Institute of Aircraft Design
Lehrstuhl für Luftfahrtsysteme

Development and Implementation of a 3D-Scanning-Toolchain for the Geometric Characterization of UAVs

Bachelor's Thesis

Submitted in partial fulfillment of the requirements for the degree
B.Sc. Mechanical Engineering at the Department of Mechanical Engineering
of the Technical University of Munich.

Supervised by Prof. Dr.-Ing. Mirko Hornung
Dipl-Ing. Sebastian Köberle
Institute of Aircraft Design

Submitted by Lina van Brügge
Student ID: 03695548

Submitted on October 31, 2020

Sequential number LS-BA 20/10

Abstract

It is nowadays possible to reconstruct components via 3D-scanning, even if no geometric data is available beforehand. This process is called "Reverse Engineering". Unfortunately, this is often difficult in the case of aircraft, as they consist mostly of free-form geometries. Therefore important dimensions like the dihedral and profiles cannot be measured easily. Certain tools have to be used instead which are not spread in the public most of the time. The goal of this thesis is the development of a toolchain to extract all necessary geometries for a complete reconstruction of an aircraft and the validation of its results.

For this thesis, two aircraft were scanned and examined. A program in C++ was written for this. It is capable of generating dimensions like dihedral, twist or chord length fully automated. Also standardized airfoil profiles are saved. Then, these data was entered in XFLR5 and ADEBO to generate the aircraft models. For the validation process, the extracted airfoils were analyzed based on curvature, total difference and polars and compared to original profiles if available. Also the flight performance of the generated models was simulated.

While the extracted data is similar in many aspects to the original designs, the toolchain needs to be improved. The computed polars of the airfoils and the model have the same trend as the designs. However, the curvature oscillates at some points and the generated airfoils do not meet the accuracy of wind tunnel tolerances. An average error of $2.65 \cdot 10^{-3}$ was e.g. reached for the wing of one aircraft. The toolchain was also applied on a scanned jet for testing its behavior with different aircraft types.

Keywords: Reverse Engineering, UAV, 3D-Scanning, model generation

Kurzfassung

In der heutigen Zeit ist es mithilfe von 3D-Scans möglich, Bauteilen zu rekonstruieren und auszuwerten. Dabei müssen vorher keine Geometrie-Daten vorliegen. Dieser Prozess wird "Reverse Engineering" genannt. Bei Flugzeugen gestaltet sich diese Rekonstruktion allerdings oft sehr schwierig. Durch die komplexe Geometrie aus Freihand-Formen können wichtige Maße nicht ohne Weiteres gemessen werden. Ziel dieser Arbeit ist deshalb die Entwicklung einer Methode, um alle nötigen Geometrien für eine vollständige Rekonstruktion auszulesen, und die anschließende Validierung der Ergebnisse.

Ausgangspunkt bildeten zwei unbemannte Luftfahrzeuge des Lehrstuhls für Luftfahrtsysteme, welche im Laufe der Arbeit gescannt und ausgewertet wurden. Hierfür wurde eine Programm in C++ geschrieben. Es ist in der Lage, automatisch Maße wie den Anstellwinkel oder die Wurzellänge des Flügels aus den 3D-Scan Daten auszulesen. Zusätzlich werden standardisierte Flügelprofile gespeichert. Die ausgelesenen Daten wurden anschließend in ADEBO und XFLR5 zur Modellerstellung genutzt und das Flugverhalten simuliert. Analysiert wurden außerdem die Flügelprofile anhand von Krümmung, der totalen Abweichung und von Polaren.

Obwohl die extrahierten Daten sehr ähnlich zu den Entwürfen sind, kann die Toolchain an einigen Stellen noch verbessert werden. Die Polaren von dem Flugzeugen und der Profile zeigen ähnliche Verläufe auf wie die Entwürfe. Allerdings verfehlen die generierten Flügelprofile zum Beispiel die Genauigkeit von Windtunnel Toleranzen um eine Größenordnung. So konnte bei einem Flügelprofil eine Abweichung $2.65 \cdot 10^{-3}$ erreicht werden. Als Ausblick auf weitere Applikationsmöglichkeiten wurde die Methodik an einem gescannten Jet und damit anderem Flugzeugtyp getestet.

Schlagworte: Reverse Engineering, 3D-Scannen, UAV, Modell-Generierung

Acknowledgment

At this point, I want to thank my friends, my family and especially my supervisor who supported me during this thesis. First of all, special thanks go to my supervisor Sebastian Köberle who always provided advice and assistance when I was stuck with a problem. Additionally, I want to thank Michael Engel, Christoph Honal and Jan van Brügge for correcting my mistakes and never getting tired of discussing problems and possible solutions.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADDAM Aircraft Design Data Model

ADEBO Aircraft Design Box

AoA Angle of Attack

AVL Athena Vortex Lattice

COTS commercial Off-The-Shelf

GSL GNU Scientific Library

GUI Graphical User Interface

IMPULLS Innovative Modular Payload UAV - TUM LLS

LLT Lifting Line Theory

MLS Moving Least Square

OOBB Object Oriented Bounding Box

PCD Point Cloud Data

PCL Point Cloud Library

SFML Simple and Fast Multimedia Library

UAV Unmanned Aerial Vehicle

1 Introduction

"Reverse Engineering" has many fields of application in the modern world. In the engineering, it describes different processes to reconstruct a component even if there is no geometric data available (Valigi et al., n.d.). In this thesis, a new Reverse Engineering procedure is designed to characterize UAVs. A UAV is a pilotless aircraft which can either be programmed to fly autonomously or be controlled by a pilot located remotely (Fahlstrom and Gleason, 2012).

The design and development of any aircraft is expensive. Before the aircraft can be launched, it has to be guaranteed that it can fulfill the specific performance requirements of the project. This is done in many steps of validation and flight testing, which is time consuming, labor intensive and therefore expensive. The Matlab toolbox ADEBO developed at the *Institute of Aircraft Design* at the *Technical University of Munich* simplifies this process (see section 3.5). This software allows to design complete aircraft models using just one software. However, until today, it was not possible to fully validate the functionality of this tool on UAVs due, among other things, to the absence of enough different aircraft geometries to test the implemented methods against. In this thesis, a toolchain to support this process is developed.

