More specifically, the system “stores data according to defined integrity measures, allows users to search and retrieve data in small segments, and provides functionality for data manipulation” (Haufe et al., 2022, p.2 [6]). This is reasoned in the lack of centraliesed, well-structured system which often leads to inefficient processes, data duplication and limited collaboration within and outside the DLR. For flight data-driven research projects, it is very difficult to obtain representative data; data in standardised formatting and the required context are not accessible. The required citability of data is also not given. With the twinstash the following requirements should be fulfilled: the research data must be saved in accoredance with the DFG’s "Guidelines for safeguarding good scientific practice" and research data should also be stored sustainably for subsequent use, secured and made publicly accessible in the sense of Open Access, as laid down in the "Berlin Declaration on Open Access to Knowledge in the Sciences and Humanities" of 2003. To cover these requirements the twinstash is developed. In the futher the FAIR data standards: the FAIR principles (Findable, Accessible, Interoperable, Reusable) should be fulfilled and thus ensure that research data is easy to find, accessible and reusable. All these features can be performed through the integrated application programming interface (API) and the most of them could be used through the browser-based user interface which is based on the API.

2.2.2 Structure and technical design

The basic structure of twinstash contains a frontend and a backend. The frontend is connected to the backend through a Representational State Transfer Application Programming Interface (REST API). The frontend includes a Web GUI and a Python client, with the potential to extend to other clients, such as a C++ and Java client to provide more flexi-

blity for the researchers to connect their evaluation programs. In the initial step, the REST API is described, as this provides the opportunity to insert data into the system. It is mentioned that the REST API "(...) imposes certain constraints on how data can be identified and manipulated through an API." (Haufe et al., 2022, p.1 [6]). This is achieved through the client by proving and eventually rejecting the requests, if they do not fit into the constrains which will be provided through the REST API. These HTTP requests are formatted strings, and the REST API only accepts TLS-encrypted HTTP, reasoning that "(...) ensuring that no communication may be eavesdropped by third parties." (Haufe et al., 2022, p. 2 [6]). TLS stands for Transport Layer Security, a cryptographic protocol for communication security over a computer network that aims to provide privacy, integrity, and authenticity through cryptography. The REST API of twinstash provides various methods, including data download, upload, manipulation, and deletion, as well as endpoints like "search" to enable extensive search capabilities for the data stored in twinstash. Next, the frontend is examined, as it contains the Python client and the Web GUI, which provide the ability to view and insert data through the Python client and Web GUI. The Web GUI runs in modern web browsers, allowing users to access it by navigating to the twinstash URL (currently only available within the DLR intranet). Through the Web GUI, users can search and visualize the data stored in twinstash. The Python client is provided as a Python package, enabling access to all twinstash functionalities via Python functions. For authentication, twinstash uses an API key associated with the user. Next, the backend's technical design is reviewed. The backend includes a Web GUI provider that serves HTML, JavaScript, CSS, and image files, which are rendered client-side in the web browser. For Web GUI deployment, twinstash uses Angular, which provides "(...) high-level features to increase efficiency in GUI development, such as the separation of HTML and business logic, components for encapsulation, TypeScript for compile-time error handling, and efficient routing between subpages." (Haufe et al., 2022, p. 2 [6]). twinstash uses Keycloak for authentication, an open-source Identity and Access Management system that handles all necessary authentication for the backend. One key module of the backend is the twinstash service, which provides twinstash's core functionalities. The twinstash service "(...) stores data according to defined integrity measures, allows data to be searched for retrieval in small slices, and provides functionality for data manipulation." (Haufe et al., 2022, p. 2 [6]). This service is implemented in Python and uses the Flask framework for its REST API, with data stored in MongoDB, a NoSQL Database Management System. The next step involves a detailed examination of the data model. In the data model, "Each root element is a project which contains flights and files such as project descriptions or an image of the aircraft." (Haufe et al., 2022, p. 5 [6]). For this thesis, the storage of metadata, especially metadata from the flight contained in the Flight Test Cards, is particularly important. Metadata can be attached to the project, flight series, or a file. Because MongoDB uses JSON-based storage, twinstash stores all metadata for a project, flight, series, or file in JSON documents. These documents are organized in a tree structure by storing references to the parent documents. In the final step, it's essential to review the structure of the different data types

that can be stored in the database to determine how to connect data from the flight test card to data in the database. Projects and flights are structured similarly, so they are described together. They include the property "type," which can take values such as project, flight, series, or file, as well as properties such as id, name, created_at, created_by, and user_tags. The id and the two "created" tags are system-generated; "created_by" contains information about the individual who created the data element, and "created_at" records the time of data creation. For storing flight metadata, it is important to note that "The property user_tags can freely be filled by the user, including with a complex tree-like sub JSON." (Haufe et al., 2022, p. 5 [6]). However, this thesis focuses on creating a data model for data from the flight test card, making it necessary to find a way to link this data model to the user_tags of a flight. The "series" and "files" types are also similar and are therefore described together. They contain the property "parent," referring to a flight. The series type includes the property "unit," which "(...) stores the unit of the parameter represented by this series." (Haufe et al., 2022, p. 5 [6]). The series holds data as a 2D data set and may contain a "represents" tag, which "(...) can be used to annotate a series with system information such as time, longitude, latitude, or height, which is interpreted by twinstash to construct a flight trajectory." (Haufe et al., 2022, p. 5 [6]). The key difference between the data stored in twinstash and the data from the flight test cards lies in their origin and purpose. The data in twinstash is automatically collected by the measurement system of the research aircraft, while the flight test card data contains information about maneuvers and their descriptions. The data is either predefined before the flight and the times of this maneuvers get added or documented by the flight test engineer during the flight. Currently, twinstash only stores data from the measurement system. However, there is a need to also integrate the data from the flight test cards to obtain maneuver information and correlate the predefined parameters with the values measured by the aircraft's measurement system. That means that the structure above is only for the description of the flight data and the tags are important for linking these data to the data from the flight test cards. Files may also include a "represents" tag, allowing for the current option of storing a sensor_grouping JSON file for each aircraft. This JSON file enables the grouping of sensors and provides descriptive names for the sensors, which are often cryptic. The series also include an automatically generated "statistics" property, which provides the following information for each uploaded series: minimum, maximum, mean, median, and standard deviation. Finally, there are several properties unique to files, such as data_id, size, as well as the same properties as series: type, id, name, parent, user_tags, created_at, and created_by. In the subsequent parts of this thesis, it is essential to discuss which properties of the data model could be useful for linking the data model for maneuver data from the flight test cards.

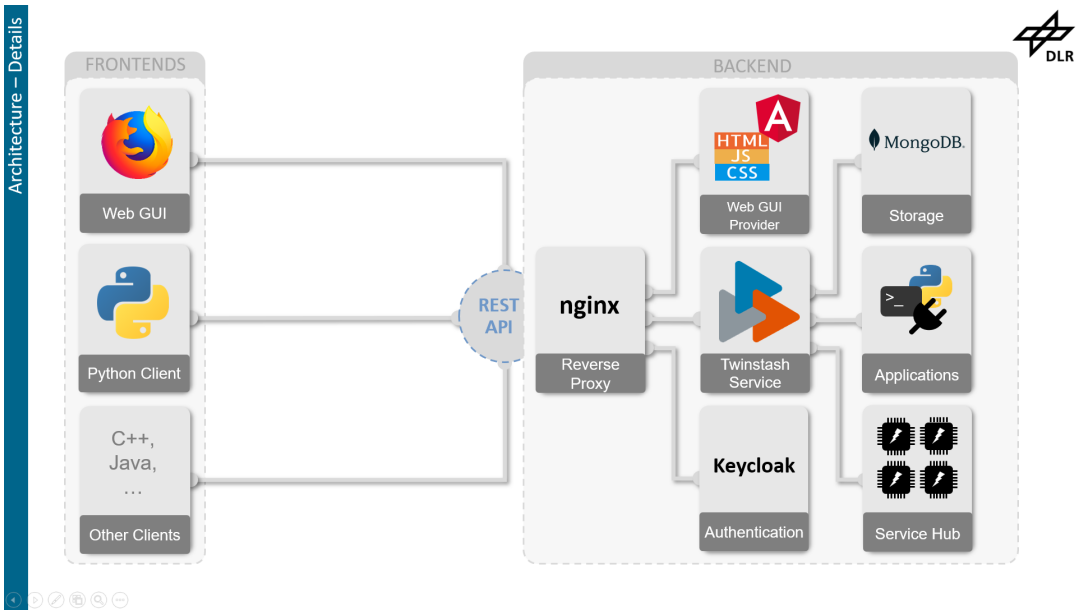


Figure 2.5: twinstash Architecture containing frontend and connection to the backend as well as the backend modules

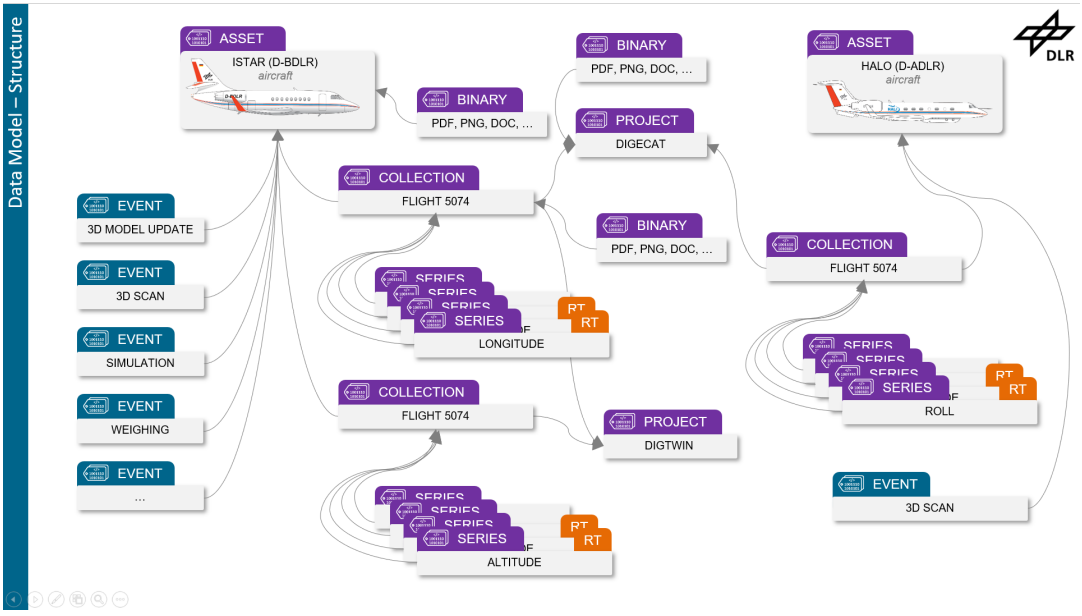


Figure 2.6: twinstash Datamodel Example

2.3 Databases and Data Models

This section of the thesis provides an overview about relational models, along with fundamental information on normalization of relational models. This serves as a foundation for storing the data and justifies the selection of the storage solution.

2.3.1 Relational Models

In this section, the relational model will be briefly introduced to provide a basic rationale for its usage and to explain why certain decisions are made during its implementation. The relational model used in this thesis is the one provided by E.F. Codd. The author noted that "A database can be of two major types: production-oriented or exploratory." (E.F. Codd, 1990, p.5 [1]). However, in both cases, it is crucial to ensure accuracy, consistency, and integrity of the data. This necessity arises from the large quantities of data being shared among multiple users who perform actions on the data independently of one another. A key aspect of the relational model is its method for managing duplicate data within a single table. To eliminate redundancy, repeating data can be extracted and stored in a separate table, which is then related to the original table through a foreign key. This approach ensures that the same element is not repeatedly defined in multiple cells of one table, promoting data normalization and improving efficiency. As a first step in replacing duplicates, the meaning of each element must be clearly defined so that all users understand the values. The author describes this by stating, "(...) the sharing of meaning requires that there exist a single, simple, and explicit description of the meaning of every row in every relation." (E.F. Codd, 1990, p.6 [1]). In the author's opinion, consistent naming conventions that are universally understandable to all users cannot be achieved because "the sharing of data requires the sharing of its meaning." (E.F. Codd, 1990, p.6 [1]). This highlights the challenge of ensuring that data names convey the same meaning across diverse user groups with varying contexts and interpretations. To address this issue, he developed the relational model and stated, "The relational model can be construed as a highly disciplined approach to database management. Adherence to this discipline by users is enforced by the DBMS, provided that this system is based whole-heartedly on the relational model." (E.F. Codd, 1990, p.6 [1]). Fundamentally, a relational database is a collection of relations of assorted degrees. This means that query and manipulation operations are performed on relations and generate relations as results. The author also mentioned that "Relational databases that have many relations, each with few rows (tuples), are often called rich, while those that have few relations, each with many rows, are called large." (E.F. Codd, 1990, p.7 [1]). Overall, the relational model offers a structured approach to avoiding duplicates and making data more comprehensible across different users. This approach is particularly relevant for this thesis and the DLR, as many scientists with diverse knowledge backgrounds will work with the data. Therefore, finding a standardized solution for storing data from the flight test cards is essential.

2.3.2 Normalization

In this section, the normalization of relational models is discussed to structure data into different tables and provide a more stringent and efficient data storage approach. First, the author outlines the objectives of normalization as described by E.F. Codd [2]:

1. "To free the collection of relations from undesirable insertion, update and deletion dependencies;" (E.F. Codd, p.1 , 1972[2])
2. "To reduce the need for restructuring the collection of relations as new types of data are introduced, and thus increase the life span of application programs;" (E.F. Codd, p.1 , 1972[2])
3. "To make the relational model more informative to users;" (E.F. Codd, p.1 , 1972[2])
4. "To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by." (E.F. Codd, p.1 , 1972[2])

The author mentions that as normalization progresses from the first to the third normal form, the conditions for structuring data become increasingly stringent. This process is particularly important for large, integrated databases where logical guidelines are required to ensure data integrity and efficiency. When describing the querying capabilities of the normal forms, Codd meantioned: “(...) the three normal forms are query equivalent in the sense that the set of queries answerable by a collection C in first normal form is transformable into queries yielding the same information from the second and third normal forms of C, there is a difference in information content of the three forms." (E.F. Codd, p.28 , 1972[2]). The second and third normal forms, while improving data structure, introduce additional tables and associations between tables, which can lead to more complex queries. Despite these added complexities, the improved structure and reduced redundancy offer significant advantages, how these advantages are achieved is illustrated in Figure 2.7. The conclusion is that applying normalization enhances the overall quality of the data model in this thesis by improving consistency, efficiency, and maintainability.

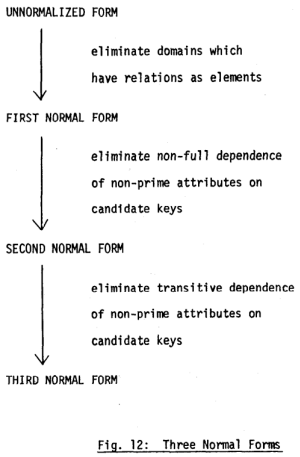


Figure 2.7: Three Normal Forms (E.F. Codd, p.27, 1972, [2])

2.4 PDF Parsing

This section of the thesis discusses PDF parsing methods, focusing on OCR tools and table detection mechanisms to identify suitable tools for the case study. The advantages and disadvantages of each approach will be evaluated and aligned with the specific requirements of the case study.

2.4.1 OCR - Optical Character Recognition

In the first step it is important to evaluate suitable OCR models for reading handwritten data from PDF files. To explore different types of models, to get an overview of relevant techniques. For getting insights to relevant techniques of OCR models the paper "Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)" by Memon et al. (2019) [7], is used, for getting positive and negative aspects of different models. They state that "In the current decade, researchers have worked on different machine learning approaches which include Support Vector Machine (SVM), Random Forests (RF), k-Nearest Neighbor (kNN), Decision Tree (DT), Neural Networks, etc." (Memon et al., 2019, p. 3 [7]). It should be focused on these newer machine learning approaches, as they offer more accurate results compared to older methods like template matching. However, problems still exist, such as the issue noted by the authors: "Characters written by different individuals create large intra-class variability, which makes it difficult for classifiers to perform robustly" (Memon et al., 2019, p. 23 [7]). This issue is significant, and to address it in the early stages of our project, to ensure that all our flight test cards are written by the same person to avoid problems in recognizing the data from the cards, in the long term we want to recognize all handwritten data in the flight test cards not only these which were written by one and the same person. Next, it should be investigated a statistical-based Support Vector Machine (SVM), which offers broad recognition

capabilities for handwritten characters. As Hamdan and Sathish (2021) [5] state: "This research article developed the framework to recognize various stylish characters, providing better recognition rates and accuracy" (Hamdan & Sathish, 2021, p. 103 [5]). Their SVM is combined with an artificial neural network (ANN). The ANN consists of interconnected processing units, analogous to biological neurons, where the system learns by adjusting the weights between the units. The SVM process takes a dataset as input and divides the data into training and testing sets. In the first step the training process should get described and then compared to the testing process, for getting the difference between the training and testing process. For the training, they state: "This process consists of input data and a map for the predefined label" (Hamdan & Sathish, 2021, p. 98 [5]). After receiving the input, the data undergoes preprocessing, followed by feature extraction. Using the extracted features, the SVM model is trained, leading to the proposed trained SVM framework, which is then validated. The testing set follows the same steps, but without training the model. They explain that "These nodes are associated and adjusted for potential errors in learning time samples in a controlled environment" (Hamdan & Sathish, 2021, p. 98 [5]). For feature extraction, they use kernel-based learning models incorporating principal component analysis. Contextual feature extraction is applied for optical pattern text categorization and time series prediction. The feature vectors are mapped into a large dimensional space, and hyperplanes are used for class separation and classification. They add that "This technique also proves more robust for HCR with image classification through text content in the images" (Hamdan & Sathish, 2021, p. 98 [5]). They provide a formula for this kernel approach, including a kernel function, a threshold for the hyperplane, and Lagrange multipliers for optimization. To summarize their approach, it should be examined their feature extraction and classification in more detail. The feature extraction process takes an image as input, which passes through convolutional and sampling layers. The output of these layers is fed into a second convolutional layer and sampling layer, whose output serves as input for the classification task. In the classification stage, the output of the feature extraction is flattened into feature vectors. These vectors pass through a fully connected layer network, resulting in a flattened softmax vector, which represents the predicted class. This model is promising for our needs, as it offers "94% accuracy and a good recognition rate compared to existing methods" (Hamdan & Sathish, 2021, p. 103 [5]). In the next step, it should be compared this high accuracy OCR model, to the OCR-D tool which is an open-source library for optical character recognition form the German research community (DFG), which is also the reason for the name OCR-D. The OCR-D project is divided into various modules, but it should be focused only on those relevant to our use case. One module implements Tesseract into the OCR-D workflow. Tesseract is a widely used open-source OCR tool. The OCR-D project has two main goals with the Tesseract integration: supporting its connection to other modules and improving Tesseract's stability, code quality, and performance (Weil, 2020, p. 223 [3]). They achieved this by extending the existing Python interface and adding support for recognizing single characters. They mention that this improved the post-correction model of OCR text

recognition (Weil, 2021, p. 223 [3]), resulting in more accurate character and word coordinates. Standardized functions and data types were used to enhance the maintainability of Tesseract. According to the authors, character recognition rates exceed 90%, and the error rate is below 10% (Weil, 2020, p. 223 [3]). With specific retraining, the error rate could potentially drop to below 1%. To further evaluate this approach, it should be examined the structure of Tesseract. The paper "An Overview of the Tesseract OCR Engine" (Smith, 2007 [10]) describes the Tesseract algorithm, which accepts a binary image as input with optional polygonal text regions. The process follows a step-by-step pipeline, starting with a connected component analysis, where the outlines of components are stored. These results, along with the number of detected child and grandchild outlines, are stored as "Blobs." The Blobs are organized into text lines, which are then analyzed for fixed-pitch or proportional text. The text lines are broken down into words based on character spacing. The recognition process is a two-pass procedure. In the first pass, the system attempts to recognize each word. If successful, the word is passed to an adaptive classifier for training, which helps improve recognition for subsequent words on the page. If the adaptive classifier learns too late, it makes a second pass to re-recognize unsatisfactory words. In the final phase, fuzzy spaces are resolved, and alternative hypotheses for the x-height are checked to locate small-cap text. A key issue with Tesseract's recognition method is that the features in unknown data do not always match those in the training data, which stems from the fact that the training features are derived from segments of the polygonal approximation. During recognition, however, fixed-length features are extracted and matched against prototype features using a many-to-one method. This process is computationally expensive: "The main problem is that the computational cost of comparing the distance between an unknown and a prototype is very high" (Smith, 2007, p. 3 [10]). Tesseract's classification is a two-step process. First, a "class pruner" creates a shortlist of character classes that might match the unknown, by summing up all bit-vectors of potentially matching classes. The top classes, corrected for the expected number of features, are then shortlisted for further comparison. Unknown features are checked against a bitvector of prototypes, and if they match, the actual similarity is calculated. In the final classifier stage, the summed feature and prototype evidences determine the best overall class configuration. In the adaptive classifier, "The only significant difference between the static classifier and the adaptive classifier, apart from the training data, is that the adaptive classifier uses isotropic baseline/x-height normalization" (Smith, 2007, p. 4 [10]). Based on our project's requirements, OCR-D with Tesseract seems to be the best fit. The primary reason is that OCR-D is open-source and modular, allowing us to select and combine only the modules that suit our needs. Additionally, Tesseract is a well-documented OCR tool, which, as noted in the OCR-D paper, should perform well.