1.1 Problem Description

The main objective of this thesis is to obtain geometries of a given UAV needed for a complete reconstruction in, e. g., ADEBO or aerodynamic simulation tools like XFOIL (see section 3.6). The validation of the extracted results is also an important item. For this task, two UAVs of the *Institute of Aircraft Design* were chosen: The IMPULLS (Hornung, 2011) and the DG-800 S (CARF-Models, n.d.) (see Fig. 1.1)

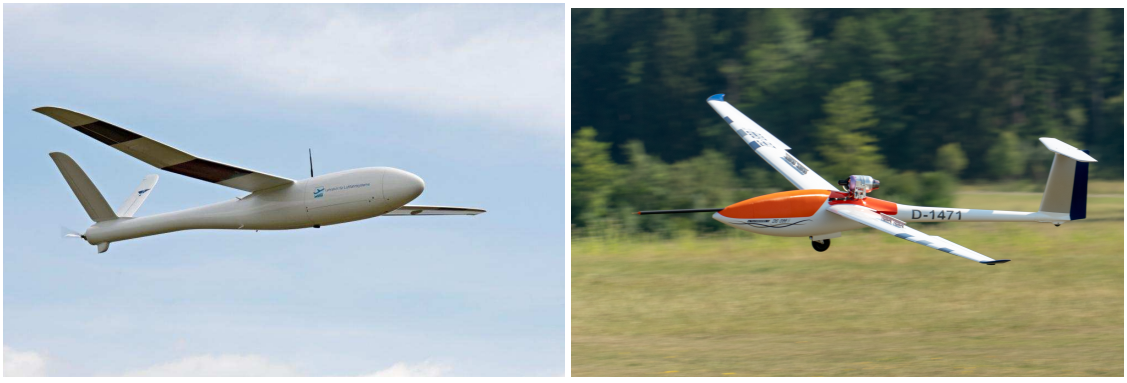


Figure 1.1: IMPULLS (left) (Hornung, 2011) and DG-800 S (right)

IMPULLS was developed at the institute and therefore has a known geometry. It is used in this thesis as a case study and to validate the developed toolchain. The DG-800 S is a commercial Off-The-Shelf (COTS) UAV of CARF-Models Ltd., Hong Kong, China. The validation of ADEBO could be accelerated if COTS aircraft could also be used. However geometric data of such UAVs is for the most part confidential and not published. The toolchain presented in this thesis is able to obtain the geometry of an aircraft only through surface examination. It can be also applied to COTS UAVs like the DG-800 S with unknown geometry. The data extracted by the toolchain is then used to reconstruct the two UAVs in ADEBO. To validate the process, the data is then compared to the original aircraft under geometric and aerodynamic aspects.

1.2 Scope of this Thesis

In this thesis, a toolchain is presented which is able to extract all needed geometries and airfoils to reconstruct an UAV model in ADEBO (see section 3.5) or airflow simulation tools like XFLR5 (see section 3.6). In order to achieve this, a 3D-scan of the UAV is generated (see section 4.1) with the hand-held scanners HandySCAN 300 and MaxShot of Creaform Inc., Lévis, Canada (see section 3.1). The scan data is then processed using a tool implemented in C++, which is presented in chapter 4. This tool is able to extract geometric aspects like dihedral or the twist of the wing. Additionally, airfoils are generated automatically and post-processed for the use in aerodynamic simulation tools.

To improve the usability for end users, a shell script was written. It invokes all needed routines and provides an automated extraction (see section 4.2). Furthermore, a GUI was implemented to ease the selection of sections on the complete UAV (see section 4.3).

The toolchain presented in this thesis is a continuation of the toolchain created by Kaan Çetin (Çetin, 2020). The preceding toolchain is summed up and presented in section 2.2.2. Additionally, current research projects at the institute with similar objectives are discussed (see section 2.2.1).

In preparation for this thesis, the geometry of IMPULLS and the DG-800 S was extracted. With this data, models in different software tools were constructed to validate the toolchain (see section 5). The models were compared by geometric aspects. Airflow simulations were also performed and then compared with available data of the IMPULLS and flight test of the DG-800 S. Additionally, a case study is presented (see section 5.3), in which different methods for the post-processing of the generated airfoils are examined and compared.

In chapter 6, the toolchain is compared to its predecessor. Weaknesses of the current method and their possible solution are also discussed. The toolchain should be capable to extract the geometry of every type of aircraft. To test this theory, a jet was scanned and the toolchain was applied to it in a final outlook on future work (see chapter 7).

2 State of the Art

In the following section, different Reverse Engineering approaches are described and then compared to the problem at hand. Following that, the theses of Kaan Çetin and Sebastian Busch are summarized. They form the basis for the toolchain presented in this thesis.

2.1 Reverse Engineering Processes

"Reverse Engineering" gained a lot of importance in the modern world in industry and research in the engineering field. There are several areas of application, one being the rebuilding of a component when no original drawing or other other construction plans are available (Várady et al., 1997). Through the collection of surface information, a given component can be analyzed and then modified if needed. Detailed data of the component at hand are fundamentally required. These can be for example obtained via 3D-scanning. There are two types of surface detection: Contacting and contactless types. Contacting methods reconstruct the surface by touching the component and evaluating the position of the sensor. Non-contact methods work by capturing reflected waves. The difference of the emitted and captured wave signals are analyzed and provide the surface information of the component. The utilized type of waves can vary: Possible are for example light, sound or magnetic waves. Any type of wave that interacts with the component in some manner (Várady et al., 1997), can be chosen. All 3D-scanning methods are limited by restrictions. Magnetic waves for example can only interact with the component if it is magnetizable. The most commonly used 3D-scanner use laser light. One method for gathering surface information is triangulation where a photosensitive device is utilized to capture the reflection of a light source and calculate the angle and distance of the component (Wang et al., 2018). These reflections are then interpreted as a point cloud or mesh of partial areas (Valigi et al., n.d.). Examples for extracting geometric information from these meshes or point clouds are discussed in the next section.

In 2005, Kim and Ahn introduced a fully-automated algorithm which extracts geometric primitives of a given point cloud (Kim and Ahn, 2005). For demonstration purposes, they used a 3D-scan of scattered building block toys where their presented algorithm separated the individual blocks and determined associated parameters. The point cloud is first split into small segments, which are then fed into an orthogonal distance fitting algorithm in order to obtain the curvature of the area e.g. linear. Then simple geometric primitives like cubes or cones are fitted to the point cloud. After the computation, the whole point cloud can be described as a set of primitives which in turn are defined by their respective position and shape. The

biggest advantage of this method is the fully autonomous extraction of the geometric data and its ease of use. Unfortunately, this method cannot be applied to the characterization of aircraft because their complex continuous exterior is difficult to describe using only simple geometric primitives.

A completely different approach for the segmentation of a point cloud and the extraction of geometries is presented in (Nielson, 2011), where a mesh is reconstructed from a noisy point cloud. Nielson uses an implicit function to describe the point cloud. Areas are fit to the cloud and then cut at the intersection of the different areas. Then the separate areas are merged into a surface mesh. In contrast to (Kim and Ahn, 2005) this method is able to describe arbitrary forms and shapes, including aircraft. This approach however is not usable for the given task. ADEBO and all airflow simulations need a certain set of text files of airfoil profiles which represent sections of the wing and other various inputs aside from shape. In the method of Nielson, only the surface information is reconstructed, the generated mesh would have to be sectioned afterwards for the airfoils. This sectioning could also be applied to the original point cloud without the additional computational costs of the mesh generation first.

A project which is very similar to the objective of this thesis is introduced by Dantsker (Dantsker, 2015). There, the characteristics of an UAV are extracted using a hand-held 3D-Scanner. After scanning the UAV outliers in the data are discarded. Then the point cloud is processed further in Matlab. The Matlab script allows the user to plot, save or adjust a given point cloud and most importantly to slice the cloud. Before the slicing, the the coordinate system of the aircraft is aligned such that the fuselage is in line with the y-axis. The output data is then processed in *XFLR5* (see section 3.6). In the work of Dantsker, it is not possible to generate lift, drag and rotational moment curvatures for the entire flight regime with *XFLR5*, but stability derivatives like C_{L_α} can be calculated within a limited angle of attack (Dantsker, 2015).

He compares the results of *XFLR5* to other simulation tools and flight tests (Dantsker and Vahora, 2018). The tools used are Athena Vortex Lattice (AVL) and *Ansys Fluent*. The former is a program which employs an extended vortex lattice model and is able to create a slender body for the fuselage, unlike *XFLR5*. The latter is a computational fluid dynamics tool and therefore more accurate in the calculation of lift, drag and moment curvatures than the low-

order tools AVL and XFLR5. The three simulation results are then compared to a flight test. He concludes that the simulations are close to the experimental results, but the used extraction method could be improved in the future.

2.2 Research at the Institute of Aircraft Design

The *Institute of Aircraft Design* examines flight performance and dynamic characteristics of aircraft weighing up to $65\ kg$. During this research, tools like ADEBO are further developed and validated. For this task, detailed analyses of the important structures of the aircraft like propeller, wing and propulsion unit are needed. Therefore, several toolchains are in development which can extract all relevant geometry and aerodynamic characteristics of the aircraft. For example, Busch created a toolchain for the characterization of propellers (Busch, 2020) and Çetin developed a predecessor of the toolchain presented in this thesis (Çetin, 2020).

2.2.1 Characterization of Propeller Geometries of UAVs

In his thesis, Busch especially focused on the possibilities and conditions for a good 3D-scan result. Post-processing of the generated data and editing of the 3D-scans are only done as much as absolutely needed in order to preserve the quality characteristics of the scan. Afterwards, the scans are compared to available CAD models or polars of the propeller geometry.

As scanning device the HandySCAN 300 of Creaform Inc., Lévis, Canada was used, a handheld 3D-scanner with a maximum resolution of $0.1\ mm$ (Creaform and AMATEK, 2018*a*). First, a default measuring method is introduced. A board with reference markers is clamped between two stands, which are also covered with positioning targets. These target points are needed as reference points for the triangulation of the surface. The propeller is held by a foam block and sprayed with white, anti-reflective powder. Scanning is done with a resolution of $0.2\ mm$ and a shutter speed of $60\ ms$ (see Fig. 2.1).

Ten different propellers were scanned in the course of Busch's work. Overall, Busch had difficulties to obtain good results at the leading and trailing edge of the propeller. He traced this problem back to the resolution of the scanner and describes the possibility of better results with a different 3D-scanner with higher resolution. To eliminate holes and inaccurate edges Busch scanned the propeller several times and combined the extracted surfaces in the software VXElements (see section 3.1).