2.4.2 Table Detection

Detecting tables is a crucial task for us because all maneuver-relevant data is stored in tables. This detection task must be highly accurate and capable of easily recognizing various types of tables, as tables can differ between projects based on specific project requirements. Therefore, methods from the paper "Image-based Table Recognition: Data, Model, Evaluation" (Zhong et al., 2020 [12]) should be applied. They use an Encoder-Dual-Decoder (EDD) model, which "... consists of an encoder, an attention-based structure decoder, and an attention-based cell decoder" (Zhong et al., 2020, p. 4 [12]). The two decoders serve different purposes: first, they identify the table structure in the document, and then they extract the content from the cells. An additional advantage of using two decoders is that the cell decoder can use information from the table structure recognition to help locate the cells that need to be recognized. The encoder is a convolutional neural network (CNN) used to capture features of the table images, while both the structure and cell decoders are recurrent neural networks (RNNs) with attention mechanisms. A key feature of their model is that the structure decoder generates HTML tags to define the table's structure. This is helpful for us because HTML is a machine-readable, easy-to-understand format that simplifies working with table data. After the structure decoder recognizes a new cell, the cell decoder is triggered, using the state of the structure decoder to compute the cell's content, while the structure decoder remains in a hidden state to preserve computational focus on cell content extraction. They also note, "This ensures a one-to-one match between the cells generated by the structure decoder and the sequences generated by the cell decoder" (Zhong et al., 2020, p. 4 [12]). The outputs of the two decoders are merged into an HTML representation. During training, the structural and cell tokens are created separately, with structural tokens including HTML tags and cell content tokens treating HTML tags as individual tokens. This tokenization results in two loss functions in the EDD network: the cross-entropy loss for generating structural tokens and the cross-entropy loss for generating cell tokens. Cross-entropy measures the distribution between elements over a given set and calculates the expected value based on the distribution. For our use case, this approach, using two decoders, could be highly beneficial because the critical data is stored in the cells. It is needed to first understand the headers to know what should be stored in the database. Recognizing a cell's content immediately after identifying the structure ensures that all maneuver attributes get captured before extracting the data from the corresponding cell. This allows us to parallelize the system by simultaneously searching for the structure where the maneuver data will be stored while starting to populate the database with the data right after recognition. However, it is needed to decide whether it is necessary to store the merged data back into an HTML file or if it should be skipped this step and store the data directly in our database. Next, it should get compared this approach with another table detection method by analyzing the paper "Table Detection Using Deep Learning" (Gilani et al., 2017 [4]). They present a two-module approach: image transformation and table detection. They explain, "Image transformation

is applied to separate these regions, while the table detection module uses Faster R-CNN as a basic element of the deep network" (Gilani et al., 2017, p. 2 [4]). Faster R-CNN relies on the combination of a Region Proposal Network (RPN) and Fast R-CNN. Image transformation is a crucial step in converting document images into natural images, making it easier to fine-tune existing R-CNN models. The distance transformation creates a derived representation of the digital image, calculating the precise distance between text regions and white spaces to estimate the table's location. For image transformation, they use a binary image as input and compute the Euclidean, linear, and max distance transforms on each RGB channel of the image. For table detection, they note that "Faster R-CNN was originally proposed for object detection and classification in natural images" (Gilani et al., 2017, p. 3 [4]). The table detection process is a unified network for object detection, starting with the RPN, which takes the transformed image as input and returns a set of rectangular object proposals with an objectness score. Unlike Zhong et al.'s approach, where the modules run separately and are task-specific, in this model, the "RPN shares a common set of convolutional layers with the detector module of the Faster-RCNN" (Gilani et al., 2017, p. 3 [4]). To generate region proposals, they developed a small network that slides over the convolutional feature map output by the last shared feature map. The "RPN takes an n x n spatial window of the input convolutional feature map as input" (Gilani et al., 2017, p. 3 [4]) and maps each sliding window to a lower-dimensional feature. Next, "The feature map is then passed to two fully connected layers that include a regression layer and a classification layer" (Gilani et al., 2017, p. 3 [4]), with these layers shared across all spatial locations. The architecture includes an n x n convolutional layer followed by a 1 x 1 convolutional layer for regression and classification. For each sliding window, Faster R-CNN predicts multiple regions at each location, with region proposals parametrized into reference boxes known as anchors. Faster R-CNN generates region proposals that are scale and translation-invariant. The RPN is trained end-to-end using Stochastic Gradient Descent (SGD) and backpropagation. After training, the detection network generates region proposals, which are passed to the region-based object detection CNN module to utilize these proposals. The detection module is a unified network composed of RPN and Fast R-CNN with shared convolutional layers. It detects tables from the test set and returns the coordinates of the bounding boxes for the predicted tables. This model differs from Zhong et al.'s [12] approach. However, for our decision-making process, especially considering accuracy, where they outperformed other models like Tesseract in three categories, it should be preferred the approach of Zhong et al. [12] They had better results in comparison, but during implementation, it is needed to evaluate how it fits into the OCR-D tool for character recognition. It may also be possible to improve their module for table or region detection since the OCR-D module is modular and extendable. First of all there is another approach to consider, because in recent years, table detection has improved and evolved due to advances in deep learning. Consequently, a new approach should be considered, with particular attention given to the study "PubTables-1M: Towards Comprehensive Table Extraction from Unstructured Documents" (Smock et al., 2021 [11]). In

their approach, they use data-driven methods based on deep learning for table extraction and define the following three subtasks: table detection (TD) for locating the table, table structure recognition (TSR), which identifies the structure of the table - such as rows, columns, and cells - and functional analysis (FA) for recognizing keys and values in the table. They believe that deep learning methods "(...) can learn to be more robust to the wide variety of table presentation formats" (Smock et al., 2021, p.1 [11]), which is essential given the high variability and lack of strict formatting for tables. For the proposed model, they mention that it "models TSR and FA jointly using six object classes: table, table column, table row, table column header, table projected row header, and table spanning cell" (Smock et al., 2021, p.7 [11]). By intersecting the pairs of table columns and table row objects, they implicitly created a seventh class, the table grid cell. Using these objects, they model a hierarchical structure based on physical overlap. This means that in the TSR and FA model, bounding boxes are extended for adjacent objects until they meet halfway, filling the empty space between these objects. This process is applied to objects from other classes, and afterward, there are no gaps or overlaps between rows, columns, and cells. For all three tasks in table extraction (TE), they apply Detection Transformer (DETR) to better demonstrate their proposed dataset and object detection modeling approach. In their pipeline, they use two different DETR models: one for TD and one for TSR and FA. In addition to DETR, they trained a Faster R-CNN model to compare results, contrasting the new deep learning approach with previous approaches using Faster R-CNN. They also note that "All models use a ResNet-18 backbone pretrained on ImageNet with the first few layers frozen" (Smock, 2021, p. 7 [11]). They avoided over-engineering the models and training procedures for each task to allow the data to drive the results. In their review process, they report that "For table detection, DETR slightly outperforms Faster R-CNN on AP but significantly outperforms on AP" (Smock, 2021, p. 7 [11]). For evaluation, they differentiate between Non-Canonical Data and Canonical Data. Canonicalization corrects oversegmentation in table structure annotation by, for example, merging adjacent cells under specific conditions like blank spaces. They mention that "(...) canonical data significantly improves performance for TSR models" (Smock, 2021, p.8 [11]). Based on the results, which show DETR outperforming Faster R-CNN in each tested task, this approach should be considered as the preferred solution.

3 Conception

In this section of the thesis, the conceptual evaluation of tools and data models will be conducted to justify their selection and identify suitable solutions that meet the requirements set by the DLR. Additionally, the flight test cards used in this case study will be analyzed in greater detail to convert them into an appropriate data model. The rationale behind the choice of this data model will also be explained.

3.1 Designing a pipleine for processing and storing flight test card data

This section will discuss suitable parsing and extraction methods for PDF files, aligning them with the structural requirements of the flight test cards. Additionally, the suitability of different database types will be evaluated based on the specific needs of the DLR.

3.1.1 Evaluation of PDF parsing methods

The flight test cards are all stored as PDF files in the DLR, and there is a need to store this data in a database. Consequently, it is necessary to extract the data from the PDF files and convert it into machine-readable formats. As mentioned in the section on PDF parsing, this requires the use of optical character recognition (OCR). Additionally, due to the structure of the flight test cards, which store maneuver data in tables, table detection is also essential. The OCR model selected for this thesis is the OCR-D tool, as outlined in the theory and state-of-the-art section. The OCR-D tool is well-suited for this task because it is an open-source module, which is particularly advantageous. Most OCR algorithms are designed for continuous text, but in this thesis, the majority of the content is structured text. Continuous text refers to text that is written in a flowing, uninterrupted manner, such as the text found in books or letters. In contrast, structured text is organized into predefined formats, such as tables, boxes, or other layouts that provide a clear structure to the content. This means that if improvements to the OCR are needed, the tools can be adapted for structured text. Another advantage of OCR-D is its modular construction, which allows for the addition of a separate table detection module to the OCR-D pipeline. This is crucial for accurately recognizing the text, which is often arranged in tables, and could significantly improve the quality of the results provided by the OCR. OCR-D is also based on Tesseract, a well-known and extensively documented OCR tool, making it easier to run and troubleshoot during the development process. Furthermore, the developers of OCR-D have enhanced the stability, code quality, and performance of the Tesseract OCR, providing a solid foundation for its use in this thesis. These enhancements include an

improved post-correction model, which delivers more accurate character and word coordinates. This is particularly beneficial because flight test cards are handwritten during flights, often resulting in illegible and poorly structured text. More precise character and word recognition, along with improved coordinates, helps mitigate these issues. As described in Chapter 2, the parsing pipeline is implemented in Python, facilitating integration with the DLR infrastructure for storing measurement data. This choice is justified because twinstash, the system used to store flight measurement data, provides a Python API. This API simplifies the process of linking the data extracted from the flight test cards to the measurement system data, ensuring seamless integration. Another important aspect of tooling for PDF parsing that should be evaluated for this thesis is table detection. As already mentioned above, most of the content relevant to this thesis is stored in tables. Given this table structure, the OCR should be integrated with an accurate table detection algorithm to achieve more precise results from the flight test card PDF files. In this thesis, it is recommended to use the tool developed by Smock et al. [11] This recommendation is based on its innovative approach, which employs a data-driven method leveraging deep learning for extracting table content. Another advantage of this tool is that it is open-source, meaning it can be easily utilized for this thesis. Additionally, it offers high variability and works with different table formats, which supports the diverse requirements stemming from the high variability of flight test cards. This variability arises from the different tasks that need to be performed and the varying styles used by individuals creating the flight test cards. Therefore, using OCR-D combined with the Table Transformer module from Smock et al. could provide good and flexible results. Based on the modular concept of OCR-D, it should be possible to integrate these two modules to detect various types of tables and extract the data from them effectively.

3.1.2 Assessment of a suitable database type

For this thesis, a relational database will be used, justified by the analysis in Chapter 2. The DLR requires fast data storage and retrieval for analytics, making relational databases suitable due to create summarization of the flight which flight is containing which maneuver, retrieval, and reporting features. A key goal is linking measurement data with flight test cards, enabling searches for maneuvers and flight aspects not directly captured in the data. The structured storage in relational tables enhances system predictability and aligns with the structured format of the original flight test cards. Additionally, the database offers efficient search capabilities across multiple cards, improving DLR workflows without altering data structure. The relational format also ensures consistency, allowing comparisons across different projects and engineers by merging diverse flight test card types into a unified structure. The thesis evaluates relational database management systems (RDBMS) using a review matrix based on project needs, prioritizing factors like open-source status, OS compatibility, Python integration, datatypes and scalability. Open-source solutions are preferred for cost-effectiveness and maintainability but have medium

priority. OS compatibility has low priority but is preferred for local operation during testing. Python integration is a high priority to streamline engineering and reporting, as the pipeline and web interface backend are Python-based in the DLR, to match the systems of the DLR it would be important to use Python. Using the Python integration it is possiblie to connect to the database and manipulate data with python code running SQL. The database must support datatypes like strings, integers, floats, datetime, and binary data to store flight test cards fully. Scalability is medium priority, ensuring future storage needs are met. The RDBMS compared are Oracle, MySQL, and PostgreSQL. Both MySQL and PostgreSQL are open-source, while Oracle is closed-source. All three systems run on Windows and Linux and support Python integration through third-party tools. Each supports required data types, though MySQL offers direct binary file storage, unlike PostgreSQL, which stores binary data as arrays requiring conversion. Both vertical and horizontal scaling are supported by all three systems. While scalability mechanisms are provided, extensive scaling isn't needed for this thesis. Based on the review, MySQL is the preferred solution. While PostgreSQL and MySQL meet most priorities, MySQL's ability to store binaries directly gives it an edge, aligning with the high-priority requirement for datatypes flexibility and that it is open-source and free to use. Thus, MySQL offers the best overall solution for the thesis needs.

Category	Priority
Open-Source	Medium
Operating System	Low
Python Integration	High
Datatypes	High
Scalability	Medium

Figure 3.1: Relation Database Management System Reviewing Matrix

3.2 Development of a data model for maneuvers

This section provides additional details about the data stored in the flight test cards and presents a conceptual design for a possible data model to store the information relevant to the DLR. In this case study, the focus is primarily on the maneuver data.

3.2.1 Case Study Flight Test Card Content

In this subsection, the content of the flight test card should be further discussed and evaluated to determine which data from the flight test card is important for this thesis, as the first step will focus on the data in the flight test maneuver tables.

Pre-flight Information and Reference Values

As already mentioned in the section on the structure of the flight test card, it begins with a header containing information about the flight. This part could be relevant for this thesis, as we must match and link the data from the measurement system with the data from the flight test card. Therefore, we could use information from the header, such as the run and date of the flight test, which could also be stored as the flight test name. An example of a naming of a flight could be found in figure 3.2 this structure is need for the matching flight test card data with the measurement data.The section containing the flight crew includes the PIC, copilot (COP), lead flight test engineer, technical crew, and additional crew members, such as task specialists for the flight test. This information will be ignored due to privacy protection for crew members. The next section contains data on loadings and fuel. Load data includes the zero fuel weight (ZFW), zero fuel weight center of gravity (ZFW CG), total weight (TOW), and total weight center of gravity (TOW CG). The fuel section includes wing left-hand fuel (WING LH), wing right-hand fuel (WING RH), and total fuel. For this case study, these values will be ignored, but for future work, they could be useful as they may help scientists better understand the flight conditions, which is a key argument for improving the understanding of flight tests in this thesis. This can be disregarded, as the focus of this thesis is on extracting maneuver data from the flight test cards, and this particular data is not directly related to the maneuvers. The Takeoff (T/O) section stores configuration data such as wind speed (WIND), Outside air Temperatur/Dew Point. QNH signifies the atmospheric pressure adjusted to mean sea level. This calibration is used by pilots to set their aircraft's altimeter, anti-ice status, the anti-ice status indicates whether the anti-ice system is active, helping to prevent icing on the aircraft, flaps, this is a movable portion of the wing which can produce extra lift by lowering, runway condition (RWY Condition), and takeoff speeds like V1, VR, V2, and VREF.

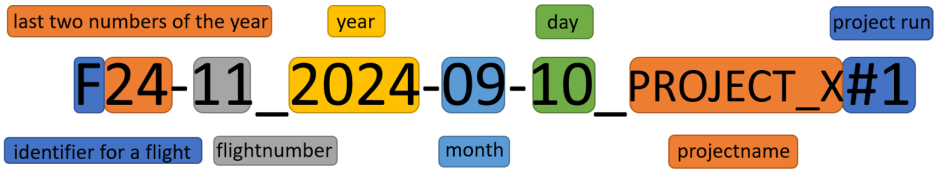


Figure 3.2: Measurement data flightname structure

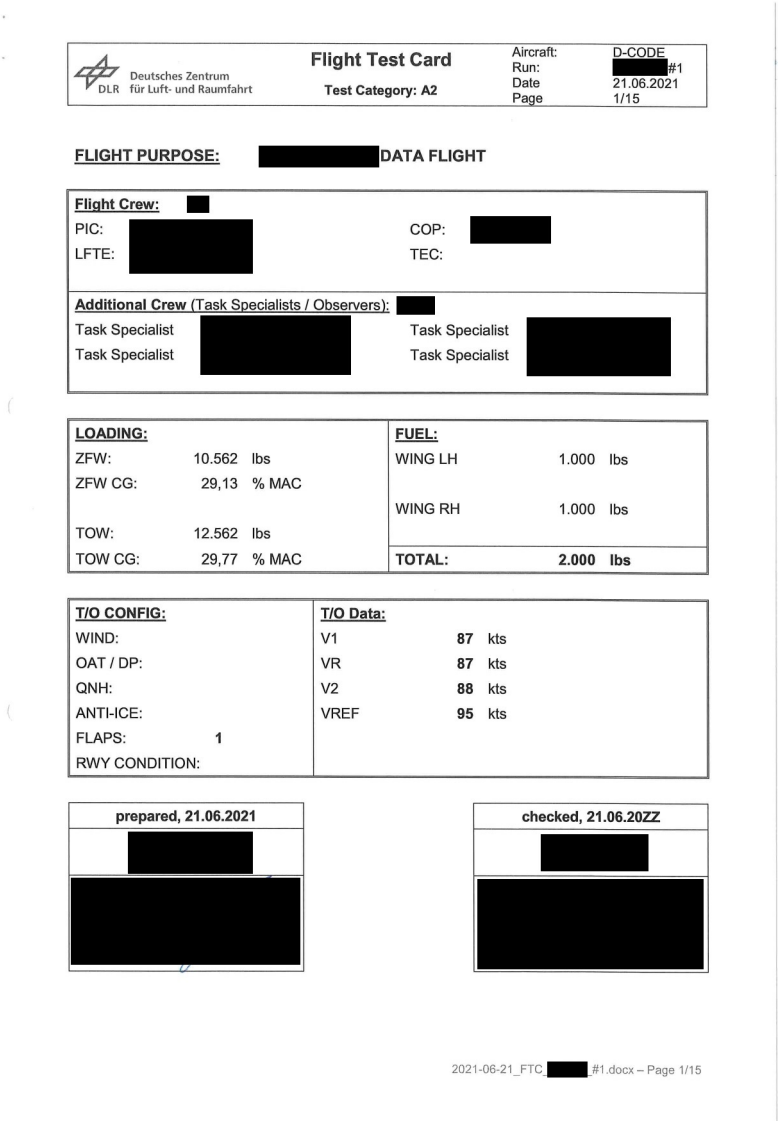
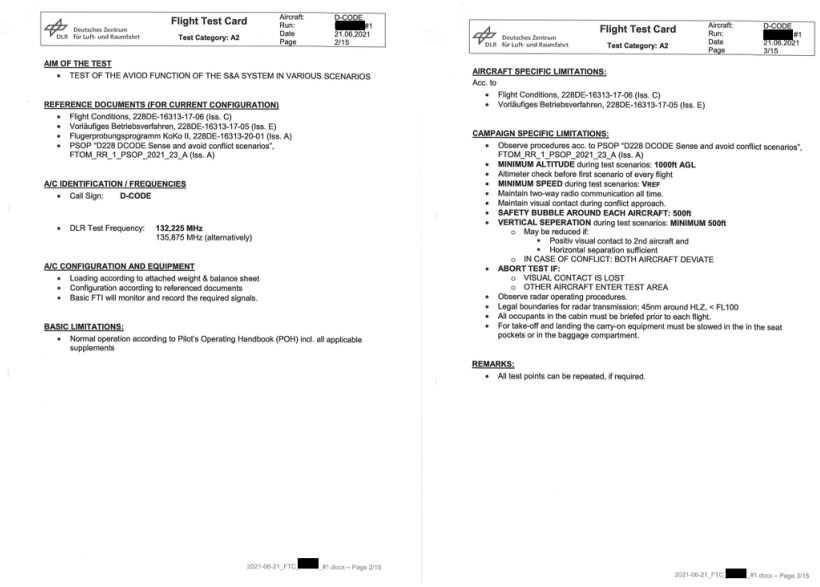


Figure 3.3: Pre-flight Information and Reference Values Flight Test Card Example

Flight Limitations and Aim

Following this, there is a section that outlines the test's aim, defining the goal of the flight test and referencing all related documents, including flight conditions and scenario doc-

umentation. The next section covers aircraft (A/C) identification and communication frequencies, including the call sign and test frequencies for the communication during the flight test. This is followed by a section containing the A/C configuration and equipment, such as loading and configuration details, and whether the measurement system is required and recording. After the aircraft information, operational limitations are defined, divided into basic limitations (which generally refer to the Pilot's Operating Handbook, or POH), aircraft-specific limitations (which refer to the aircraft's flight conditions and operating procedures), and campaign-specific limitations, which may include minimum altitude, minimum speed, or safety advisories like a safety bubble around each aircraft and test-aborting conditions. Following the limitations section, remarks are provided for the test, which may include instructions on whether test points can be repeated.