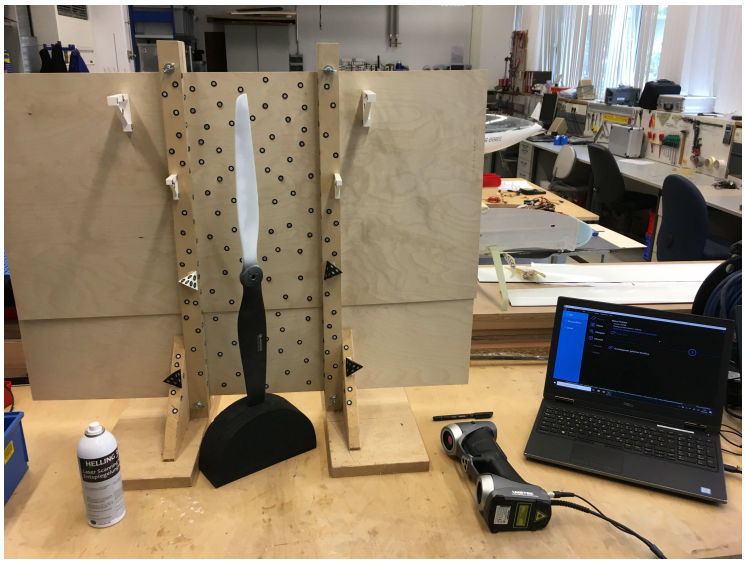
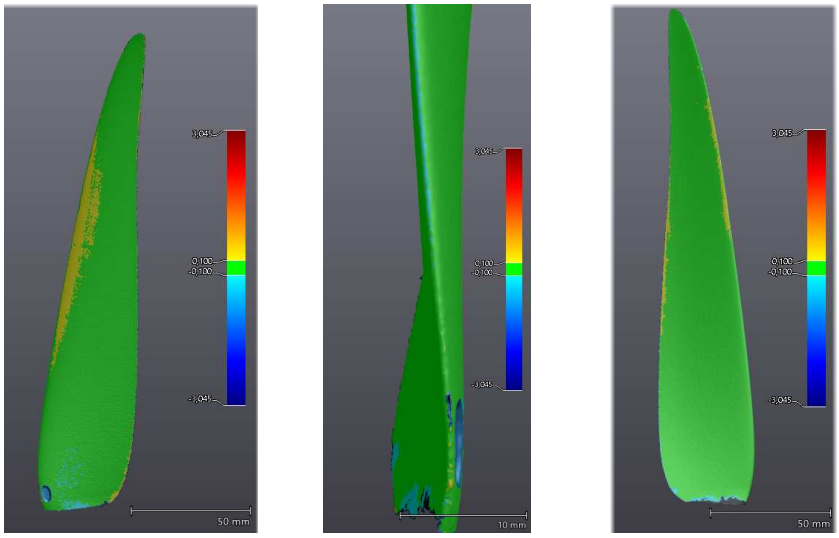


Figure 2.1: Default scanning method created by Busch (Busch, 2020)

The scan data was compared in two different manners. First, CAD models of known propeller geometries are compared to the scanned data. Most of the scan showed a deviation less than $0.1\ mm$ to the CAD model, which is significantly lower than the resolution of $0.2mm$ of the scan (Busch, 2020, S.35). The deviation increases only at the edges and in direction of the tip towards the blade (see figure 2.2). This result shows that areas without sharp edges or strong curvature can be scanned with a lower resolution compared to the edges. This is one of the reasons why a resolution of $1\ mm$ was chosen in this thesis (see section 4.1).



a) Profiloberseite **b)** Leading Edge **c)** Profilunterseite

Figure 2.2: Comparison of the scanned Madrono 2 propeller to the CAD model (Busch, 2020)

The propeller is also examined under dynamic aspects. For this, the profile of the blade has to be extracted and then analyzed by an airflow simulation tool like XFOIL or XFLR5 (see section 3.6). The obtained model is aligned such that the y-axis points to the tip of the propeller blade. Then, equidistant planes are placed along the y-axis. The intersection of the propeller and the planes is exported and processed with a Python tool. An airflow simulation is executed using some of the extracted profiles and is then compared to the polars of the original profile. The profiles show similar lift and drag polars when compared to the original profiles, but the angle of attack is shifted by several degrees (Busch, 2020).

2.2.2 Predecessor Toolchain by Kaan Çetin

Çetin restricted his work to the extraction of geometries of the DG-800 S (Çetin, 2020). His toolchain can be divided into five main steps:

1. Scanning the UAV
2. Pre-processing of the scan
3. Generating sections and extracting geometry
4. Post-processing of the generated data
5. Analyzing the extracted data

Scanning the UAV

First, the UAV is scanned. Due to the size of a UAV, the aircraft is directly covered by positioning targets instead of the background. Also, the resolution differs when compared to Busch's approach (see section 2.2.1). Çetin found that a resolution of $0.5mm$ suffices for a good image of the surface. This can also be traced back to the significant of greater size of the UAV. For the scan, Çetin not only used the HandySCAN 300 of Creaform Inc., Lévis, Canada, but also the MaxShot by Creaform Inc. (see section 3.1).

MaxShot is needed for the creation of a target model of the wing. Due to their size, an image of the targets generated using the HandySCAN is too inaccurate to form the model (see section 4.1.1). For the surface scan and the remaining parts of the UAV HandySCAN is utilized. As mentioned in section 2.1, the scan data can be interpreted as a point cloud or mesh. Çetin used the mesh mode in his work due to the possibility of normal calculation and surface optimization afterwards.

Pre-processing of the scan

After the scanning, the data is not yet ready for direct processing. The scanning was done in several spatial due to the size of the aircraft. The different meshes have to be aligned, combined and mirrored to produce the complete UAV. Also, noise and outliers have to be removed. Those are results of false reflections and triangulation errors and would interfere with the processing. The surface is then exported as a list of non-ordered points and passed to the main program. The generated 3D-scan of the DG-800 S is presented in figure 2.3.

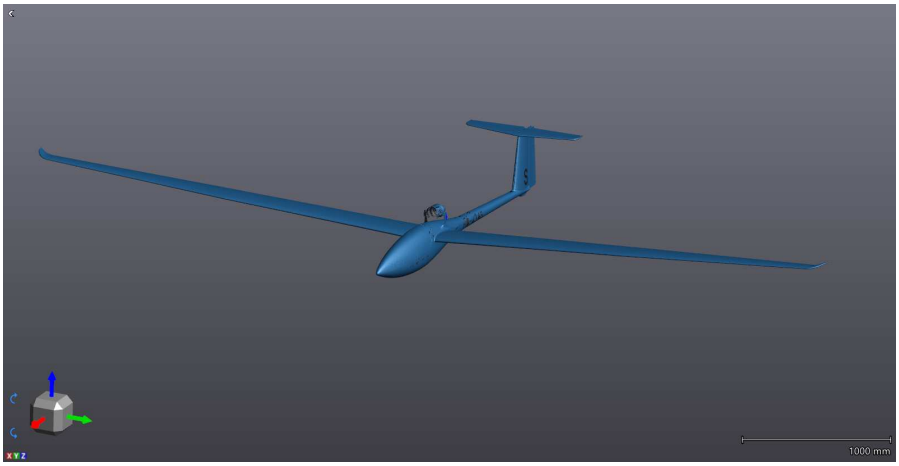


Figure 2.3: 3D-scan of the DG-800 S created by Çetin (Çetin, 2020)

Generating sections and extracting geometry

The main program of the toolchain was written in C++. The reason for this is mainly the usage of the PCL library, which provides powerful tools for processing point clouds (see section 3.2), but is only available for C++.