Deutsches Zentrum DLR für Luft- und Raumfahrt

Flight Test Card

Test Category: A2

Aircraft: D-CODE
Run: [redacted]#1
Date 21.06.2021
Page 2/15

AIM OF THE TEST

- TEST OF THE AVIOD FUNCTION OF THE S&A SYSTEM IN VARIOUS SCENARIOS

REFERENCE DOCUMENTS (FOR CURRENT CONFIGURATION)

- Flight Conditions, 228DE-16313-17-06 (Iss. C)
- Vorläufiges Betriebsverfahren, 228DE-16313-17-05 (Iss. E)
- Flugerprobungsprogramm KoKo II, 228DE-16313-20-01 (Iss. A)
- PSOP "D228 DCODE Sense and avoid conflict scenarios", FTOM_RR_1_PSOP_2021_23_A (Iss. A)

A/C IDENTIFICATION / FREQUENCIES

- Call Sign: **D-CODE**
- DLR Test Frequency: **132,225 MHz** 135,875 MHz (alternatively)

A/C CONFIGURATION AND EQUIPMENT

- Loading according to attached weight & balance sheet
- Configuration according to referenced documents
- Basic FTI will monitor and record the required signals.

BASIC LIMITATIONS:

- Normal operation according to Pilot's Operating Handbook (POH) incl. all applicable supplements

2021-06-21_FTC_[redacted]_#1.docx – Page 2/15

Deutsches Zentrum DLR für Luft- und Raumfahrt

Flight Test Card

Test Category: A2

Aircraft: D-CODE
Run: [redacted]#1
Date 21.06.2021
Page 3/15

AIRCRAFT SPECIFIC LIMITATIONS:

Acc. to

- Flight Conditions, 228DE-16313-17-06 (Iss. C)
- Vorläufiges Betriebsverfahren, 228DE-16313-17-05 (Iss. E)

CAMPAIGN SPECIFIC LIMITATIONS:

- Observe procedures acc. to PSOP "D228 DCODE Sense and avoid conflict scenarios", FTOM_RR_1_PSOP_2021_23_A (Iss. A)
- **MINIMUM ALTITUDE** during test scenarios: **1000ft AGL**
- Altimeter check before first scenario of every flight
- **MINIMUM SPEED** during test scenarios: **VREF**
- Maintain two-way radio communication all time.
- Maintain visual contact during conflict approach.
- **SAFETY BUBBLE AROUND EACH AIRCRAFT: 500ft**
- **VERTICAL SEPERATION** during test scenarios: **MINIMUM 500ft**
  - May be reduced if:
    - Positiv visual contact to 2nd aircraft and
    - Horizontal separation sufficient
  - IN CASE OF CONFLICT: BOTH AIRCRAFT DEVIATE
- **ABORT TEST IF:**
  - VISUAL CONTACT IS LOST
  - OTHER AIRCRAFT ENTER TEST AREA
- Observe radar operating procedures.
- Legal boundaries for radar transmission: 45nm around HLZ, < FL100
- All occupants in the cabin must be briefed prior to each flight.
- For take-off and landing the carry-on equipment must be stowed in the in the seat pockets or in the baggage compartment.

REMARKS:

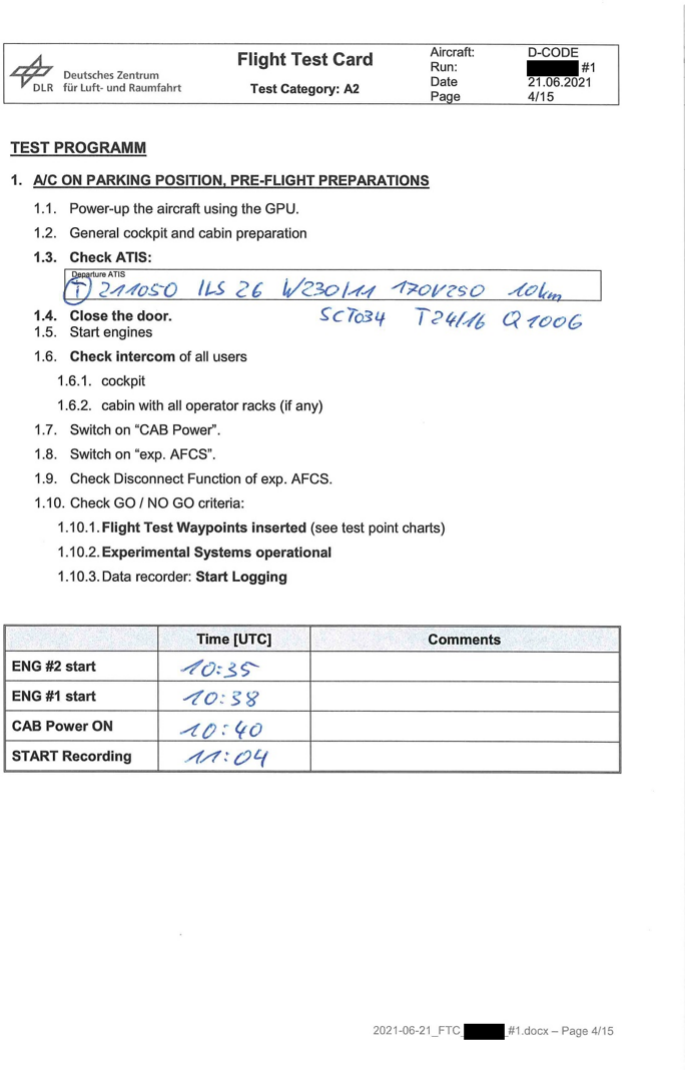
- All test points can be repeated, if required.

2021-06-21_FTC_[redacted]_#1.docx – Page 3/15

Figure 3.4: Flight Limitations and Aim Flight Test Card Example

Aircarft on parking position, pre-flight preparation

These pre-flight and safety remarks are followed by the test program, beginning with "A/C on parking position, pre-flight preparations." This section primarily describes procedures for starting the aircraft, such as powering it up using the GPU, which is the ground power unit, general cockpit and cabin preparation, and the ATIS check, which is manually recorded by the flight test engineer. This is followed by the remaining preparation steps, and at the end, the section includes a table with start information, such as engine starts, cabin power-on, and recording start, with columns for UTC time and comments on any remarks made during the start procedure.



Deutsches Zentrum für Luft- und Raumfahrt (DLR)

Flight Test Card

Test Category: A2

Aircraft: D-CODE
Run: [redacted] #1
Date 21.06.2021
Page 4/15

TEST PROGRAMM

1. **A/C ON PARKING POSITION, PRE-FLIGHT PREPARATIONS**
   - 1.1. Power-up the aircraft using the GPU.
   - 1.2. General cockpit and cabin preparation
   - 1.3. **Check ATIS:**

     Departure ATIS
     T 211050 ILS 26 W230/11 170V250 10km
     SCT034 T24/16 Q1006
   - 1.4. **Close the door.**
   - 1.5. Start engines
   - 1.6. **Check intercom** of all users
     - 1.6.1. cockpit
     - 1.6.2. cabin with all operator racks (if any)
   - 1.7. Switch on "CAB Power".
   - 1.8. Switch on "exp. AFCS".
   - 1.9. Check Disconnect Function of exp. AFCS.
   - 1.10. Check GO / NO GO criteria:
     - 1.10.1. **Flight Test Waypoints inserted** (see test point charts)
     - 1.10.2. **Experimental Systems operational**
     - 1.10.3. Data recorder: **Start Logging**

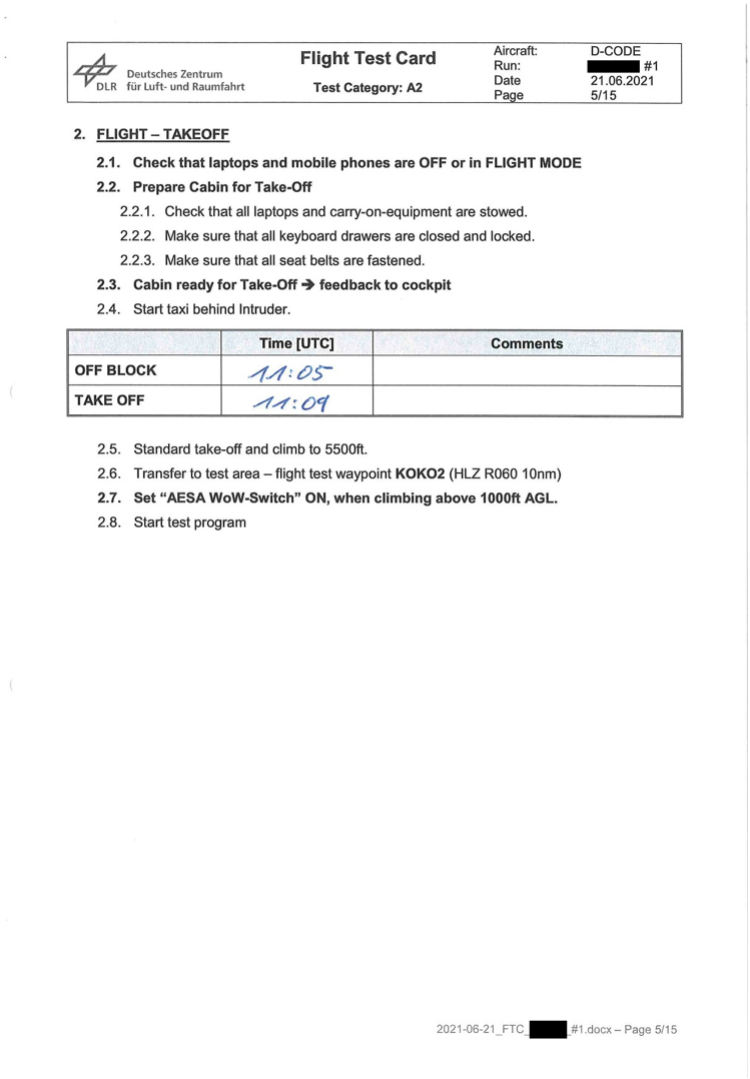
	Time [UTC]	Comments
ENG #2 start	10:35	
ENG #1 start	10:38	
CAB Power ON	10:40	
START Recording	11:04	

2021-06-21_FTC_[redacted]#1.docx – Page 4/15

Figure 3.5: Aircarft on parking position, pre-flight preparation Flight Test Card Example

Takeoff

The Takeoff section contains additional machine-written tasks to be completed, such as "Check that laptops and mobile phones are OFF or in FLIGHT MODE." Off-block and takeoff times are stored in a table, which is filled out by the flight test engineer during the procedure, with columns for time and comments.



Deutsches Zentrum für Luft- und Raumfahrt (DLR)

Flight Test Card

Test Category: A2

Aircraft: D-CODE
Run: [redacted] #1
Date 21.06.2021
Page 5/15

2. FLIGHT – TAKEOFF

2.1. Check that laptops and mobile phones are OFF or in FLIGHT MODE

2.2. Prepare Cabin for Take-Off

2.2.1. Check that all laptops and carry-on-equipment are stowed.

2.2.2. Make sure that all keyboard drawers are closed and locked.

2.2.3. Make sure that all seat belts are fastened.

2.3. Cabin ready for Take-Off → feedback to cockpit

2.4. Start taxi behind Intruder.

	Time [UTC]	Comments
OFF BLOCK	11:05	
TAKE OFF	11:09	

2.5. Standard take-off and climb to 5500ft.

2.6. Transfer to test area – flight test waypoint **KOKO2** (HLZ R060 10nm)

2.7. Set "AESA WoW-Switch" ON, when climbing above 1000ft AGL.

2.8. Start test program

2021-06-21_FTC_[redacted]#1.docx – Page 5/15

Figure 3.6: Takeoff Section Flight Test Card Example

Maneuver

After this, the section containing the actual flight test, which is very important for this thesis, begins. This section provides a detailed overview of the maneuvers, first listing the maneuver name and type, such as "Radial Approach, 90 degrees," followed by a description and image of the maneuver, with a detailed procedure in machine-written text. Following this is the definition of the test points, including the test point number, aircraft name, and parameter values required for each test point, such as altitude, speed, distance, and radial values, as well as these parameters for the intruder aircraft, the intruder is the aircraft which is flying in the same airspace with the main aircraft in the project which is used for the case study; therefore they have also defined parameters for the second aircraft. After this definition, a table should be filled out by the flight test engineer during the

flight, with columns for test point number, time (including start, avoid, and stop), and remarks for each test point. Each maneuver also includes waypoints, listing waypoint name, reference, latitude, and longitude, all in machine-written text. Other maneuvers are similar, containing different parameters or omitting intruder aircraft, but all include test points with time, parameters, and remarks; however, not all maneuvers contain waypoints. This thesis aims to create a data model for this structure and extract this information from the document. A key issue could be the lack of standardization for flight test cards, which may vary between flight test engineers and tests.

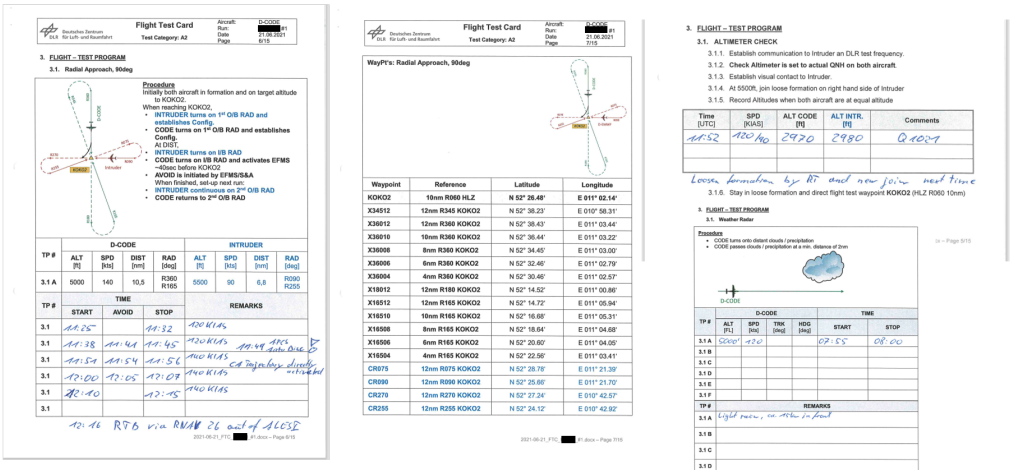
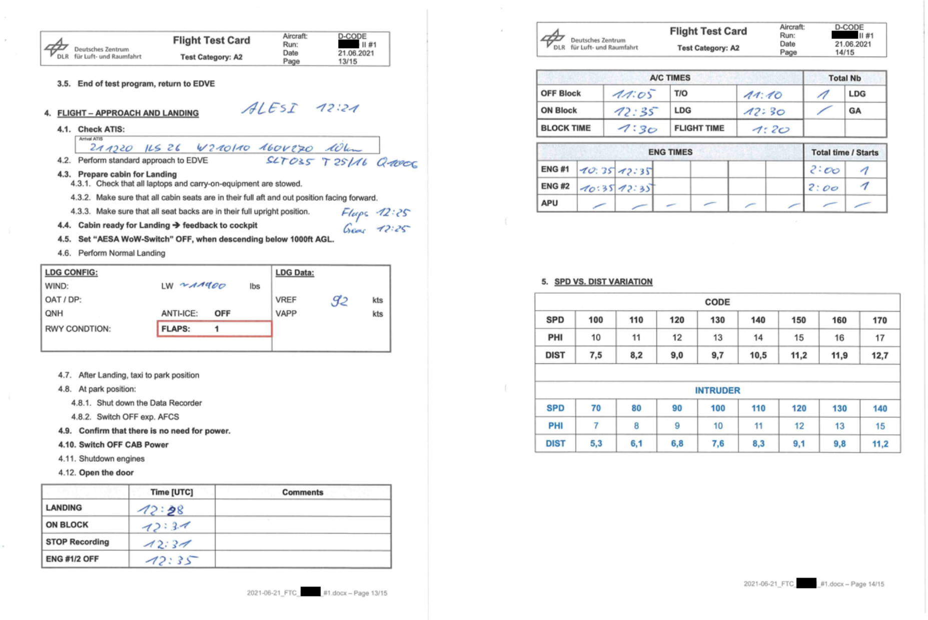


Figure 3.7: Flight test card maneuver Section Flight Test Card Example

Approach and Landing

After the flight test section, the approach and landing section follows, beginning with the Automatic Terminal Information Service (ATIS) check, this means that the crew gets some basic automatic information like the name of the airstrip, active runway, the wether at the airstrip and airstrip conditions, which is contained in a box and filled out by the flight test engineer. This is followed by procedures for landing, which are machine-written. For landing, the landing configuration (LDG config) is recorded as handwritten data in a designated box. It includes information similar to the values from the departure section but reflects the conditions at the time of landing. As with the start procedure, the landing time, on-block time, stop recording, and engine shutdown times are stored in a table with columns for time and comments. This is followed by total aircraft statistics, including off-block time, on-block time, block time. The block time is the time when the crew started and arrived back at the parking position. Takeoff time, landing time (LDG), flight time, landing count, and go-arounds (GA). This section also contains a table for engine

(ENG) statistics, such as engine start times, APU time, total time, and start count. The next section covers speed (SPD) and distance (DIST) variations for the D-CODE and intruder aircraft, containing all SPD variations linked with PHI and DIST.



Deutsches Zentrum für Luft- und Raumfahrt DLR

Flight Test Card

Test Category: A2

Aircraft: D-CODE
Run: [redacted] II #1
Date 21.06.2021
Page 13/15

3.5. End of test program, return to EDVE

4. FLIGHT – APPROACH AND LANDING ALESI 12:21

4.1. Check ATIS:

Arrival ATIS: 211220 ILS 26 W210/10 160V270 10km

4.2. Perform standard approach to EDVE SCT035 T25/16 Q1006

4.3. Prepare cabin for Landing

4.3.1. Check that all laptops and carry-on-equipment are stowed.

4.3.2. Make sure that all cabin seats are in their full aft and out position facing forward.

4.3.3. Make sure that all seat backs are in their full upright position. Flaps 12:25

4.4. Cabin ready for Landing → feedback to cockpit Gear 12:25

4.5. Set "AESA WoW-Switch" OFF, when descending below 1000ft AGL.

4.6. Perform Normal Landing

LDG CONFIG:			LDG Data:		
WIND:	LW	~11900 lbs			
OAT / DP:			VREF	92	kts
QNH	ANTI-ICE:	OFF	VAPP		kts
RWY CONDTION:	FLAPS:	1			

4.7. After Landing, taxi to park position

4.8. At park position:

4.8.1. Shut down the Data Recorder

4.8.2. Switch OFF exp. AFCS

4.9. Confirm that there is no need for power.

4.10. Switch OFF CAB Power

4.11. Shutdown engines

4.12. **Open the door**

	Time [UTC]	Comments
LANDING	12:28	
ON BLOCK	12:31	
STOP Recording	12:31	
ENG #1/2 OFF	12:35	

2021-06-21_FTC_[redacted]_#1.docx – Page 13/15

Deutsches Zentrum für Luft- und Raumfahrt DLR

Flight Test Card

Test Category: A2

Aircraft: D-CODE
Run: [redacted] II #1
Date 21.06.2021
Page 14/15

A/C TIMES				Total Nb	
OFF Block	11:05	T/O	11:10	1	LDG
ON Block	12:35	LDG	12:30	/	GA
BLOCK TIME	1:30	FLIGHT TIME	1:20		

ENG TIMES							Total time / Starts	
ENG #1	10:35	12:35					2:00	1
ENG #2	10:35	12:35					2:00	1
APU	–	–	–	–	–	–	–	–

5. SPD VS. DIST VARIATION

CODE								
SPD	100	110	120	130	140	150	160	170
PHI	10	11	12	13	14	15	16	17
DIST	7,5	8,2	9,0	9,7	10,5	11,2	11,9	12,7
INTRUDER								
SPD	70	80	90	100	110	120	130	140
PHI	7	8	9	10	11	12	13	15
DIST	5,3	6,1	6,8	7,6	8,3	9,1	9,8	11,2

2021-06-21_FTC_[redacted]_#1.docx – Page 14/15

Figure 3.8: Approach and Landing Section Flight Test Card Example

Weight and Balance

On the final page, a diagram lists items in the aircraft with their weight, lever arm of the single elements based on their postion in the aircraft, and moment, as well as a diagram for the empty weight center of gravity.

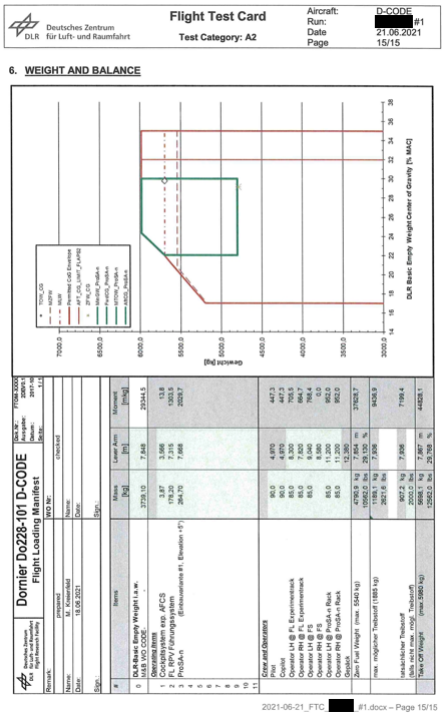


Figure 3.9: Weight and Balance Section Flight Test Card Example

3.2.2 Design of the selected data model

Based on the results of the section about the theory and the state of the art, the data model will be created using these findings. First of all, the data model should be implemented in a relational database. For this thesis a relational model was choosen, this is reasoned in the ability to query data efficiently. Another advantage of the relational model is that they provide a fixed structure through the useage of tables to store the data, which is not given by non-relational models. To ensure this, it is crucial to maintain data consistency to achieve accurate results. To create a relational model, it is essential to identify entities and relationships that can capture the different types of maneuvers of the flight test cards analyzed in this thesis. This is accomplished through a case study of two different projects and their flight test cards. The model will include all entities necessary for these two types of flight test cards but can likely be adapted for use with other flight test cards as well.

Flight test table

Based on the structure of the flight test card, there will be an entity named flight test with a unique key called name. This is justified because there will always be only one flight with a specific name, which includes the project name, the total flight number of the year, the date, and the flight number within the project. The name is a combination of the Run and the Date, as shown in Figure 3.10. This information is not stored separately

in the data model because this combination allows connecting the flight test card with the data name from the measurement system. This system also uses the combination of the run and date, ensuring consistent naming conventions. As the run and date are merged into the flight name, scientists can search for it without needing separate fields, resulting in consistent naming. The date, project, and flight name could technically be stored as separate attributes. However, in the context of the DLR use case, searches are rarely performed for the date alone. Most of the time, users search for the name of the flight test data to retrieve associated maneuvers from that flight. Therefore, to align with the structure of the flight test data database and facilitate seamless merging of content between the flight test data database and the flight test card database, the information should be stored in the same format as the flight test data.

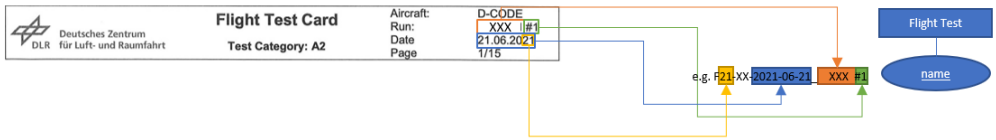


Figure 3.10: Flight Test Cards Header including Flight Test Name and the storing in the data model

Maneuver tables

The flight test entity is connected to Maneuvers, as a flight test card can include various maneuvers. The Maneuver entity contains fields for a unique maneuver_id, name, image, and procedure. The maneuver_id is essential for uniquely identifying each maneuver, as names may be repeated with different procedures. The image and procedure fields are optional because not all flight test cards include an image or a detailed procedure description. For storing different maneuvers, the Maneuver entity is linked to a radar table and an intruder maneuver table, both of which are disjoint relations. This means that the linked tables inherit from the maneuver table because they represent maneuvers as well. However, to store them correctly, additional information specific to their context must be appended. This is achieved by extending the data through the radar and intruder maneuver tables, which include the necessary attributes to distinguish and handle these specialized types of maneuvers. This design leverages the advantages of a relational database by organizing parameter data for each maneuver in a structured and consistent format. The radar table provides additional information, such as track (trk) and heading (hdg). The intruder maneuver table includes fields for intruder_alt, intruder_spd, and intruder_ias. These parameters are the minimum required to describe an altimeter check maneuver involving an intruding aircraft, ensuring that only the essential data is included compared to other intruder maneuvers. To avoid multiple NULL values in the tables for radial and maneuvering approaches, another disjoint relation is introduced. This relation also inherits from the intruder maneuver table because, as with the maneuvers mentioned above, additional information specific to these approaches must be stored. At the same

time, the information from the intruder maneuver table remains necessary, ensuring that the shared attributes are preserved while extending the schema to accommodate the specific needs of radial and maneuvering approaches. These approaches also require fields such as intruder distance (intruder_dist), distance (dist), and radial position (rad) of the main aircraft. To further minimize NULL values, the radial maneuvering table is split into two separate tables: one for radial maneuvers and another for maneuvering maneuvers. The radial maneuver table additionally includes the intruder_rad field. Meanwhile, the maneuvering table stores parameters such as ground speed (g_spd), intruder ground speed (intruder_g_spd), and intruder track (intruder_trk).

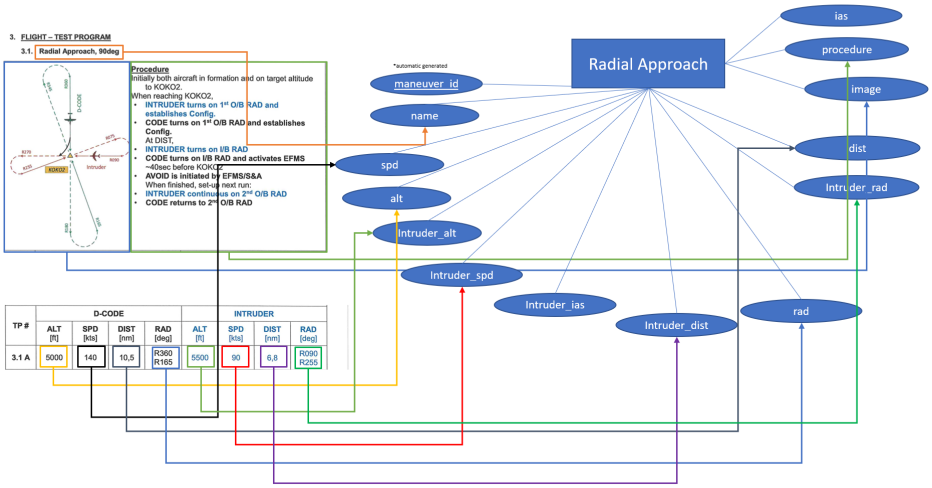


Figure 3.11: Radial Approach Example for storing maneuver data

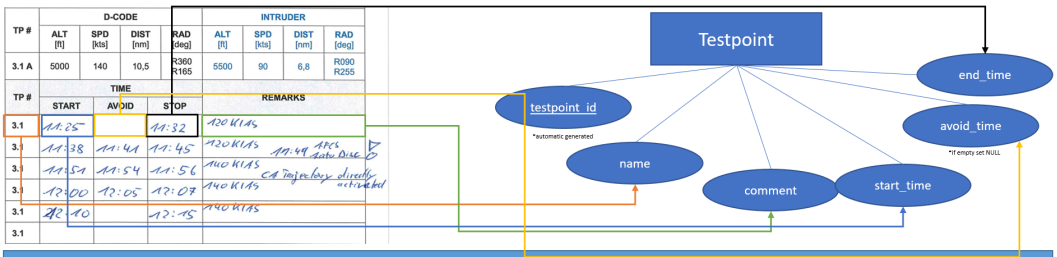
maneuver_id	name	image	procedure	alt	spd	ias	intruder_alt	intruder_spd	intruder_ias	rad	dist	intruder_dist	intruder_rad
1	Radial Approach, 90deg		Initially both aircarft in formation ...	5000	140	NULL	5500	90	NULL	360, 165	10.5	6.8	090, 255

Figure 3.12: Radial Approach data stored in a table

Test point table

The Maneuver entity is related to test points (TP). A single maneuver can have multiple test points, and these test points may repeat. To facilitate better searchability and organization, test points are stored as a unique entity. An example of repeated test points is shown in Figure 3.13, where test point "3.1 A" is flown five times, each instance accompanied by different remarks and time values. Due to this repetition, the test point name is not unique. Therefore, each test point is assigned a unique testpoint_id for proper identification. In addition to the unique identifier, the test point entity includes fields for name and comment. It also stores time-related fields, including start_time, end_time,

and avoid_time. While all these fields are optional, at least the start_time must be provided. Storing time values directly in the test point entity simplifies analysis of maneuver durations, as well as their start and end times. This structure allows for easy querying without the need for complex joins.



Testpoint

testpoint_id	name	comment	start_time	avoid_time	end_time
1	3.1	120 KIAS	11:25	NULL	11:32
2	3.1	120 KIAS 11:49 AFCS Auto Disc!	11:38	11:41	11:45
3	3.1	140KIAS CA Trajectory directly activated	11:51	11:54	11:56
4	3.1	140KIAS	12:00	12:05	12:07
5	3.1	140KIAS	12:10	NULL	12:15

Figure 3.13: Testpoint extracted form the flight test card and stored in a table

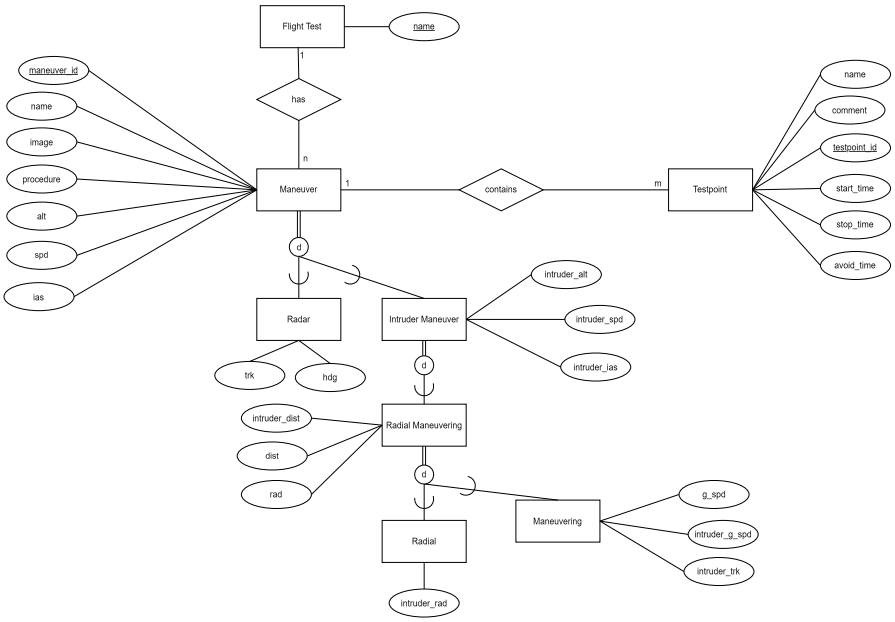


Figure 3.14: Entity Relationship Diagram for Flight Test Cards

3.3 Selection of the solution approach for the pipeline

The pipeline consists of various modules designed to integrate the previously mentioned components, including those described earlier. This section explains the rationale behind the selection of these modules and their role in the pipeline.

3.3.1 Converting modules

This subsection outlines the modules in the pipeline that handle the conversion of input or output data, enabling its reuse in subsequent processes and other modules. These modules also contribute to enhancing the results delivered by the utilized components.

PDF to image converter

The PDF-to-image converter is essential because the OCR-D module used for binarization requires images as input. This module converts the pages of a PDF file into individual images. As part of this process, it creates a dedicated directory for the image files and assigns descriptive filenames to ensure proper page order and source identification. Each filename includes the page number and flight name. The directory containing the images is then passed as input to the binarization module, as it is required for the creation of the binarization module's configuration file.

Recognized content to image converter

This module utilizes the bounding boxes provided by the table detection process to extract content from the PDF image. This step ensures that the OCR-D module processes smaller, focused input, making it easier to recognize text accurately. For example, each cell of a table is extracted as a separate image, allowing the recognition process to focus on the content of a single cell. Additionally, this module stores the positional and relational information of the cells and table elements, as this data is required in the post-recognition process to merge the extracted data and determine how the elements are interconnected.

3.3.2 Connection modules

This section provides an overview of the modules used to integrate the various components of the pipeline. These modules are responsible for merging the outputs of other modules to extract more detailed information, structuring the data for storage, and performing the actual storage of flight test card data. Additionally, they establish the connection between the flight test card data and the flight data stored in the twinstash system.

Merge relational information with OCR results

This module is responsible for merging the OCR-D results for individual inputs, such as a cell, with other elements on the page, as well as the positional information obtained from the table detection process. This merging is necessary to establish relationships between text content based on their positions, enabling the correct assignment of content to the corresponding maneuver or test point based on their alignment. For instance, if recognized content is positioned at the same height as other recognized content, they are considered related and assigned to the same test point. An example of this process is illustrated in Figure 3.15. The module takes the Page-XML output from OCR-D and the structural information from the table detection module as inputs, combining them to create a cohesive dataset.

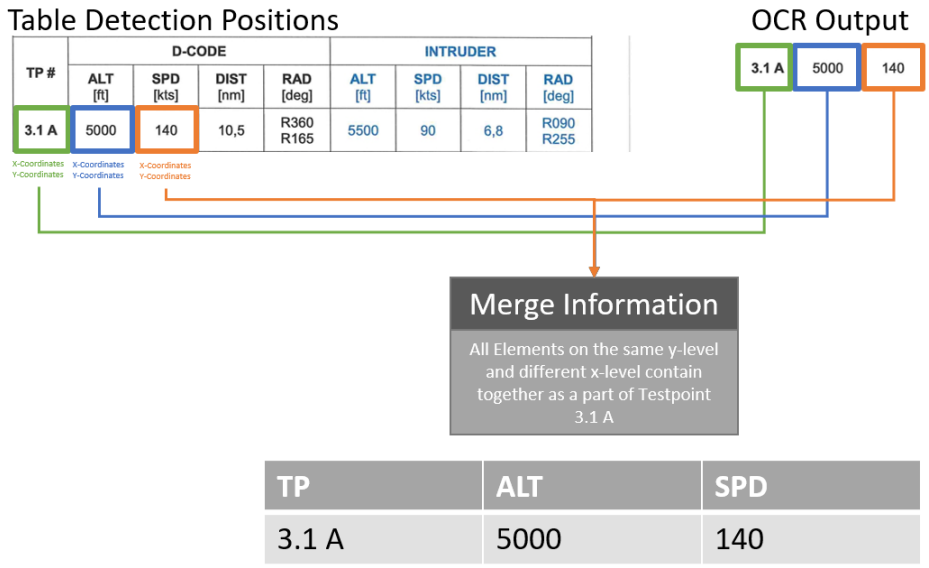


Figure 3.15: Conception for merging of relational information and ocr results

Storage of the flight test card data

The responsibility of the storage module for the flight test cards is straightforward. It reads the structure created during the merging process, extracts the content, and inserts the data into the database. This process involves starting with the creation of a flight test entry by adding its name to the flight test table. Next, the module creates the maneuvers associated with the flight and appends the corresponding test points along with their values, establishing the necessary relationships between the elements.

Connection to the flight test data

The connection to the flight test data is also straightforward. The tooling searches the TwinStash for an entry matching the name of the flight test. If a matching flight is found, it updates the user tags with information indicating that flight test card data exists in the database and provides details on how to locate the associated data.

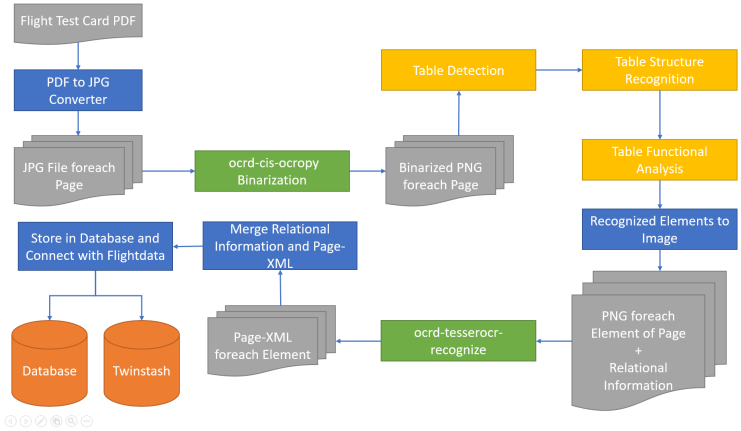


Figure 3.16: Workflow of the extraction pipeline

4 Implementation of the case study

This chapter outlines the implementation of various modules designed for extracting and storing data from flight test cards. It also explains how these modules are integrated into a pipeline for the automatic extraction of data and its storage in a database.

4.1 Implementation of the PDF Parsing Modules

This section covers the implementation of the parsing modules for the flight test cards, as well as the training of the models used for parsing. It provides an overview of the modules and their functionality. Additionally, the section describing the pipeline will detail how these modules are interconnected.

4.1.1 Implementation of OCR for data parsing

As mentioned in Chapter 3, the OCR-D framework will be used for parsing flight test cards. The OCR-D module is Unix-based, which led to the decision to implement it on an Ubuntu system. The Tesseract module within OCR-D supports character recognition but requires segmentation at the line level. This segmentation is provided as Page-XML input. The Page-XML file contains information about the segmented regions of the image, as well as the URL of the image to be processed for recognition. Consequently, before utilizing the recognition module, it is necessary to perform segmentation on the input data.

Binarizing of flight test cards

The first step in the pre-processing for text recognition is binarizing the input image. This is accomplished using the ocrd-cis-ocropy module, which generates a grayscale version of the input image along with a Page-XML file describing the pages contained in the image. For this case study, the Page-XML file contains minimal information, as each image corresponds to a single page. Grayscaling simplifies the recognition process by applying a threshold to remove unnecessary elements, such as background noise, and eliminates the need for the recognition module to handle colors. This preprocessing step enhances the accuracy of text recognition when applied to the flight test cards. The result of the binarization process is illustrated in Figure 4.1.