The point cloud representing the UAV first has to be aligned to determine the reference point of the extracted data. To do this, the smallest possible box is fitted around the point cloud and the center of this box becomes the origin. Afterwards, the fuselage and the wing is partitioned equidistantly by filtering out a thin part of the cloud at a determined distance and projecting it onto a plane orthogonal to the dihedral (see Fig. 2.4). This operation is repeated at every section boundary several times with different dihedrals. The airfoil profile is chosen via a minimization of the enclosing area of the foil: The foil with the smallest area is assumed to be orthogonal to the wing (see Fig. 2.5). The fuselage sections are generated by the same projection method. However, one section is sufficient because it is already aligned in direction of a main axis (in this case the y-axis).

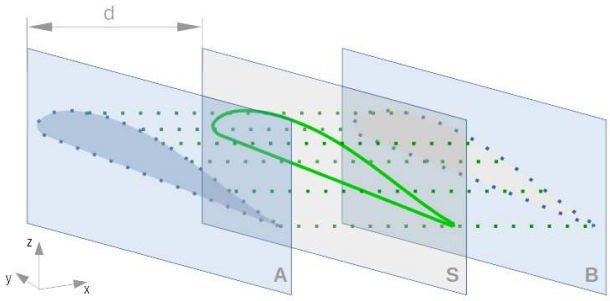


Figure 2.4: Visualization of the projection method, created by Çetin (Çetin, 2020)

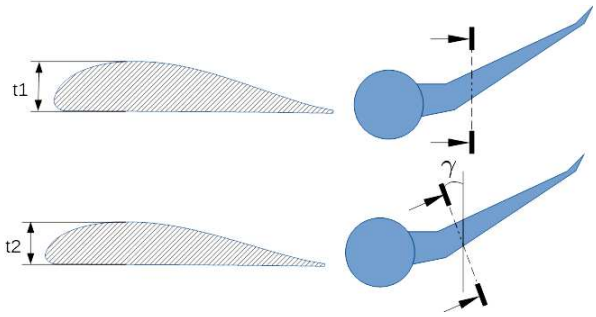


Figure 2.5: Selection of the dihedral and therefore the sectioning plane via area minimization created by Çetin (Çetin, 2020)

Post-processing of the generated data

The saved sections have to be edited in order to analyze them. Any aerodynamic simulation tool and also ADEBO expects a set of normalized, 2-dimensional points which are ordered from the trailing edge to the leading edge of the wing and back. A series of Python scripts is used for this task. They convert, order and downsize the number of data points of the generated sections.

Analysis

The data was analyzed in XFOIL (see section 3.6)), AVL and ADEBO. Çetin was able to generate polars of the wing which converged to an Angle of Attack (AoA) of 20 degrees and reconstruct the DG-800 S in AVL and ADEBO. In figure 2.6, the ADEBO reconstruction is pictured.

2.2.3 Disadvantages of the current State of the Art

Busch's and Çetin's approaches have some weaknesses and disadvantages. Both toolchains can only be run manually. The operator has to know the software tools and every separate step to generate results. Busch's whole work was done mainly in the software VXElements (see (Busch, 2020)). This software of Creaform Inc., Lévis, Canada is needed for the scan

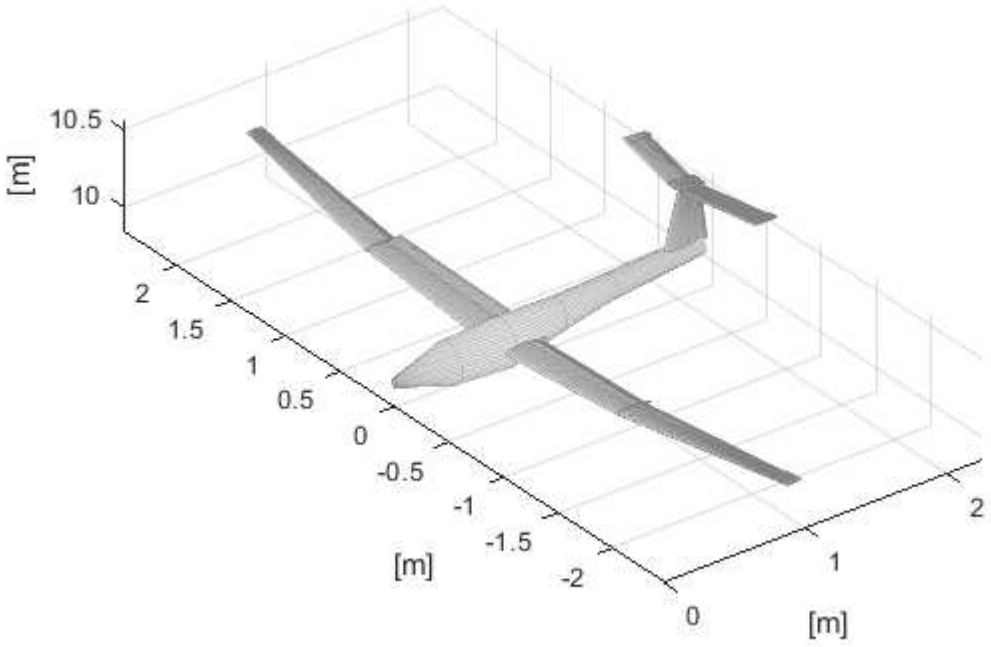


Figure 2.6: ADEBO model of the DG-800 S created by Çetin (Çetin, 2020)

generation with HandySCAN 300 or MaxShot as well as the post-processing. The usage of the toolchain therefore requires that every user has a license key and the knowledge of VX-Elements to operate it. The generated results are also difficult to reproduce because manual inputs are necessary.