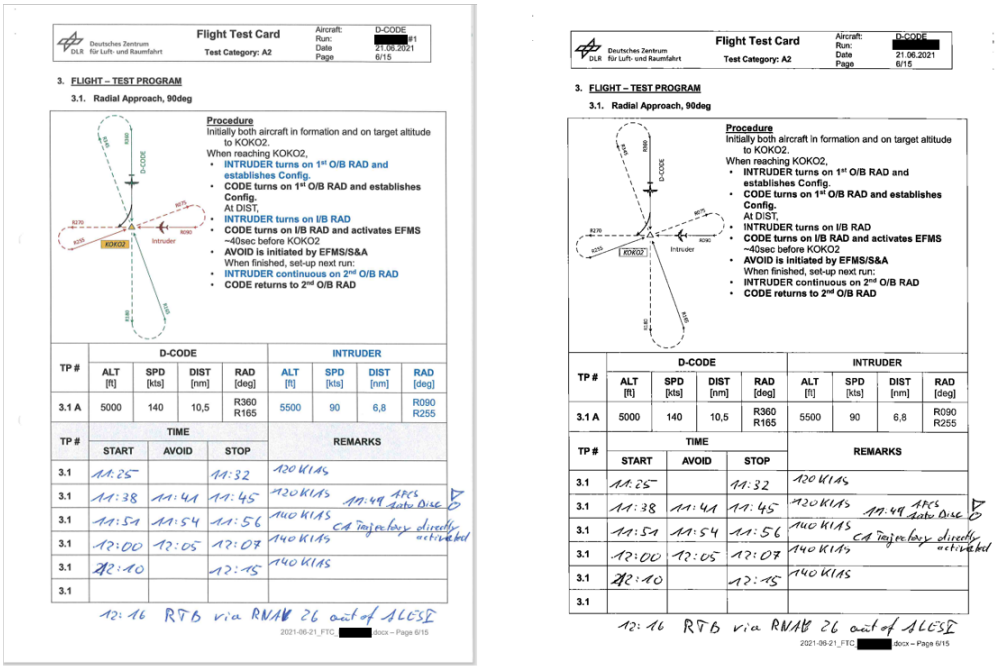


Figure 4.1: Example for one page of flight test cards as input (left) and output (right) of the binarization

Segmentation of the flight test cards

The segmentation process is divided into different parts that interconnect. First, the results from the binarization step are used for region-based segmentation. This process takes the binarized image and the corresponding Page-XML file as input and segments the image into distinct regions. This task is performed by the tesserocr-segment module. An example of a Page-XML file is shown in Figure 4.3. It includes the label for the text region, a unique ID for the region, the orientation, the reading direction, and the text line order within the text region. Additionally, it stores the coordinates of the text region as bounding box points. The output of this module, used for region analysis, includes an image with only text and essential content, while elements like table borders and other unnecessary lines are removed shown in figure 4.2 as well as the bounding boxes for the regions contained in the Page-XML. This step is crucial as it eliminates obstacles that could negatively impact the accuracy of text recognition, which is performed after the segmentation. The region segmentation is followed by line segmentation. This step uses the results from the region segmentation as input, relying on the bounding boxes provided in the Page-XML and the image. The Page-XML file contains additional information compared to the Page-XML generated during region-based segmentation. This includes the label and ID of each text line, as well as the coordinates of each text line represented as bounding boxes, as shown in Figure 4.4. It intersects the content within the bounding

boxes and produces a Page-XML file containing detailed information about the content of each bounding box.

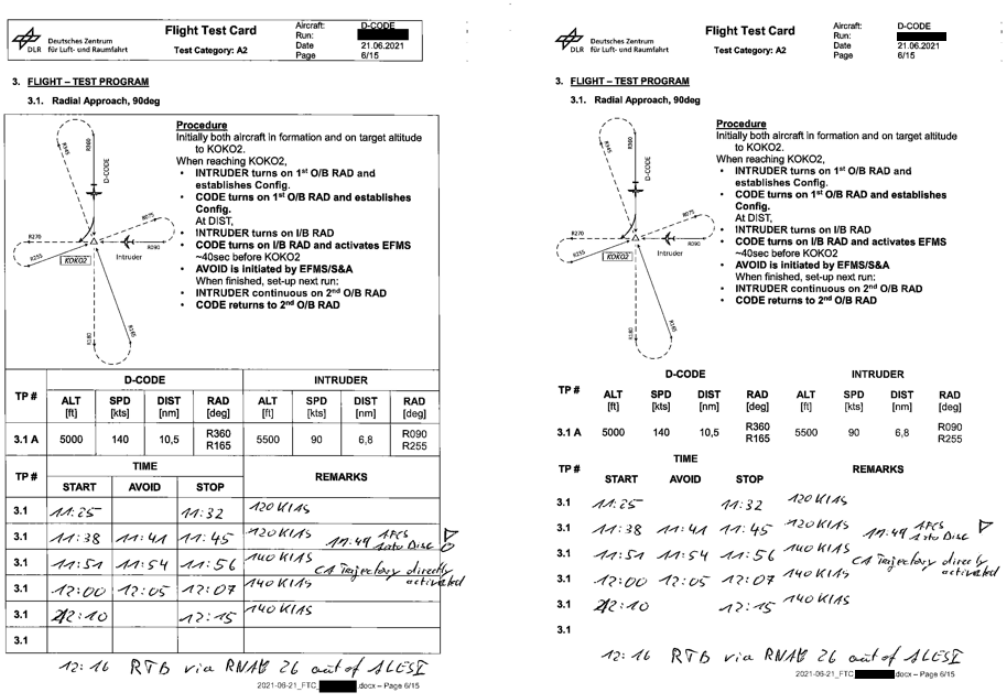


Figure 4.2: Example for one page of flight test cards as binarized input (left) and output (right) of the regional segmentation

```
<pc:SeparatorRegion id="region0027">
    <pc:Coords points="125,1527 1171,1527 1171,1539 125,1539"/>
</pc:SeparatorRegion>
<pc:SeparatorRegion id="region0028">
    <pc:Coords points="121,298 127,298 127,1596 121,1596"/>
</pc:SeparatorRegion>
<pc:SeparatorRegion id="region0030">
    <pc:Coords points="1165,301 1171,301 1171,1597 1165,1597"/>
</pc:SeparatorRegion>
<pc:SeparatorRegion id="region0031">
    <pc:Coords points="222,911 227,911 227,1596 222,1596"/>
</pc:SeparatorRegion>
<pc:SeparatorRegion id="region0032">
    <pc:Coords points="695,912 700,912 700,1596 695,1596"/>
</pc:SeparatorRegion>
<pc:SeparatorRegion id="region0033">
    <pc:Coords points="380,1169 383,1169 383,1596 380,1596"/>
</pc:SeparatorRegion>
<pc:SeparatorRegion id="region0034">
    <pc:Coords points="537,1169 541,1169 541,1596 537,1596"/>
</pc:SeparatorRegion>
<pc:SeparatorRegion id="region0035">
    <pc:Coords points="125,1590 1171,1590 1171,1601 125,1601"/>
</pc:SeparatorRegion>
```

Figure 4.3: Example labeling for region segmentation in page-xml format

```
<pc:TextRegion id="region0010" orientation="0.208141690663098" readingDirection="left-to-right" textLineOrder="top-to-bottom">
    <pc:Coords points="337,670 438,670 438,834 337,834"/>
    <pc:TextLine id="region0010_line0000">
        <pc:Coords points="338,694 339,694 340,702 339,702"/>
    </pc:TextLine>
    <pc:TextLine id="region0010_line0001">
        <pc:Coords points="339,709 340,709 340,721 339,721"/>
    </pc:TextLine>
    <pc:TextLine id="region0010_line0002">
        <pc:Coords points="339,728 340,728 340,740 339,740"/>
    </pc:TextLine>
    <pc:TextLine id="region0010_line0003">
        <pc:Coords points="339,747 340,747 340,759 339,759"/>
    </pc:TextLine>
    <pc:TextLine id="region0010_line0004">
        <pc:Coords points="339,766 340,766 340,778 339,778"/>
    </pc:TextLine>
    <pc:TextLine id="region0010_line0005">
        <pc:Coords points="338,672 420,672 421,834 339,834"/>
    </pc:TextLine>
    <pc:TextLine id="region0010_line0006">
        <pc:Coords points="339,803 431,803 431,816 339,816"/>
    </pc:TextLine>
    <pc:TextLine id="region0010_line0007">
        <pc:Coords points="339,815 438,815 438,834 339,834"/>
    </pc:TextLine>
</pc:TextRegion>
```

Figure 4.4: Example labeling for line segmentation in page-xml format

Recognition Process

For text recognition from the image, the tesserocr-recognize module is used. This module accepts two types of input: the image output from the region segmentation and the Page-XML file from the line segmentation. Additionally, the OCR-D recognition module requires a model trained to recognize characters in various languages and styles. For this case study, a model specifically trained on flight test cards is used to ensure accurate recognition of their content. More details about the model are provided in the next section. The output of the recognition process is a Page-XML file containing the extracted content identified by the recognition module.

```
<pc:TextRegion id="region0005" orientation="0.2081416906630980" readingDirection="left-to-right" textLineOrder="top-to-bottom">
    <pc:Coords points="142,211 495,211 495,279 142,279"/>
    <pc:TextLine id="region0005_line0000">
        <pc:Coords points="144,212 469,211 469,229 144,230"/>
        <pc:Word id="region0005_line0000_word0000">
            <pc:Coords points="145,213 160,213 160,228 145,228"/>
            <pc:TextEquiv conf="0.9378450775146480">
                <pc:Unicode>3.</pc:Unicode>
            </pc:TextEquiv>
        </pc:Word>
        <pc:Word id="region0005_line0000_word0001">
            <pc:Coords points="180,213 262,213 262,229 180,229"/>
            <pc:TextEquiv conf="0.9266905212402340">
                <pc:Unicode>FLIGHT</pc:Unicode>
            </pc:TextEquiv>
        </pc:Word>
        <pc:Word id="region0005_line0000_word0002">
            <pc:Coords points="268,221 281,220 281,223 268,224"/>
            <pc:TextEquiv conf="0.1996247863769530">
                <pc:Unicode>--</pc:Unicode>
            </pc:TextEquiv>
        </pc:Word>
        <pc:Word id="region0005_line0000_word0003">
            <pc:Coords points="287,212 345,212 345,228 287,228"/>
            <pc:TextEquiv conf="0.9494461059570310">
                <pc:Unicode>TEST</pc:Unicode>
            </pc:TextEquiv>
        </pc:Word>
        <pc:Word id="region0005_line0000_word0004">
            <pc:Coords points="352,212 469,212 469,229 352,229"/>
            <pc:TextEquiv conf="0.8363275146484370">
                <pc:Unicode>PROGRAMM</pc:Unicode>
            </pc:TextEquiv>
        </pc:Word>
        <pc:TextEquiv conf="0.7699868011474610">
            <pc:Unicode>3. FLIGHT -- TEST PROGRAMM</pc:Unicode>
        </pc:TextEquiv>
    </pc:TextLine>
    <pc:TextLine id="region0005_line0001">
        <pc:Coords points="179,258 179,279 495,279 495,257"/>
        <pc:Word id="region0005_line0001_word0000">
            <pc:Coords points="180,259 216,259 216,275 180,275"/>
            <pc:TextEquiv conf="0.9392526245117190">
                <pc:Unicode>3.1.</pc:Unicode>
            </pc:TextEquiv>
        </pc:Word>
```

Figure 4.5: Example output from recognition process in page-xml format

Recognition Model training

To train the model, which is a lstm, a dataset of handwritten text was created by extracting individual inputs from the flight test cards as PNG images and providing their corresponding text content in TXT files as ground truth. For this thesis, 10 different flight test cards were used, containing 322 unique handwritten content pieces. The training process begins with the LSTM training script provided by OCR-D Tesseract. This script generates box files, which describe the individual characters in the image files using the ground truth provided in the TXT files. After the box files are created, they, along with the corresponding images, are used for training. The training process itself is handled by the ocrd-tesseract module. Using the hyperparameters provided in their GitHub repository is justified by the assumption that they conducted an analysis to determine the optimal hyperparameters for their model. The evaluation of the model's performance will be discussed in the system evaluation section.

4.1.2 Implementation of Table Detection for data parsing

This section describes the dataset creation, training process, and implementation of the table detection module. It provides details on how the dataset is generated, the training process is conducted, and how the module is refined and optimized for the case study presented in this thesis.

Dataset Creation

To create the dataset, an algorithm was developed to eliminate human bias when selecting the training, test, and validation data. This algorithm takes the paths of all flight test cards in the case study as input and converts each page of the PDF files into a JPG image. The dataset consists of a total of 117 pages from 17 different flight test cards. These were selected manually based on their content, ensuring that most of the flight test cards include all possible table types to achieve better results during training. The flight test cards are then automatically sorted into test, training, or validation datasets. The algorithm also records the name of each flight test card and its dataset type in a JSON file, providing an overview of the dataset's structure. In the same step, XML files are generated to describe the bounding boxes of the tables. These files include the essential structure, such as the required XML format and the name of the corresponding file. This automation reduces the amount of manual input needed when filling out the XML files. After running this algorithm, a manual process is used to extract the bounding boxes from the images designated for training, testing, and validation. This was done by the author of this thesis using a self-written algorithm that allows drawing a bounding box on the image and labeling the type of table element. These bounding boxes are then automatically written into the corresponding XML files based on the drawn bounding boxes. Using the XML and JPG files, the required file lists for training are automatically generated. The algorithm reads the filenames and their paths, then stores them in the correct file list, including both the path and the filename. One of the key challenges during the labeling process involves questions such as whether an image belongs to the table or if the table should be split into multiple smaller tables. Answers to these questions are determined during the testing process.

Model training table detection

The model was trained using the training algorithm provided by Smock [11]. For the training, the hyperparameters provided in the GitHub repository for table detection are used. As described in the related work section, the model is a Faster R-CNN. The parameters used for training are summarized in Figure 4.6, and the model's performance was evaluated based on Average Precision and Average Recall. Due to the limited amount of training data available for this thesis, transfer learning was employed. This approach utilized Smock's pretrained model, which was fine-tuned by continuing the training process with data derived from the flight test cards. The input data, consisting of flight test cards, was divided into training, testing, and validation datasets. Specifically, the training dataset included 67 JPG files containing one or more tables from two different projects. The test dataset comprised 29 JPG files, while the validation dataset included 21 JPG files. During the systematic evaluation, models trained over different epochs were compared to identify the best-performing version. Additionally, the fine-tuned model was compared with two alternatives: (1) the model trained exclusively on the flight test card data (without transfer

learning) and (2) the pretrained model provided by Smock [11].

Model training table structure recognition

The structure model, which is a Faster R-CNN model, was trained using the training algorithm provided by Smock et al. [11]. Because the focus is on the pretrained model and the transfer learning model, the hyperparameters provided in the GitHub repository are used. Based on the analysis which were make by Smock et al.[11] they fit the best. To obtain a comprehensive comparison of the different models, transfer learning was also applied to train the structure recognition model. The provided training configuration from Smock was used for transfer learning, as these parameters are specifically tailored to the pretrained model. During training, Smock's algorithm evaluates several metrics, such as class error, loss, loss rate, and more (as shown in Figure 4.6). The model is trained until either the Average Precision and Average Recall start to decline or overfitting occurs. This process involved fine-tuning the pretrained model using 39 labeled JPG files for training, 25 labeled JPG files for testing, and 20 labeled JPG files for validation. These JPG files contained only tables, with headers, cells, and rows annotated in the dataset. The results of this transfer learning approach were systematically evaluated and compared against both the pretrained model provided by Smock [11] and a model trained without transfer learning.

```
Epoch: [0]  [ 0/33]  eta: 0:02:04  lr: 0.000050  class_error: 33.33  loss: 7.6848 (7.6848)  loss_ce: 1.1999 (1.1999)  lo
ss_bbox: 4.4227 (4.4227)  loss_giou: 2.0623 (2.0623)  loss_ce_unscaled: 1.1999 (1.1999)  class_error_unscaled: 33.3333 (
33.3333)  loss_bbox_unscaled: 0.8845 (0.8845)  loss_giou_unscaled: 1.0311 (1.0311)  cardinality_error_unscaled: 12.5000
(12.5000)  time: 3.7583  data: 0.4423
```

Figure 4.6: Example for training parameter

Model implementation

For the implementation of the table detection and table structure recognition models in this thesis, the provided Python interface for evaluation is utilized. A configuration file is included to set the necessary arguments that should not be modified by the user, such as the evaluation mode (eval), the configuration file paths for table detection and structure recognition, and the model's path. The parameters provided by the pipeline depend on the active evaluation step and include the data type (e.g., detection or structure) and the path to the image files to be evaluated in the pipeline. From the model's evaluation code, the bounding boxes generated in the results are extracted for subsequent steps, along with the table types identified by the model.

4.2 Implementation of the Database and Pipeline

This section discusses the implementation of the data model as well as the overall pipeline. It provides details on how the data is stored and how it is ingested into the database.

4.2.1 Implementation of the selected data model

For the implementation of the data model, MySQL is used. The reason for using MySQL is described in Chapter 3.1.2. This chapter focuses on the implementation of the data model.

Flight test table implementation

As described above, the data model contains a table named flighttest, which has a primary key named name stored as VARCHAR(255). This choice is based on the variable length of flight names in the DLR. The flight names in historical records have never exceeded 255 characters. Based on this data, using a VARCHAR(255) data type is sufficient to accommodate the maximum required space for storing flight names.

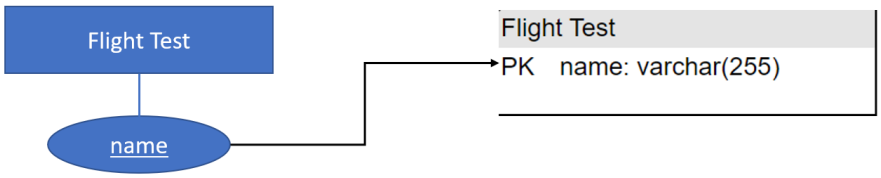


Figure 4.7: Enity-Realtionship-Diagram Flight Test Entity to Table in Database

Maneuver table implementation

The next table stored in the MySQL Relational Database Management System (RDBMS) is the maneuver table. This table includes the maneuverid, which is an integer serving as the primary key, and the name of the maneuver, stored as a VARCHAR with a length of 125. This length is chosen to accommodate the varying lengths of maneuver names while ensuring enough space. An optional image, if available, is stored as a binary large object (BLOB), and the procedure is stored as LONGTEXT to handle detailed and lengthy maneuver descriptions, particularly for complex maneuvers. In addition, the table stores key parameters used for each maneuver in the case study, including the altitude (alt) of the main aircraft, the speed (spd), and the indicated airspeed (ias). These values are stored as integers. For each maneuver, either the spd or the ias value will be provided, depending on the version of the flight test card. Lastly, the table includes the flightname as a foreign key, enabling a one-to-many relationship between the flight test table and the maneuver table.

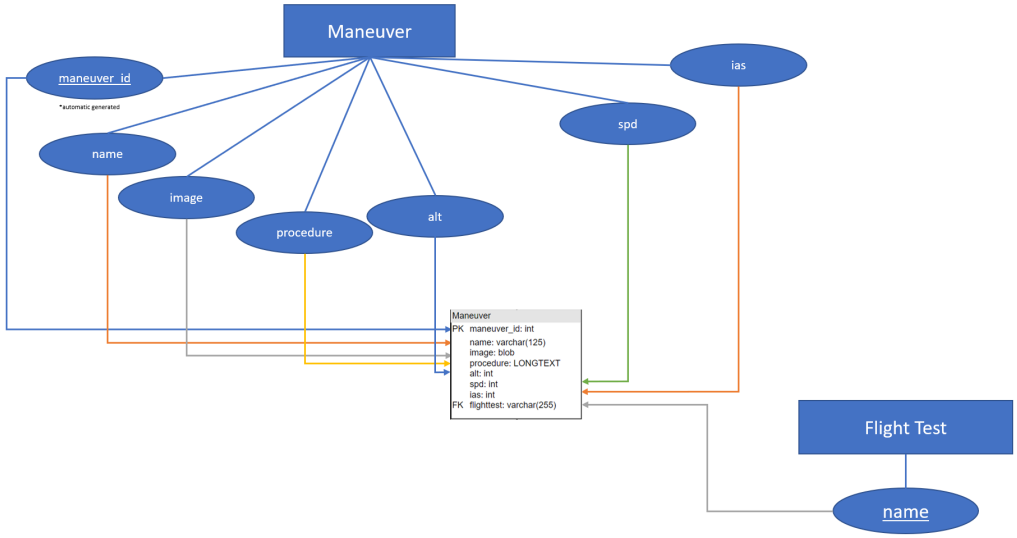


Figure 4.8: Enity-Realtionship-Diagram altimeter check maneuver Entity to Table in Database

Radar maneuver table implementation

Next is the radar maneuver table, which extends the basic maneuver table and stores values specific to radar maneuvers. For storage, a table for the class will be created containing the values from the maneuver table along with the additional values specific to the radar maneuvers. This approach is justified because it allows new subclasses of maneuvers to be easily appended in the future. Furthermore, this structure avoids the need for joins during read and write operations, simplifying requests and improving efficiency. Non-polymorphic requests can also be handled more straightforwardly with this design. These values include the track (trk) and the heading (hdg) of the main aircraft. Both values are stored as integers. This design helps minimize NULL values by ensuring that radar-specific data is stored separately, rather than including these fields in the basic maneuver table, where they would not apply to all maneuvers. For integration into the data model, a dedicated table is created for each maneuver type by appending the necessary elements to those defined in the basic maneuver table. However, in the case of radar maneuvers, all of these variables are stored within the radar maneuver table itself.

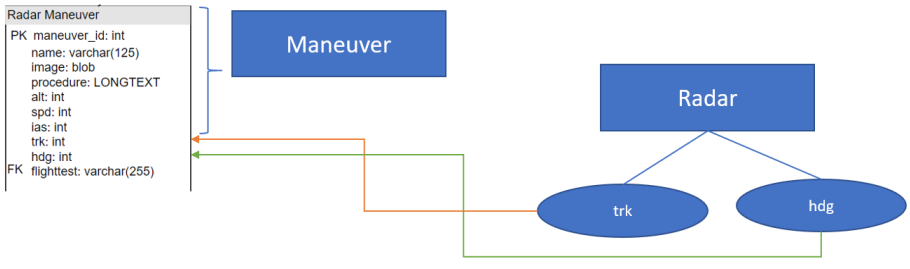


Figure 4.9: Enity-Realtionship-Diagram Radar maneuver Entity to Table in Database

Altimeter check maneuver table implementation

Another table extending from the basic maneuver table is the altimeter check maneuver table, which covers the minimum set of parameters required for maneuvers involving an intruding aircraft. Similarly to the other maneuvers, a new table will be created for this maneuver, containing both the additional information specific to it and the inherited parent information. This ensures that the maneuver-specific attributes are stored appropriately while maintaining a clear relationship with the parent maneuver data. This table stores the intruder_alt, intruder_spd, and intruder_ias as integers. Similar to the basic maneuver table, only one of the values, either intruder_spd or intruder_ias, is set, while the other remains NULL, depending on the version of the flight test cards.

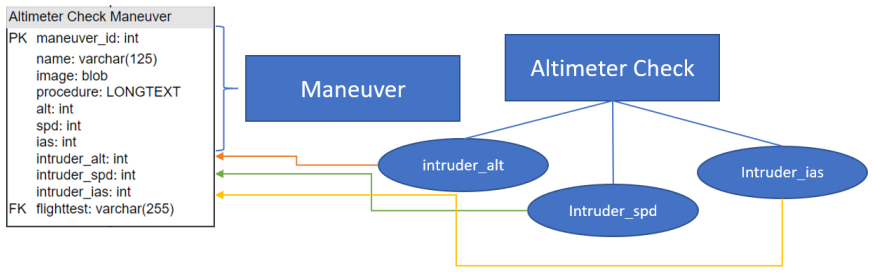


Figure 4.10: Enity-Realtionship-Diagram altimeter check maneuver Entity to Table in Database

Radial approach maneuver table implementation

Next a table is created to cover values required for the radial approach maneuvers. This table extends the intruder maneuver table and stores the following values: intruder_dist, dist, and rad of the main aircraft. For this maneuver, an additional table will also be created and merged with the parent information. This ensures that the specific attributes of the maneuver are stored separately while maintaining a clear relationship with the parent data, supporting both extensibility and efficient data management. The distance values are stored as floats, while the rad is stored as a VARCHAR(4) because it follows a specific structure: the letter 'R' followed by up to three integers (e.g., 'R090'). As well as the intruder_rad. Like the rad of the main aircraft, the data for this parameter is stored as a VARCHAR(4) because the format for the intruding aircraft is the same as that of the main aircraft. The radial approach stores also the data which is contained in the altimeter check like intruder_alt, intruder_spd and intruder_ias as integer.

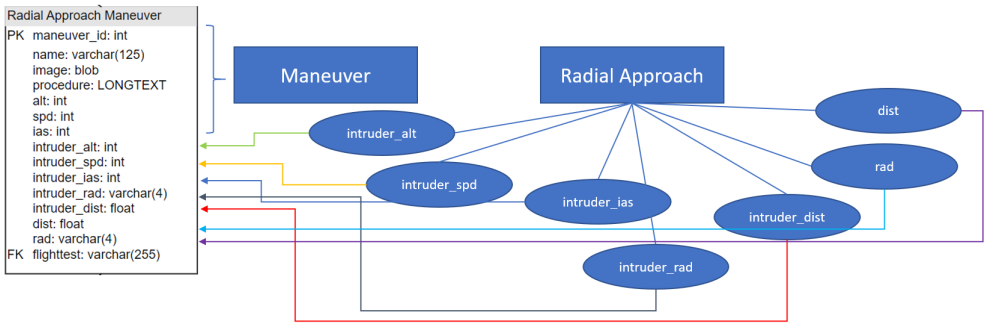


Figure 4.11: Enity-Realtionship-Diagram radial approach maneuver Entity to Table in Database

Maneuvering approach maneuver table implementation

Lastly, in our case study, the maneuvering approach has parameters that are specific to this type of maneuver. As with the radial approach, an extra table will be created for the maneuvering approach and merged with the parent data. The advantages of this approach are the same as those mentioned in the radar maneuver section, including improved extensibility for adding new subclasses, simplified read and write access without requiring joins, and easier handling of non-polymorphic requests. This means that the table extends from the maneuver table and adds values such as the ground speed (g_spd), intruder ground speed (intruder_g_spd), and the track of the intruding aircraft (intruder_trk). These values are unique to this maneuver and are stored as integers. As well as the values which are also stored in the radial approach maneuver like intruder_dist, dist, and rad of the main aircraft and intruder_dist, dist, and rad of the main aircraft. Same like the radial approach it also store the data which is contained in the altimeter check like intruder_alt, intruder_spd and intruder_ias as integer.

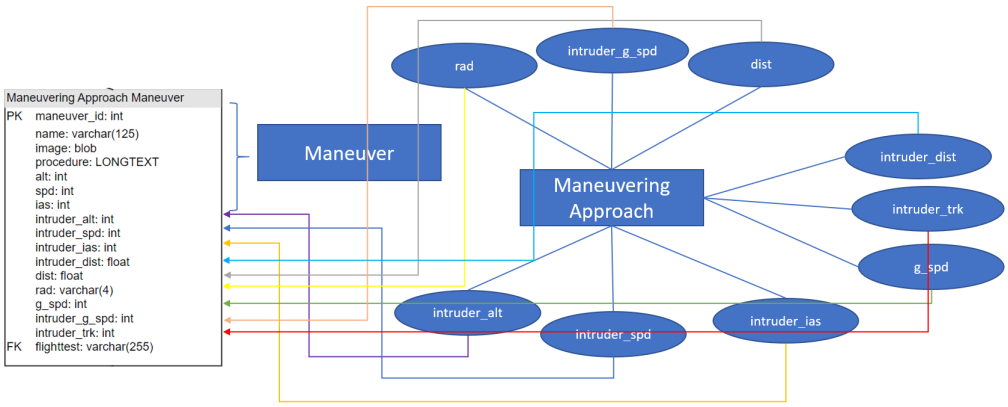


Figure 4.12: Enity-Realtionship-Diagram maneuvering approach maneuver Entity to Table in Database

Testpoint table implementation

The maneuvers are followed by the test point table, which is associated with the maneuvers. The test point table includes several attributes. The testpointid serves as a unique identifier, while the name is stored as a VARCHAR(7), with its length based on the typical structure of a test point name, such as "number-dot-number-whitespace-letter" (e.g., "1.2 A"). Although the average length of such names is five characters, a small buffer is added to set the length to seven. The comment field is stored as a VARCHAR(255), allowing sufficient space for comments of varying lengths. The table also stores the start_time, end_time, and avoid_time for each test point. In the worst case, only one of these values will be populated, while the others will remain NULL. These time values are stored in the DATETIME format. Finally, the maneuverid is included as a foreign key to establish a one-to-many relationship between the maneuver table and the test point table 4.13.

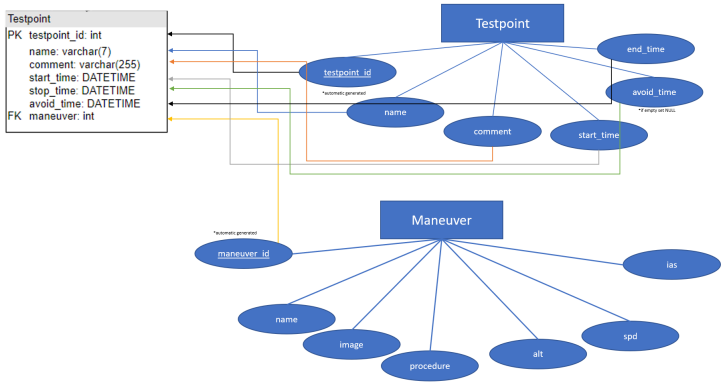


Figure 4.13: Enity-Realtionship-Diagram testpoint Entity to Table in Database

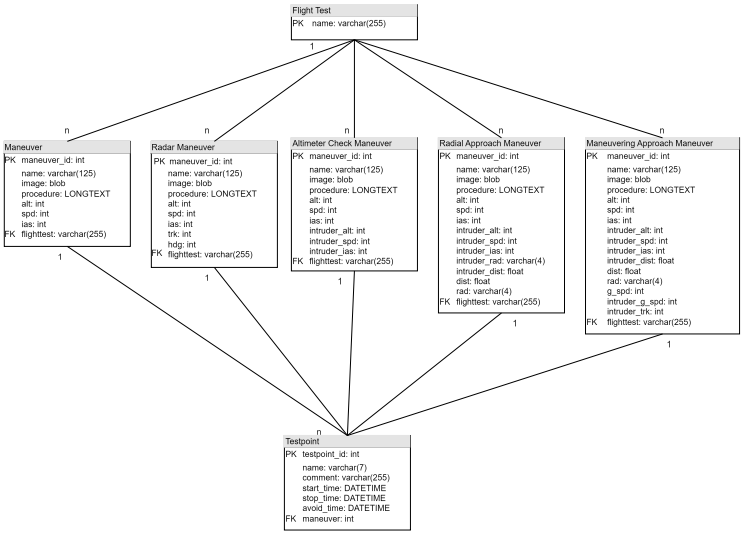


Figure 4.14: Data model with datatype for Flight Test Cards

Advantages of the relational model

This section highlights the advantages of implementing the relational model compared to the current storage method, which utilizes JSON files within TwinStash. To demonstrate these advantages, a subset of four essential queries, required for analyzing flight test data at the DLR, is presented. Each query is shown alongside its results and compared to the feasibility of implementing the same queries using JSON files without a fixed structure. The comparison emphasizes the efficiency, structure, and ease of querying provided by the relational model, particularly in contrast to the challenges posed by an unstructured JSON-based approach. Possible queries that highlight the advantages of the relational model include retrieving all flights that contain a specific maneuver, identifying all maneuvers flown at a specific speed, listing all maneuvers performed during a particular flight, and extracting all test points of a specific maneuver within a single flight test. The last query is particularly relevant because the same maneuver might be executed multiple times during a flight test under different conditions, such as at the beginning with a full fuel load and later with reduced fuel. A query for flights containing a specific maneuver joins the FlightTest table with the RadialApproachManeuver table and retrieves all flight names where the RadialApproachManeuver.name matches "Radial Approach, 90deg." The query filters for only those flights where this maneuver is included, returning four flights 4.15 out of five in the database.

```
SELECT FlightTest.name
FROM FlightTest
INNER JOIN RadialApproachManeuver
    ON FlightTest.name = RadialApproachManeuver.flighttest
WHERE RadialApproachManeuver.name = "Radial Approach, 90deg"
```

Code 1: Example SQL-Query for all flight tests containing a Radial Approach, 90deg

#	name
1	2021-06-21_FTC_[redacted]_#1
2	F24-11_2024-09-10_FTC_[redacted]#1
3	F24-12_2024-09-16_FTC_[redacted]#6
4	F24-12_2024-09-16_FTC_[redacted]#6

Figure 4.15: Result of all flight with a Radial Approach , 90deg in the Database

A query for maneuvers performed at a specific speed combines the results from multiple maneuver tables, such as ManeuveringApproachManeuver, RadialApproachManeuver, RadarManeuver, AltimeterCheckManeuver, and Maneuver, where the indicated airspeed (ias) is 120 knots. It aggregates results from all relevant tables using 'UNION' to provide a comprehensive list of maneuvers at 120 kts.

```
SELECT name FROM ManeuveringApproachManeuver WHERE ias = 120
UNION
SELECT name FROM RadialApproachManeuver WHERE ias = 120
UNION
SELECT name FROM RadarManeuver WHERE ias = 120
UNION
SELECT name FROM AltimeterCheckManeuver WHERE ias = 120
UNION
SELECT name FROM Maneuver WHERE ias = 120
```

Code 2: Example SQL-Query for all maneuvers at 120kts

#	name
1	Radial Approach, 90deg
2	Radial Approach, FAST BEAM
3	Radial Approach, HEAD ON

Figure 4.16: Result of the maneuvers at 120kts in the Database

A query for all maneuvers in a specific flight searches for all maneuvers within a particular flight, identified by the flight name, by modifying the ias field to flighttest. It collects data from different maneuver tables and returns a list of all maneuvers flown in that specific flight 4.17.

```
SELECT name FROM ManeuveringApproachManeuver
    WHERE flighttest = "F24-12_2024-09-16_FTC_PROJECT#6"
UNION
SELECT name FROM RadialApproachManeuver
    WHERE flighttest = "F24-12_2024-09-16_FTC_PROJECT#6"
UNION
SELECT name FROM RadarManeuver
    WHERE flighttest = "F24-12_2024-09-16_FTC_PROJECT#6"
UNION
SELECT name FROM AltimeterCheckManeuver
    WHERE flighttest = "F24-12_2024-09-16_FTC_PROJECT#6"
UNION
SELECT name FROM Maneuver
    WHERE flighttest = "F24-12_2024-09-16_FTC_PROJECT#6"
```

Code 3: Example SQL-Query for all maneuvers from a specific flight

#	name
1	Radial Approach, 90deg
2	Radial Approach, FAST BEAM
3	Radial Approach, HEAD ON
4	Altimeter Check

Figure 4.17: Result of the maneuvers of a specific flight query in the Database

A query for test points of radial maneuvers in a specific flight retrieves test points for a specific flight that contains radial approach maneuvers. The data extracted includes Testpoint.name, Testpoint.start_time, Testpoint.stop_time, Testpoint.avoid_time, and the corresponding RadialApproachManeuver.name. The INNER JOIN ensures that only test points linked to radial approach maneuvers are retrieved 4.18.

```
SELECT Testpoint.name, Testpoint.start_time,
    Testpoint.stop_time, Testpoint.avoid_time, RadialApproachManeuver.name
FROM Testpoint
INNER JOIN RadialApproachManeuver
    ON Testpoint.maneuver_id_radial = RadialApproachManeuver.maneuver_id
WHERE RadialApproachManeuver.flighttest = "F24-12_2024-09-16_FTC_PROJECT#6"
```

Code 4: Example SQL-Query for all radial approach maneuver testpoints from a specific flight

#	name	start_time	stop_time	avoid_time	name
1	3.2A	2024-09-16 12:47:00	2024-09-16 12:50:00	(NULL)	Radial Approach, 90deg
2	3.2A	2024-09-16 13:09:00	2024-09-16 13:13:00	2024-09-16 13:12:00	Radial Approach, 90deg
3	3.2A	2024-09-16 13:30:00	2024-09-16 13:34:00	2024-09-16 13:33:00	Radial Approach, 90deg
4	3.2B	2024-09-16 12:57:00	2024-09-16 13:03:00	2024-09-16 13:02:00	Radial Approach, 90deg
5	3.2B	2024-09-16 13:17:00	(NULL)	(NULL)	Radial Approach, 90deg
6	3.2B	2024-09-16 13:39:00	(NULL)	2024-09-16 13:44:00	Radial Approach, 90deg
7	3.3A	2024-09-16 13:50:00	2024-09-16 13:54:00	2024-09-16 13:53:00	Radial Approach, FAST BEAM
8	3.3A	2024-09-16 14:11:00	2024-09-16 14:15:00	2024-09-16 14:14:00	Radial Approach, FAST BEAM
9	3.3A	2024-09-16 14:30:00	2024-09-16 14:35:00	2024-09-16 14:34:00	Radial Approach, FAST BEAM
10	3.3B	2024-09-16 13:58:00	2024-09-16 14:05:00	2024-09-16 14:04:00	Radial Approach, FAST BEAM
11	3.3B	2024-09-16 14:19:00	2024-09-16 14:25:00	2024-09-16 14:24:00	Radial Approach, FAST BEAM
12	3.3B	2024-09-16 14:40:00	2024-09-16 14:45:00	2024-09-16 14:45:00	Radial Approach, FAST BEAM
13	3.4A	2024-09-16 14:51:00	2024-09-16 14:55:00	2025-01-31 15:54:00	Radial Approach, HEAD ON
14	3.4A	2024-09-16 15:11:00	2024-09-16 15:15:00	2024-09-16 15:14:00	Radial Approach, HEAD ON
15	3.4B	2024-09-16 14:59:00	2024-09-16 15:05:00	2024-09-16 15:05:00	Radial Approach, HEAD ON

Figure 4.18: Result of the testpoint query in the Database

These queries demonstrate the power of relational databases in enabling detailed, flexible, and efficient data retrieval for analysis in the context of flight test data. They allow for complex data relationships, such as flight maneuvers, flight data, and test points, to be efficiently queried and analyzed. This provides significant advantages over more static and unstructured data storage methods, such as JSON files which require looping through

the data to extract relevant information. Additionally, the structure of JSON files must be consistent; if different files have varying structures, users may struggle to retrieve the required data efficiently. In such cases, users must manually identify and merge different keys, making the process more error-prone and time-consuming. This lack of standardization complicates querying and integration, whereas a relational database ensures structured storage and efficient retrieval through predefined schemas and SQL queries.

4.2.2 Pipeline development and execution

This section discusses the modules of the pipeline and provides details about their implementation. The focus is on explaining additional steps not mentioned in the previous sections and outlining how the data from individual modules is interconnected to function as a unified pipeline.

Image converter

The image converter is implemented in Python and performs two primary tasks: extracting pages from a PDF and converting them into JPEG files, as well as naming the created images based on a flight name derived from the PDF filename. For the conversion process, the pypdfium2 library is utilized, which allows the extraction of PDF pages and their conversion to JPEG format. To name the generated images, a custom function extracts the flight name from the PDF filename. The function begins by splitting the filename using underscores ("_") to separate its components. It then searches for specific key parts within these components: the date is identified by splitting a segment with hyphens ("-"), while the flight number is found by locating a word containing the "#" symbol. According to the naming convention, the campaign name is the word that precedes the flight number. The campaign name is merged with the flight number to form the flight name. The flight name, combined with the extracted date, is used as the base for naming the images. The final image filenames include the date, the flight name, the keyword "page," and the page number, following the structure <date>_<flightname>_page_<page_number>.jpg. For example, if the PDF filename is campaignX_2023-12-15_flight#123.pdf, the extracted flight name would be campaignX123, and the resulting image filenames would be 2023-12-15 _campaignX123_page_1.jpg, 2023-12-15_campaignX123_page_2.jpg, and so on. This implementation ensures efficient extraction and conversion of PDF pages while producing meaningful filenames that make it easy to organize and retrieve individual pages.