This is different in Çetin's approach (see section 2.2.2). After the scanning process, his C++ and Python tools operate independent of any licensed software or tool. Still, there are many aspects which can be improved. For example, only the wing can be sectioned while the horizontal and vertical tail cannot be extracted. For the aerodynamic and geometric reconstruction of an aircraft these profiles have to be extracted manually. Also, the processing time greatly varies with the number of sections generated. With a section count of six, the processing time of the main program is between 30 and 45 minutes on an i5 quad-core CPU of Intel Corp., Santa Clara, California and 8GB RAM. Every section has to be computed several times before the minimum of the airfoil area can be determined (see Fig. 2.5). Then, the correct angle is found and thus also the dihedral. This proved to be time consuming and inefficient. That is

why a different approach was chosen in this thesis. For the sectioning, some inputs like the width of the fuselage are needed. These geometric information must be extracted by hand before processing. The Python tools also have to be started manually (Çetin, 2020). For the post-processing with Python, the sections must be copied to specific file system directories. This leads to bad usability which is improved on in this thesis.

The goal was to cut the processing time in half and decouple the computation time from the numbers of sections. This can be achieved if every section is only generated once. The toolchain should also be able to obtain the geometry of the wing as well as the vertical and horizontal tail. The sections can be placed in arbitrary distances and more geometry aspects like the position of the hinge line are extracted.

3 Applied Hardware, Software and Libraries

In this chapter, tools and libraries needed for the toolchain are discussed. Similar to Busch and Çetin (see section 2.2) VXElements is utilized for the 3D-scanning and post-processing of the scan data (see section 3.1). For the implementation, three libraries are used: the Point Cloud Library (see section 3.2), the Simple and Fast Multimedia Library (see section 3.3) and the GNU Scientific Library (see section 3.4). The generated profiles and data is then processed using ADEBO (see section 3.5) and the airflow simulation tools XFOIL and XFLR5 (see section 3.6).

3.1 VXElements

VXElements is a software by Creaform Inc., Lévis, Canada for generating, editing and analyzing 3D-scans (Creaform and AMATEK, 2019). The software is split in four different tools: VXShot, VXScan, VXModel and VXInspect. The last one is only used for analyzing and comparing the scan data to existing CAD-models or other 3D-scans and therefore was not used in this thesis. The other parts are explained in the following.

3.1.1 3D-Scanner MaxShot and its Software VXShot

The tool for the operation of MaxShot is called VXShot. MaxShot is a photogrammetric camera which can be used to generate accurate target models (Creaform and AMATEK, 2018*b*). To take an image, the object has to be covered with highly reflective positioning targets. In addition, specific square-shaped targets have to be adhered on the object and its surroundings. Pictures of each target are taken from different positions. The positioning targets are referenced through the pictures and form the target model which is passed on to VXScan for the surface extraction. This method is only needed for the target model of the wing, since in Çetin's work the HandySCAN 300 proved itself to be not accurate enough for the reference model (Çetin, 2020).

Figure 3.1: Picture of MaxShot by Creaform Inc., Lévis, Canada (Creaform and AMATEK, 2018*b*)

Figure 3.1 shows a picture of MaxShot. An exemplary target recognition structure with reference points and square-shaped targets is presented in figure 4.2.

3.1.2 3D-Scanner HandySCAN 300 and the Software VXScan

For the surface generation, the HandySCAN 300 is utilized. The corresponding software for this scanner is VXScan. The HandySCAN is a hand-held 3D-scanner which obtains surface information via triangulation. Just as MaxShot this scanner requires the object to be covered with reference points as described in section 2.2. However, low-reflective target points suffice for a recognition. After scanning the reference points, a surface model can be generated with a maximum resolution of $0.1\,mm$. The resolution is the closest distance between two measured points (Creaform and AMATEK, 2018*a*). The detailed scanning process is described in section 4.1.

Figure 3.2: Picture of the HandySCAN 300 by Creaform Inc., Lévis, Canada (Creaform and AMATEK, 2018*a*)

3.1.3 VXModel

For editing the scans, the software VXModel can be used (Creaform and AMATEK, 2018*c*). Several scans can be aligned and combined, and parts of the scan can be removed. Furthermore, it is possible to smooth, up- and down-sample the scan. Lastly, geometric primitives like planes, points and spheres can be created depending on the scan.

3.2 Point Cloud Library

The most important library used by the toolchain is the Point Cloud Library (PCL). This C++ library delivers powerful algorithms for processing point clouds (Rusu and Cousins, 2011). It was developed in reaction to 3D-scanner and 3D-sensors becoming more and more ac-

cessible to the general public, for example in form of the *Microsoft Kinect*. PCL functions as a platform for processing and editing 3D data. It provides algorithms for filtering, feature estimation, model fitting and segmentation.

PCL operates on an unified data type: The Point Cloud Data (PCD). This is a text file consisting of a header line and a list of 3D points. The 3D points can have attributes like corresponding normal vector or color, which are specified in the header. The header consists of:

1. Version: PCD file version
2. Fields: Name of the dimensions of a point
3. Size: Size of each dimension in bytes
4. Type: Type of each dimension (e. g. "f" for float)
5. Count: Number of corresponding elements in each dimension
6. Width: Number of points in the unordered cloud
7. Height: 1 for the unordered cloud
8. Viewpoint: Acquisition viewpoint for the points in form of a translation and rotation vector
9. Points: Total number of points
10. Data: Stored type of data (e.g. ASCII)

3.3 Simple and Fast Multimedia Library

The Simple and Fast Multimedia Library (SFML) is a C++ library which is designed for easy game development and rendering (Gomila, 2020). It consists of five different modules: system, window, graphics, audio and network. For the implementation only the graphic module was used, which provides functions for screen handling, drawing on the screen and input event management. The drawing process is linked to an event loop. Every time an event like a cursor click is received, the image is refreshed and presented in the opened window. In the implementation, SFML is used to provide a GUI. In this GUI, the user can choose the position and number of sections he wants to generate (see section 4.3).