Recognized elements to image

The module for converting recognized elements into images takes the output of the table transformer, including the bounding boxes and the labels of the elements within those bounding boxes. It then converts each element into a separate image. This module is implemented using OpenCV, a widely used Python library for image processing. As input,

the module requires the xmin, xmax, ymin, and ymax coordinates of the bounding box, the image path, and the label of the element. The algorithm processes the image by reading it with OpenCV's image reading command and cropping it using array slicing from ymin to ymax and xmin to xmax. Finally, the cropped image is saved with a unique filename that includes the original image name, the element label, and an index to ensure uniqueness.

Merge relational information and recognized content

To merge the relational information with the recognized content, the outputs of the table detection module and OCR-D must be combined. As outlined in the conception section of the pipeline, the OCR-D process runs after extracting the PNGs from the recognized elements and generating the relational information. The results produced by OCR-D are then utilized in this module. The required information from the table detection module includes the flight test card name, the page names of the flight test cards, the names of the subelements (e.g., tables, table column headers, columns, and rows), and their respective bounding boxes. From OCR-D, the necessary information consists of the recognized text lines extracted from the sliced images provided as input. Since the image names correspond to the subelements of the flight test cards (e.g., columns and rows), this allows for a direct merging process. The merging process results in a single dictionary that contains the flight test card name, along with all subelements and their bounding boxes. Each subelement is appended with the recognized text from OCR-D. This ensures that every subelement has both its positional information and its associated text stored, enabling contextual understanding of these elements for subsequent storage in the database.

Storage of flight test card content

To store the flight test card data in the provided database structure, the recognized elements are converted to fit the defined data structure. The input for this module is the output dictionary from the merging module. This module processes the dictionary by extracting the context, such as identifying which table headers correspond to columns and which keys correspond to values within the table. This is determined based on the positions and types of the elements. Once the context is extracted from the dictionary, the data is stored in the database. First, the flight is created to append all maneuvers associated with it. The name of the flight test card is used as the flight name in the database. The module interacts with the MySQL database management system using the MySQL library, which provides functionality to manage and insert data into the database. The MySQL Python library is used because the pipeline is implemented in Python. This ensures that all operations are executed within the Python environment, eliminating the need for external code or scripts to be called separately from the pipeline.

Connection to flight test data

To ensure the usability of the data from the flight test card, it is crucial to connect this data with the flight test data stored in the twinstash of the DLR. The information from the flight test cards is essential for a deeper understanding of the flight test data. Without this connection between the flight test data and the flight test card data, the advantages of the flight test card database are significantly diminished when compared to the currently used solution. To connect the flight test card data with the flight test data stored in twinstash, the user tags in twinstash are utilized. A key named has_ftc_data with the value true is added to the dictionary, along with a key named ftc_link containing the database link needed to query information from the flight test card database. The twinstash API is used to manipulate the user tags. Since there is no direct update command for user tags, the existing user tags of the flight, identified by the flight test name, are first downloaded. The new information, as mentioned above, is then appended to the user tags.

5 Systematic Evaluation

This chapter focuses on the testing of the pipeline elements. It outlines the testing procedure, presents the results, and provides a discussion of the findings. The goal is to evaluate and understand how well the pipeline performs.

5.1 Testing Procedure

This section outlines the testing procedure for the OCR-D module and the model specifically trained for this use case. Additionally, the testing procedure for the Table Transformer module and its trained model is described. Finally, the testing process for the pipeline as a complete system is detailed.

5.1.1 OCR-D testing

For testing the OCR-D module, this thesis employs the Character Error Rate (CER) metric, as described in the paper "A Survey of OCR Evaluation Tools and Metrics" by Neudecker et al. [8] The CER is defined as follows: "CER is the inverted accuracy, and defined as CER = (i + s + d) / n, where n is the total number of characters, i is the minimal number of character insertions, s is the number of substitutions, and d is the number of deletions required to transform the reference text into the OCR output." (Neudecker et al., p. 14, [8]). The evaluation requires both the OCR output and the ground truth strings from the evaluation dataset. The ground truth strings are manually created and provided as text files for each image to measure the Character Error Rate (CER). By calculating the CER, the accuracy of the OCR-D module can be quantitatively assessed. Based on these results, the most suitable model is selected for implementation. For the analysis of the Character Error Rate (CER), a short algorithm is implemented using the TorchMetrics library, which provides a built-in implementation for calculating CER. The algorithm takes the hand-labeled ground truth of the images as input, along with the output generated by the OCR-D system. These inputs are then parsed and compared using the CER implementation from the TorchMetrics library.

5.1.2 Table Transformer testing

For the testing procedure of the Table Transformer, an algorithm was developed to evaluate its performance. The algorithm uses a small, manually labeled dataset of flight test cards from the case study, ensuring that the testing dataset includes at least one instance of each table type present in the case study. The evaluation process involves comparing

the bounding boxes predicted by the model to the manually labeled bounding boxes. The offset for each side of the bounding box is calculated as a percentage difference using the following formula:

$$\text{offset} = \text{abs}(\text{xmin\_ref}/\text{xmin\_res} - 1)$$

This formula measures the deviation of the model's result from the reference position. For example, if the offset value is 0.5, it indicates that the predicted position is shifted by 50% from the reference value. The mean offsets for all sides are then aggregated and averaged over the entire dataset to compute the model's overall mean error for each side. This is calculated by:

$$\left(\sum_{n=1}^{\text{length of dataset}} \text{offset\_n}\right)/\text{length of dataset}$$

It is hypothesized that the model with the lowest mean error on the evaluation dataset should be selected, as it provides the most accurate and reliable results. In addition to the custom testing procedure developed in this thesis, the metrics utilized in the paper by Smock et al. [11] are also applied. The results obtained using these metrics are compared to the corresponding values provided in Smock's work to ensure consistency and evaluate the relative performance of the models.

5.1.3 Pipeline testing

The pipeline testing procedure focuses on modules that were not explicitly tested in isolation. That means it tests the merging of the relational information with the recognized content. To assess this process, a manually merged evaluation dataset is created, and the output of the merging algorithm is compared against it. The results of the merging algorithm are categorized as either correct or incorrect, meaning that the evaluation measures whether the content is merged into the correct section. No intermediate or "partially correct" solutions are considered valid, as any deviation from the expected result renders the data unusable. This strict criterion is necessitated by the time-sensitive nature of the flight test card content, where even minor errors—such as an incorrect timestamp—invalidate the analysis. The evaluation focuses solely on the positioning of the merged content, not the correctness of the content itself as derived from OCR-D. In other words, the evaluation determines whether the content is placed in the correct location after merging, without verifying the accuracy of the content. The efficiency of the merging algorithm is calculated by assigning a value of 1 for each correct result and dividing the sum of correct results by the total number of evaluated values. The correctness is expressed as a formula:

$$\text{Correctness} = \text{Number of Correct Results}/\text{Total Number of Evaluated Results}$$

This provides a quantitative measure of the algorithm's accuracy.

5.2 Testing Results

This section presents the testing results corresponding to the testing procedures described above. The performance of the various modules and models is compared, and based on the results, the final selection of models for the modules is determined.

5.2.1 OCR-D testing results

In the OCR-D testing results, the English model, which was trained on machine-written data, is compared with the model trained on handwritten data from the flight test engineer at DLR. The results show that there is insufficient training data for the handwritten model, especially when compared to the larger dataset available for the standard English model. The English model from OCR-D achieves a Character Error Rate (CER) of 0.2646 on the testing dataset. This dataset includes a variation of handwritten text, as well as short and long machine-written text. These results were anticipated, and methods to improve these values should be addressed in the outlook section. During the intersection of the results from the different inputs, it was observed that longer machine-written texts are recognized more accurately than shorter machine-written texts. The greatest difficulty, however, lies in the recognition of handwritten data. Comparing this to the model trained on the handwritten data of the flight test engineer, it was observed that the handwritten data model delivers a significantly higher CER of 0.9654, making it unusable. For machine-written data, further analysis will be conducted in the ablation study to identify potential improvements. The results of the English model and the model trained on the handwritten data are shown in table .

Model	CER
English Model	0.2646
Handwritten Model	0.9654

Table 5.1: Testing results of OCR models

5.2.2 Table Transformer testing results

This section is divided into two parts: the testing results for table detection and table structure recognition. This distinction is essential as each of these modules utilizes a uniquely trained model, and their performance needs to be evaluated separately.

Table detection testing results

First, this thesis compares the performance of the model trained using transfer learning with the baseline model provided by Smock et al. [11]. The initial comparison is conducted using the metrics generated during the training process. The results, summarized in a table 5.2, indicate that the transfer learning model achieves superior performance on

the testing dataset compared to the standard model. Another approach involved training exclusively on the flight test cards from the DLR. However, the results of this training were quite poor, which led to the decision not to investigate this approach further in this thesis. For further evaluation, the focus shifts to the FTC-DLR (transfer learing) model, as it delivers better overall results. However, an additional analysis is performed to quantify the residual offset in the model's predictions, offering insights into areas where improvement is still needed. The results can be observed in the image 5.1, where the FTC-DLR model successfully identifies all three tables, whereas the DETR model detects only one table with inaccurately placed bounding boxes. In the table, AP (Average Precision) measures the overall precision of recognition on the images. AP50 and AP75 represent the average precision at specific thresholds, meaning that a result is considered recognized if it matches at least 50% or 75% of the reference data, respectively. AR (Average Recall) measures the recall values across different thresholds, such as 50%, 75%, and 100%, providing an overall assessment of how well the system identifies relevant elements.

Model	AP50	AP75	AP	AR
DETR R18	0.346	0.023	0.168	0.293
FTC-DLR	0.041	0.020	0.019	0.317
FTC-DLR (transfer learning)	0.932	0.932	0.807	0.912

Table 5.2: Testing results of DETR R18 and FTC-DLR on table detection, AP - Average Precision, AR - Average Recall

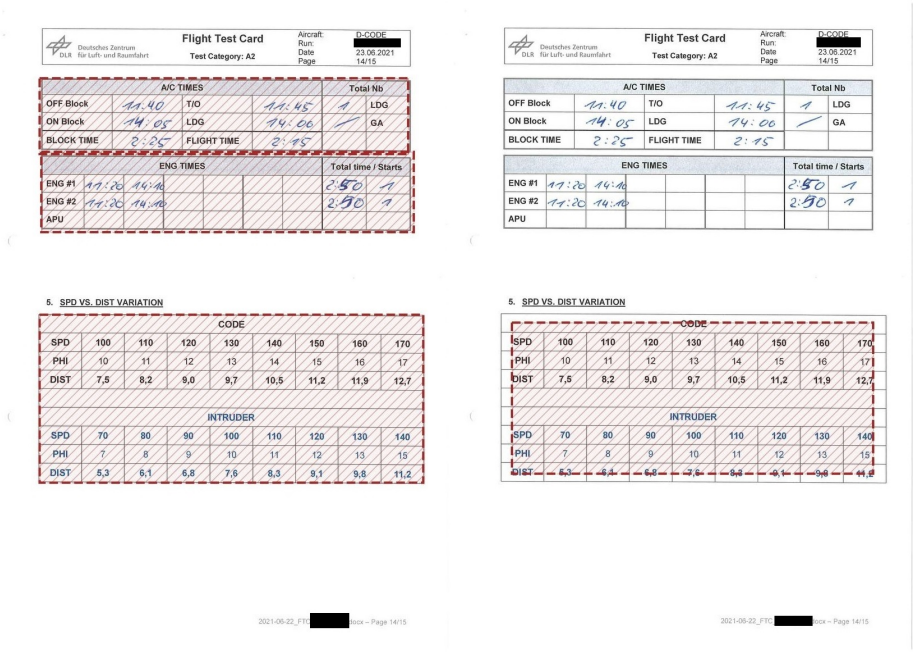


Figure 5.1: Example for bounding boxes found by FTC-DLR model and DETR

The analysis of the bounding box offsets reveals that the xmax value is generally larger than expected when compared to the hand-labeled bounding boxes. However, in the context of flight test cards, this discrepancy can offer some advantages. Specifically, the flight test engineer writes in 85% of the flight test cards in the case study above the borders of the table, and the larger bounding box could help accommodate this additional text, thereby improving the model's robustness in capturing the relevant information. The detailed results could be found in the table 5.3 for each x-value and y-value.

x min	x max	y min	y max	standard deviation
0.0445	0.0752	0.006	0.004	0.0295

Table 5.3: Average offset and standard deviation of the table detection

Table structure recognition testing results

For the testing results of the structure recognition, only the offset calculation is utilized. This approach is necessary because the evaluation using the provided testing algorithm requires word recognition, a module that is not employed in this thesis. First of all, the number of recognized elements is determined using the provided algorithm from the table transformer module. As a result of this evaluation, it becomes evident that the algorithm does not perform well in recognizing the table structure. In many cases, it either detects too many or too few elements within the table. Most notably, the header often goes unrecognized, and the algorithm frequently misses some rows and columns—particularly those located at the edges of the table. These results are summarized in the table 5.4 below. The table presents the number of labels, whether columns, rows, or header columns, that were either not recognized (represented by positive numbers) or over-detected (represented by negative numbers). A positive value indicates that certain labels were missed, while a negative value means the algorithm detected more rows or columns than the table actually contains. For the model trained exclusively on the flight test cards, the label offset was analyzed first. This analysis revealed that, in most cases, the model detected too many table elements for each table. The results are summarized in the table 5.4, which also highlights the highest average label offset count. As a result, this approach was not investigated further. The transfer learning model does not provide significantly better results, as it still varies between detecting too many or too few labels. However, the average score is better compared to the standard model, and the largest deviation from the hand-labeled data is only 11, compared to 32 in the standard model. This indicates that the transfer learning (TL) model delivers better overall results and should be used for this case study. The results are summarized in the table 5.4 and 5.5 below.

Model \ Page	1	2	3	4	5	6	7	8	9	10	11	∅
Standard	2	1	-1	-32	-11	-15	-19	2	3	3	9	0.818
ftc	9	7	11	24	24	24	24	9	11	13	27	2.455
TL	-1	2	4	11	10	8	9	-1	2	-2	-6	-0.545

Table 5.4: Missing labels in structure recognition per page and average

Model	Standard deviation
Standard	13.2879
ftc	15.3577
TL	6.5679

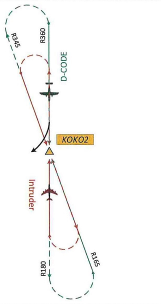
Table 5.5: Standard deviation in structure recognition

Based on the improved results of the transfer learning model, the offset is analyzed for this model, keeping in mind that it still occasionally detects too many or too few labels for the table. The results 5.6 of the offset analysis for the transfer learning model reveal that the xmin and xmax values are significantly shifted from the original coordinates. While the ymin and ymax values are not as accurate as those from the table detection, they are not as erroneous as the x values.

x min	x max	y min	y max	Standard deviation
10.8499	23.8779	0.7797	0.2606	9.6009

Table 5.6: Average offset and standard deviation of the table structure recognition

The main issue arises with tables that are uncommon, such as those containing images or tables that could be interpreted as two separate tables by the evaluator. This suggests that training the model on the dataset from a different perspective such as removing images and splitting the tables into smaller subsets could potentially improve the results.



Procedure
Initially both aircraft in formation and on target altitude to KOKO2.
When reaching KOKO2,
- INTRUDER turns on 1st O/B RAD and establishes Config.
- CODE turns on 1st O/B RAD and establishes Config.

At DIST,
- INTRUDER turns on I/B RAD
- CODE turns on I/B RAD and activates EFMS

~40sec before KOKO2
- AVOID is initiated by EFMS/S&A

When finished, set-up next run
- INTRUDER continuous on 2nd O/B RAD
- CODE returns to 2nd O/B RAD

TP #	D-CODE				INTRUDER			
	ALT [ft]	SPD [kts]	DIST [nm]	RAD [deg]	ALT [ft]	SPD [kts]	DIST [nm]	RAD [deg]
3.2 A	~~5000~~ 4800	120	9,0	R360 R165	5500	90	6,8	R180 R360
3.2 B	2000	120	8,6	R360 R165	2500	90	6,5	R180 R360

TP #	TIME			REMARKS
	START	AVOID	STOP	
3.2	07:53:00	07:56:45	07:59:00	ADS-B /radar stable track ↳ late avoid ~8nm to [illegible]
3.2	08:04:00	08:07:40	08:10:00	aircraft crossing overhead KOKO2, same altitude, 2nm
3.2	08:15:00	08:18:55	08:20:00	no radar / ADS-B track ↳ no avoid
3.2				
3.2				
3.2				

	ENG TIMES						Total time / Starts	
ENG #1	07:15	09:50					2:35	1
ENG #2	07:15	09:50					2:35	1

CODE								
SPD	100	110	120	130	140	150	160	170
PHI	10	11	12	13	14	15	16	17
DIST	7,5	8,2	9,0	9,7	10,5	11,2	11,9	12,7

INTRUDER								
SPD	70	80	90	100	110	120	130	140
PHI	7	8	9	10	11	12	13	15
DIST	5,3	6,1	6,8	7,6	8,3	9,1	9,8	11,2

Figure 5.2: Example for tables with high structure recognition failure

5.2.3 Pipeline testing results

The pipeline consists of multiple steps that primarily serve as interfaces to transfer data between modules without manipulating it. This section focuses on testing the two modules within the pipeline that are directly involved in data manipulation. The goal is to evaluate their contribution to potential errors in the pipeline, specifically regarding the storage of incorrect data.

Merge relational information and recognized content testing

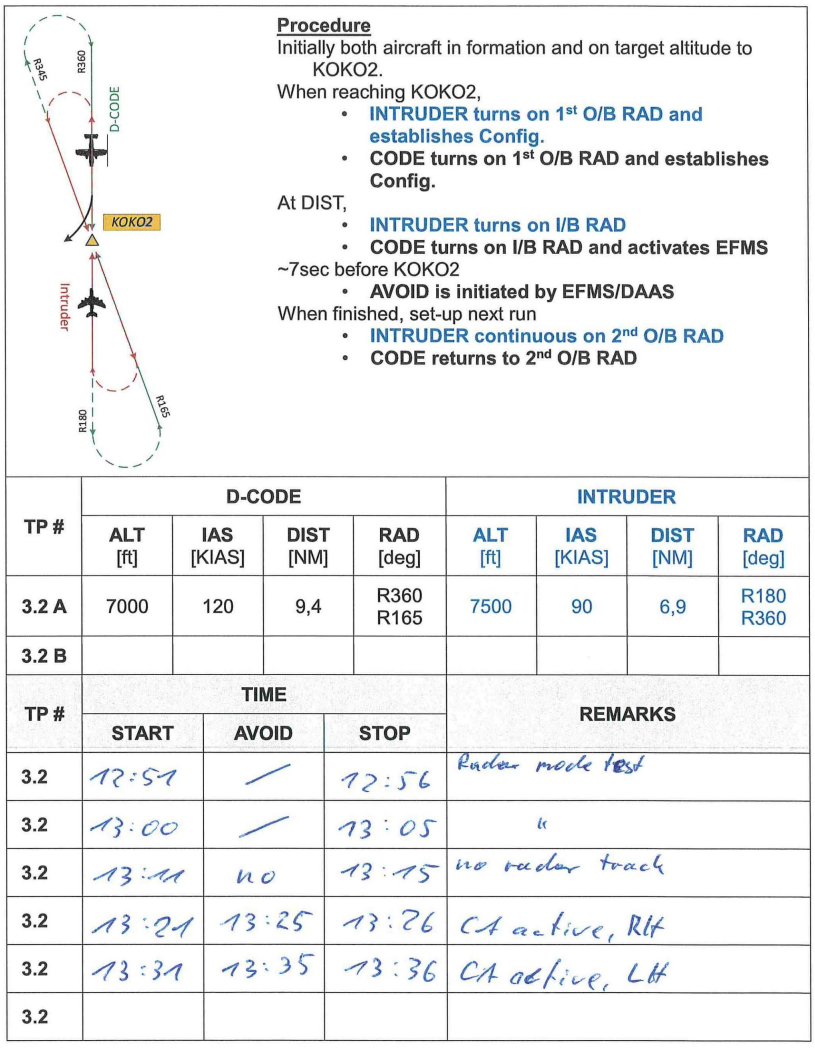
The results of the merging algorithm, which merges the OCR-D information with the information from the table detection and structure recognition, show that its performance varies based on the input data. The algorithm itself should not fail during merging, as it primarily reads a JSON file and an XML file and merges the required keys, but it could fail if it aspects any input which were not delivered by an pipeline module or there is a shift between the table structure recognition and the OCR-D. This could lead to an error reasoned in that, that the position inside the table is written in the merging algorithm. That means that only potential failure points stem from issues in the input data. Therefore, the analysis was conducted using input data from the table detection process without considering the OCR results. The main focus of this testing is to identify and categorize error classes within the pipeline that may arise during the merging process. This includes highlighting the most common errors and calculating the error rate, as introduced in the

chapter on the testing process. To demonstrate the results for valid input, a simple table 5.3 is used as an example. This table is correctly recognized by the structure recognition process and is sliced into cells based on the column and row information, which are then used as input for OCR-D. The output of the structure recognition is merged using the merging algorithm, resulting in a correctness score of 1.0, showcasing a high level of accuracy for the merging algorithm when the input data is valid and well-structured. This serves as an example of the algorithm's effectiveness under optimal conditions. This variation based on the input could occur because the merging algorithm slices the columns into cells based on the start and end x positions of the rows. If errors occur in row recognition, the merging algorithm may produce fewer data points than expected.