3.4 GNU Scientific Library

The last library which was used in the implementation is the GNU Scientific Library (GSL). It is a extensive C library for numerical computing (Galassi, 2018). The scope ranges from complex numbers, vector and matrices operations to linear algebra, statistics and curve analysis. Only the curve analysis is important for the toolchain presented in this thesis. GSL supplies routines for spline and polynomial fitting, which is further explained in section 4.4.5.

3.5 Aircraft Design Box

As explained in section 1, the Aircraft Design Box (ADEBO) is a MATLAB tool that is currently being developed by the Institute of Aircraft Design (Herbst, 2017). It enables a more flexible and knowledge based design process. The key element is the object-oriented Aircraft Design Data Model (ADDAM). ADDAM consists of an aircraft configuration element and a supervisor class, which is able to update the aircraft model. The aircraft is divided into its main parts: Wing, fuselage, vertical and horizontal tail, propulsion unit etc. These parts are then subdivided into their characteristics like aerodynamic parameters and geometry.

In this thesis, only simple geometric studies were executed with ADEBO. Here, the fuselage, wing and tail were reconstructed in ADDAM, but for example airflow simulations were not tested (see section 5.5). ADEBO functions only as interface to e.g. XFOIL (see section 3.6) and includes no own aerodynamic analysis tools. The analyses needed for the validation therefore were directly entered in XFOIL and XFLR5 (see section 5). For a reconstruction of an aircraft, two different geometry classes are predefined: Lifting Surfaces and Bodies. The former is used for the wing and tail model and the latter for the fuselage. The fuselage is defined by ellipsoids which are translated in x- and z-direction and then merged (see Fig. 3.3). The number of ellipsoids is arbitrary and can be chosen according to the desired degree of precision. Wings, horizontal and vertical tail are defined in the same manner by using a lifting surface. This class consists of several partitions with a defined airfoil, dihedral, twist and chord length (see Fig. 3.4).

3.6 XFOIL and XFLR5

XFOIL and the further implementation XFLR5 are both aerodynamic tools. XFOIL is an interactive, low order tool for airflow simulations. It enables the design and analysis of subsonic isolated airfoils. Subroutines are for example viscous analysis of existing airfoils, design and redesign by interactive modifications like leading edge radius, blending of airfoils and plotting or writing of geometries and airfoil coordinates (Drela and Youngren, 2000).

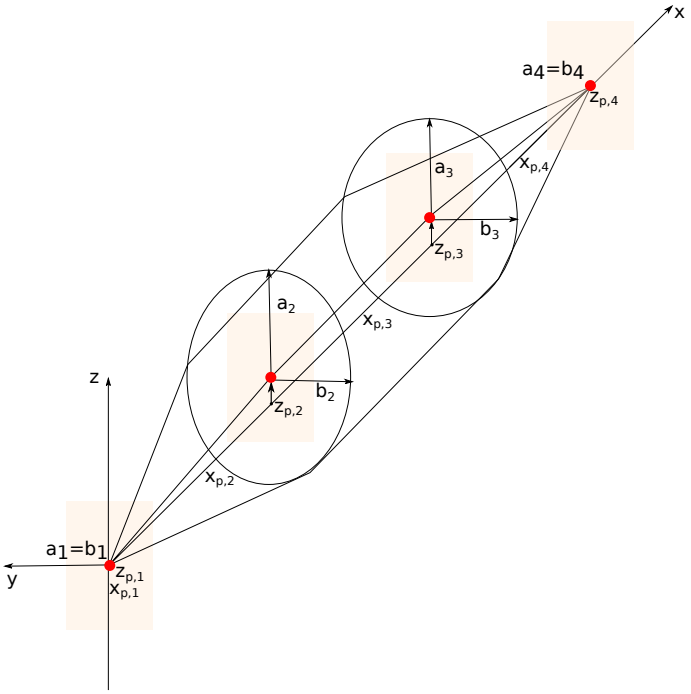


Figure 3.3: Fuselage definition in ADEBO created in accordance to Herbst (Herbst, 2017)

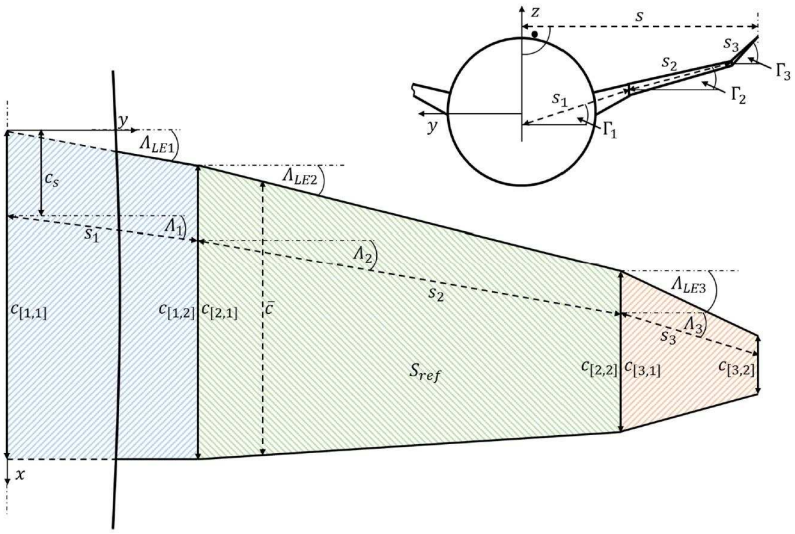


Figure 3.4: Lifting surface definition in ADEBO (Herbst, 2017)

XFLR5 is based on XFOIL. It covers most of the scope of XFOIL, but also enables the design and analysis of a whole aircraft (Deperrois, 2019). However, it is recommended only to create the lifting surfaces (consisting of wing, horizontal and vertical tail) because the fuselage leads to high computation errors. The calculated stability derivatives like lift and drag coefficients are plotted as a function of the AoA and are called polars. For the aircraft design, polars for different flight regimes have to be calculated for the separate airfoils. Afterwards, XFLR5 is

able to generate polars of the wing or the complete lifting surface. It utilizes the Lifting Line Theory (LLT), a method for the calculation of a lift distribution over the complete wing. Using LLT, it is not necessary to calculate a complex function for the whole lifting surface. The wing is partitioned instead into small sections where the lift and drag coefficients can be computed faster.