	Time [UTC]	Comments
ENG #2 start	12:05	
ENG #1 start	12:08	
CAB Power ON	12:09	
START Recording	12:18	

Figure 5.3: Example for tables with high correctness during merging the information

These results are compared to the merging results of one of the more complex tables form the testing dataset created for the structure recognition, which contains an embedded image and incorporates two different table structures within a single table. This comparison highlights the challenges the merging algorithm faces with more intricate input data and demonstrates how such complexity impacts the overall correctness of the merging process. The main issue in the pipeline arises during the slicing of cells from the columns using the x-values of the detected rows. If certain rows are not detected, their corresponding values are lost and cannot be merged by the algorithm. This is a critical problem that could be addressed by slicing the columns based on their borders instead of relying on the detected rows. As shown in image 5.5, the results of column extraction are consistently accurate, even for complex tables, making this approach a viable solution to mitigate data loss. This issue is further highlighted by calculating the correctness, which is 0.5847, and the average correctness for this merging process is 0.7923. These observations underscore the need for significant improvements in the structure recognition process, as it plays a crucial role in ensuring accurate slicing of cells out of the tables and merging these cells on the aspect positions.



TP #	D-CODE				INTRUDER			
	ALT [ft]	IAS [KIAS]	DIST [NM]	RAD [deg]	ALT [ft]	IAS [KIAS]	DIST [NM]	RAD [deg]
3.2 A	7000	120	9,4	R360 R165	7500	90	6,9	R180 R360
3.2 B								

TP #	TIME			REMARKS
	START	AVOID	STOP	
3.2	12:51	/	12:56	Radar mode test
3.2	13:00	/	13:05	"
3.2	13:11	no	13:15	no radar track
3.2	13:21	13:25	13:26	CA active, Rlt
3.2	13:31	13:35	13:36	CA active, Llt
3.2				

Figure 5.4: Example for tables with low correctness during merging the information

3.2	13:21	13:25	13:26	CA active, Rlt

Figure 5.5: Example for extracted table column

5.2.4 Proposed system implantation based on the systematic evaluation

As a result of the testing, the pipeline was created as described in Chapter 3.3 using the selected models. For table detection, the transfer learning model was chosen because it outperformed other models, achieving an average precision of 0.807 and delivering reliable results for the tables used in the case study. For structure recognition, the transfer learning model was also used, as it provided the best results in detecting expected labels within tables. It achieved an average offset per page of 0.545 and a standard deviation of 6.5679,

compared to alternative models with standard deviations of 13.2879 and 15.3577. While the results are on the lower end of what is considered optimal, they were still the best among the tested models. For OCR-D, the standard English model was selected, despite its relatively high Character Error Rate (CER). Compared to the trained model for handwritten data, which had a CER of 0.9654, the standard English model performed significantly better with a CER of 0.2646, making it the more suitable choice for implementation. The remaining steps of the pipeline were implemented as outlined in Chapter 3.3. The overall pipeline and the selected models are illustrated in Figure 5.6.

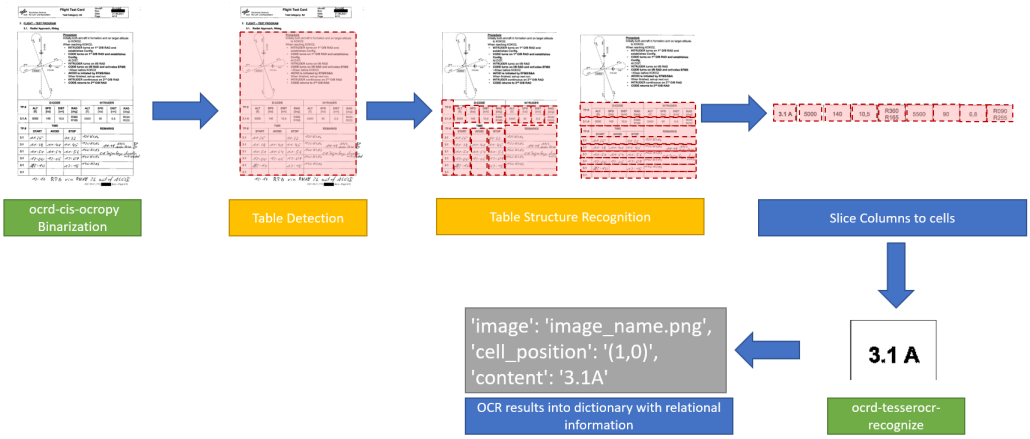


Figure 5.6: Example for pipeline with selected implantation

5.3 Ablation Study

This section presents ideas for improving the pipeline based on the results and observations made during the training process. A subset of these ideas will be evaluated to determine if they yield better results for the pipeline.

5.3.1 Possible improvements for OCR-D

For OCR-D, several approaches could potentially improve the recognition results. First, during testing, it was observed that machine-written text is recognized more accurately in larger images, even if they contain a single word or more content. This suggests that small images should have whitespace added on each side to improve text recognition. Another improvement could involve using regular expressions for timestamps, as their structure is known. This approach could automatically correct recognition errors, such as replacing a misrecognized "h" with a "4" to produce more useful results.

5.3.2 Possible improvements for the table detection

For table detection, several approaches could improve the data extraction process from the flight test cards. One such method involves extracting the tables with a small margin around the borders, particularly on the right and bottom edges. This is because flight test engineers often write content just above the borders. If the slicing occurs precisely at the border, important content from the flight test cards may be lost during table extraction. This is based on the observation that in five randomly selected flight test cards, this problem occurred in all five cases.

5.3.3 Possible improvements for the table structure recognition

A significant improvement for structure recognition could involve aligning the tables from the flight test cards with the structure of the tables used to train the base model. This could be achieved by removing images and procedures embedded within the tables or splitting larger tables into sub-tables to reduce the number of cells. Additionally, this process could include eliminating multiple headers that may appear within a single table in the DLR flight test cards. Such adjustments would simplify the table structure, making it more compatible with the base model's training data and potentially improving recognition accuracy.

5.3.4 Testing of a subset of possible improvements

This section highlights potential improvements for the pipeline extraction process. By implementing these improvements and evaluating whether the results show enhancement compared to the standard implementation, the pipeline can be further refined.

Regular Expressions for time values

There are different approaches that could be used to improve accuracy through the use of regular expressions. These could either be applied while the OCR-D is running or on the results provided by the OCR-D. The goal was to introduce an intentional error within OCR-D; however, due to the lack of information in the Wiki page and documentation on how to run digitization for individual input cells, the post-correction method was used for the ablation study instead. The post-correction method was used to evaluate whether it delivers better results compared to the standard implementation. For the post-correction implementation, regular expressions were first used to identify the required structure, which is in the format "hh:mm". As part of the ablation study, a subset of incorrectly detected timestamps was used to create a correlation table. This table provided insights into which incorrectly recognized symbols could correspond to specific numbers, based on the most common errors made by the OCR-D. The implementation then writes the most suitable solution while considering constraints, such as ensuring that start_time < end_time and adhering to the limits of 23 for hours and 59 for minutes. For this evaluation, a test set of 15 timestamps was created, along with a corresponding table for each char to number used in these timestamps, to post-correct the results. Based on this dataset, the effectiveness of the post-correction method for the timestamps was evaluated. Without post-correction, no value was recognized correctly, on values was almost correct, but with an extra symbol at the beginning and a 5 instead of a 6. The error at the beginning could be corrected using regular expressions. Also one values was not recognized at all. The simple implementation of regex-based post-correction resulted in 11 out of the 14 detected values from the OCR-D being successfully parsed into valid timestamps. This was achieved by adjusting the structure and replacing characters with the most frequently occurring number for each character. Among these, 5 values matched the expected results perfectly, with an overall Character Error Rate (CER) of 0.30, including the values that could not be found. When excluding these unrecognized values, the CER dropped to 0.1833. These results demonstrate that even a basic post-correction approach offers noticeable improvements over the standard implementation. They also suggest that with a more advanced post-correction method, it may be possible to recover all timestamps accurately.

Standard size for cells

During the evaluation of the results, it was observed that longer text or short text presented in a larger size was recognized more accurately by the OCR-D. Based on this observation, an implementation was developed to scale images to the size that produced better recognition results. This was done to determine whether the improved results were purely coincidental or if they could be consistently enhanced through this simple scaling method. The image that was recognized more accurately had dimensions of 950x80 pixels, while the others had dimensions between 420x80 and 495x99 pixels. By appending pixels to each side of the smaller images until their sizes matched, the issue of poor recognition

results could potentially be mitigated. This test examines multiple approaches, including scaling both the x and y dimensions when they are smaller than the reference size, scaling only the x dimension when the y dimension exceeds the reference size, and slicing the y dimension while scaling the x dimension. For this implementation, OpenCV in Python is used. The reference size and the image to be manipulated are provided as input. The image is read, and half of the size difference between the reference dimensions and the image dimensions is appended to each side of the image. By comparing the confidence levels provided by the OCR-D for the different inputs, slight variations in confidence were observed, as summarized in Table 5.7. The best results were achieved by appending to the x dimension while keeping the y dimension unchanged. These improvements suggest that larger-sized images are recognized more accurately than smaller ones. This is particularly evident because slicing the input along the y-axis resulted in a confidence drop of approximately 0.02 compared to other values. Overall, while resizing improves confidence, removing the borders from the image is also crucial for achieving optimal results. The differences in input configurations are illustrated in Figure 5.7.

Input Image	Confidence
original	0.9525
X and Y changed	0.9567
X changed	0.9567
X append and Y sliced	0.9362

Table 5.7: OCR-D confidence with different size inputs

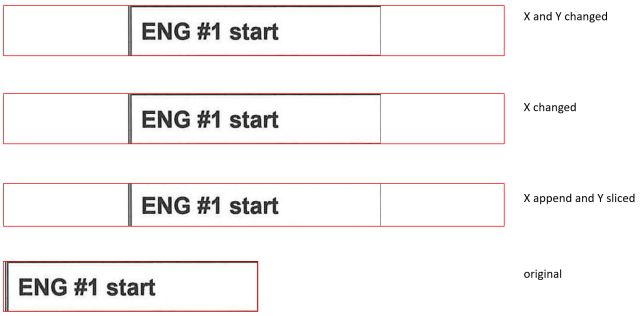


Figure 5.7: Examples of inputs for size changes for the OCR-D

Sub-tables as input for structure recognition

The testing results showed that structure recognition does not perform well on complex tables, which could sometimes be divided into two separate tables. A similar issue was observed when evaluating the merging algorithm during the extraction of cells from rows and columns. In this test case, the results of the standard implementation are compared

with those obtained by processing subsets of the more complex tables separately. An example of splitting a table into two sub-tables is shown in Figure 5.8. This type of table appears on pages 4, 5, 6, and 7 of the flight test cards. In the standard implementation, 8 to 11 labels were missing from these tables. The results of the sub-table approach show an improvement: in the first sub-table, only one label was over-detected, while in the second sub-table, three labels were over-detected. This indicates that splitting the tables reduces the number of missing labels. The results are also presented in Table 5.8. To evaluate whether these results improved detection, the average offset was also calculated. The results of this calculation are shown in Table 5.9 and indicate that splitting the tables into sub-tables significantly improves accuracy. This improvement is particularly notable along the x-axis, where the offset decreases substantially, but the y-axis also shows improvement. These findings suggest that this implementation produces significantly better results for structure recognition. This can also be observed by analyzing the standard deviation, which decreases significantly compared to the standard implementation.

Sub-table Type	over-detected labels
1	1
2	3

Table 5.8: Over-detected label in structure recognition with sub-tables

x min	x max	y min	y max	Standard deviation
0.5065	0.4022	0.1196	0.1108	0.1736

Table 5.9: Average offset of the table structure recognition on sub-tables

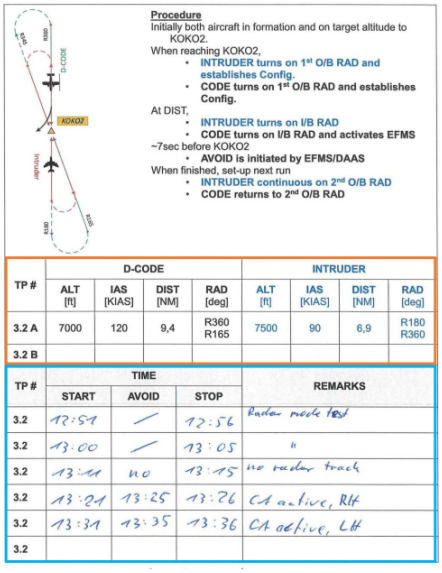


Figure 5.8: Examples of splitting one table in sub-tables for better recognition evaluation

6 Conclusion and Outlook

This section provides a summary of the thesis, outlining the work completed, a conclusion of the results achieved, and an outlook on the future steps needed to refine this concept for deployment in a production environment.

6.1 Summary

As outlined in the title of this thesis, a conceptual model and data extraction pipeline for flight test cards was developed. This pipeline incorporates table detection, table structure recognition, and OCR-D to analyze the structure and content of the DLR's flight test cards. The need for this arises from the importance of flight test cards in understanding flown maneuvers and providing better insights into the flight data stored in twinstash. Currently, the DLR stores flight test card data in PDF files, which are not efficient for quickly obtaining an overview. To address this issue, a database was created as a case study for selected flight test cards. The modules described above are integrated through algorithms written in Python to enable an automated process from PDF to database storage. As demonstrated in the results, this pipeline successfully extracts and stores data from flight test cards in PDF format. When analyzing the results, each component must be evaluated individually. Some models perform well, such as table detection, which achieves an average precision of 0.932 and a low offset compared to the reference data, with a deviation of only 0.004 in the ymax value. In contrast, structure recognition delivers poor results, often detecting too many labels for flight test cards, on average, 0.545 non-existent labels per page. However, these results can be improved by techniques such as splitting complex tables into subtables, as demonstrated in the ablation study. This improvement is reflected in the reduction of the average offset for xmin, which decreases from 10.8499 to 0.5065.A similar issue is observed with OCR-D, which has a Character Error Rate (CER) of 0.2646. This result, however, can be improved by applying regular expressions for timestamps and post-correction techniques. The ablation study demonstrates that with these simple adjustments, the CER drops to 0.3, significantly improving recognition accuracy compared to having no correctly recognized timestamps.However, it still requires improvements, as identified in the ablation study. These areas for enhancement are discussed in more detail in the outlook section.

6.2 Conclusion

The results of this thesis suggest that the proposed pipeline could provide significant advantages over the current solution used at the DLR. This is primarily due to its ability to

query flight test card data and identify flights containing specific maneuvers, which is currently not possible since the maneuvers can only be found within PDF files. This is made possible through the implementation of a relational database for the flight test cards in this case study, enabling not only searches within a single flight but also across all data in the database to find the required information. The solution presented in this thesis resolves this issue by linking the flight test card data to the flight data, enabling scientists to perform more efficient searches across these data types. The design of the database also suggests that even non-database experts can effectively perform queries. This is achieved by materializing the inheritance of individual maneuvers, eliminating the need for complex joins in queries—one of the primary challenges for users without database expertise. While the conceptual model offers clear advantages, there are still challenges to address, such as improving the OCR-D output for better accuracy and correctly recognizing table structures. How these and other issues should be resolved to ensure the system's readiness for production is discussed in detail in the outlook section.

6.3 Outlook

For future work, it must be noted that the developed pipeline provides a conceptual solution for the DLR. However, several improvements are needed before this system can be integrated into production and replace the current PDF storage method. With some improvements, this pipeline could be deployed in production, making data analysis easier compared to the current solution. First, a solution for handling complex table structures must be identified. For instance, determining the best approach to segment content from difficult tables needs further investigation. One possible enhancement is slicing large tables into smaller sub-tables and merging them after recognition, which may improve results. As demonstrated in the ablation study, this approach significantly improves the results compared to the standard implementation. Another improvement could involve enhancing the structure recognition process by detecting individual cells rather than just columns and rows. This would help minimize data loss and ensure a more accurate extraction. Additionally, the OCR-D system must be improved to better recognize handwritten data. Incorporating a text library to find the nearest matching word for OCR-D outputs could significantly enhance recognition accuracy. As highlighted in the ablation study, it is essential to evaluate the optimal size for OCR-D, as this could lead to improved recognition results. Another aspect not explored in this thesis is how users would interact with the pipeline. This raises the question of whether the process should be fully automated, from scanning flight test cards to storing data, or if a GUI should be provided, allowing users to manually upload PDF files. Finally, the database must be extended to accommodate all maneuver data, rather than being limited to the flight test card data used in this case study. Overall, the pipeline offers an approach to handling flight test cards but requires further improvements before it can be deployed in a production environment. Nevertheless, it provides a solution tailored to DLR's needs, which should be further investigated

to ensure its readiness for production.

Bibliography

[1] E. F. Codd. *The relational model for database management: version 2*. USA: Addison-Wesley Longman Publishing Co., Inc., 1990. ISBN: 0201141922.

[2] Edgar F Codd. "Further normalization of the data base relational model". In: *Data base systems* 6 (1972), pp. 33–64.

[3] Elisabeth Engl. In: *Bibliothek Forschung und Praxis* 44.2 (2020), pp. 218–230. DOI: doi:10.1515/bfp-2020-0024. URL: https://doi.org/10.1515/bfp-2020-0024.

[4] Azka Gilani et al. "Table Detection Using Deep Learning". In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. 2017, pp. 771–776. DOI: 10.1109/ICDAR.2017.131.

[5] Yasir Hamdan and Sathish. "Construction of Statistical SVM based Recognition Model for Handwritten Character Recognition". In: *Journal of Information Technology and Digital World* 3 (June 2021), pp. 92–107. DOI: 10.36548/jitdw.2021.2.003.

[6] Sebastian Haufe et al. "Digital Twins Storage and Application Service Hub (Twinstash)". In: *Deutscher Luft- und Raumfahrt Kongress 2022*. Entstanden im Projekt DIGTWIN und DIGECAT. Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V., Bonn, Aug. 2023. URL: https://elib.dlr.de/196756/.

[7] Jamshed Memon, Maira Sami, and Rizwan Ahmed Khan. "Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)". In: *CoRR* abs/2001.00139 (2020). arXiv: 2001.00139. URL: http://arxiv.org/abs/2001.00139.

[8] Clemens Neudecker et al. "A survey of OCR evaluation tools and metrics". In: *Proceedings of the 6th International Workshop on Historical Document Imaging and Processing*. HIP '21. Lausanne, Switzerland: Association for Computing Machinery, 2021, pp. 13–18. ISBN: 9781450386906. DOI: 10.1145/3476887.3476888. URL: https://doi.org/10.1145/3476887.3476888.

[9] Kate Maureen Pavlock. *Aerospace Engineering Handbook Chapter 2(v): Flight Test Engineering*. Sept. 2013. URL: https://ntrs.nasa.gov/citations/20140010192.

[10] R. Smith. "An Overview of the Tesseract OCR Engine". In: vol. 2. Oct. 2007, pp. 629–633. ISBN: 978-0-7695-2822-9. DOI: 10.1109/ICDAR.2007.4376991.

[11] Brandon Smock, Rohith Pesala, and Robin Abraham. *PubTables-1M: Towards comprehensive table extraction from unstructured documents*. 2021. arXiv: 2110.00061 [cs.LG]. URL: https://arxiv.org/abs/2110.00061.

[12] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. “Image-Based Table Recognition: Data, Model, and Evaluation”. In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 564–580. ISBN: 978-3-030-58589-1.