4 Functionality of the Toolchain

In the following chapter, the functionality and scope of the toolchain is presented. Basically, it consists of scanning, geometry extraction and analysis of the data. To automate the geometry extraction, a shell script passes the exported scan data to the different subroutines. A more detailed description of the shell script is presented in section 4.2. Figure 4.1 shows the operations which have to be executed manually by the user. As pictured in the figure, the complete extraction process and the post-processing of the data is fully automated.

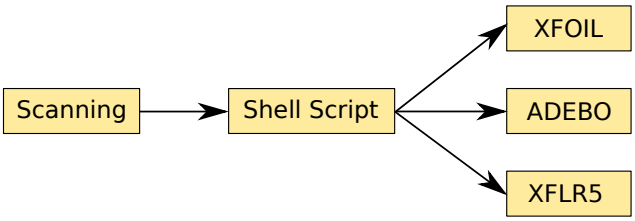


Figure 4.1: Figure of the developed toolchain for the operator

4.1 Scanning

For the scanning process, Çetin's method described in section 2.2.2 was used and adapted. First, the UAV has to be covered with reference points. The wings represent a special challenge during the scanning process. Therefore the reference points on the wings have to be highly-reflective while the remaining parts only need low-reflective targets. As mentioned in section 2.2.2, for achieving a higher precision MaxShot was used for the generation of the target model of the wing. The UAV model and the remaining target models were created with the HandySCAN 300. Since VXModel enables mirroring of scans and the UAV is assumed to be symmetrical, only one half of the UAV had to be scanned. Also, the UAV was scanned in several chunks which were combined later. As a result, the size of the separate scans decreases which accelerates the post-processing and saving of the data in VXScan.

Before the actual surface scan, a target model of the reference points has to be created. This is a requirement for the triangulation of the position using the scanner afterwards. It can either be done with HandySCAN or MaxShot. However, the generation of the model with MaxShot is more time-consuming because additional targets have to be placed. Therefore, HandySCAN was used for the model whenever possible.

4.1.1 Target Model Generation with MaxShot

MaxShot was utilized to generate a target model of the wing. During the scanning process, the wings can move. This provides an inaccurate target model when generated with HandySCAN. On the other hand, MaxShot uses more data to reference the targets. Figure 4.2 shows an exemplary setup for the model generation with MaxShot.

Figure 4.2: Picture of the structure for the target model generation with MaxShot

The square-shaped targets mentioned in section 3.1 are placed on the UAV and its surroundings. These are used to reference the small targets on the wing. VXShot uses a color code for the decoded targets. Green stands for successful and red for bad recognition. A target model is finished if all or most of the target points are colored green. Afterwards the model has to be cleaned by removing all non-green targets. Then the completed target model can be exported and sent to VXScan.

It is important that the flaps are not covered with reference points. Normally, the flaps are oriented in a non-actuated state for the scanning process. Unfortunately, the flaps can swing out during the scanning due to their mass which leads to a non-continuous surface and a incorrect airfoil geometry. Therefore, a flap derotation algorithm was implemented (see section 4.4.5) and the wing and tail were scanned with completely actuated flaps. For a correct derotation, the flap has to be actuated in direction of the hinge line. That means, if the hinge line is located on the upper side of the wing, the flap has to be actuated upwards and vice

versa. The flaps can move during the target model generation, especially if they are folded upwards. Therefore, the flaps are skipped in the generation of the target model with MaxShot. After cleaning, the model is transferred to VXScan. The flaps have to be scanned separately with HandySCAN.

4.1.2 Target Model Generation with HandySCAN

Before using HandySCAN, the scanner has to be calibrated (see (Busch, 2020)) to adjust it to the current lighting conditions. Busch found that for a good scan result, diffuse lighting works best. Then, the target model has to be created, or, in case of the wing adapted by selecting "Scan Points" in the menu of VXScan. In case of lifting surfaces, the flaps have to be covered with targets and then have to be actuated in direction of the hinge line. After this step the reference points can be recognized in the scanning process. It is important to move the scanner in a slow and continuous pattern from the center of the aircraft out. When all reference points are found, the target model can be cleaned from false recognized targets to form the basis for the surface scan.

4.1.3 Surface Scan with HandySCAN

The surface is scanned with HandySCAN. In this thesis, the UAVs were scanned with a resolution of $1\ mm$ in mesh mode. The reason for this low resolution is that the scans can be up-sampled with VXModel afterwards if needed and the scanning is accelerated. The mesh mode ensures that the surface is interpreted as a surface and not as separate points of a cloud. As a result, normal vectors can be computed and VXModel offers more options for the post-processing like a better alignment of several scans. Only one half of the UAV is scanned as mentioned in section 3.1.

The scanner has difficulties recognizing dark surfaces. For these points, a non-reflecting, white paint was sprayed on (see Fig. 2.1). Then, HandySCAN can scan the treated surfaces. This sometimes leads to a lower resolution of the scan (see (Busch, 2020)), but nevertheless suffices due to the size of the UAV.

The actuated flaps are scanned, too. It has to be ensured that the whole flap is captured as exactly and completely as possible. Sometimes, this proves to be difficult at the trailing edge due to the low resolution of the scanner. Holes can be filled after the scanning using VXModel, but this functionality is not without errors. If the UAV possesses a vertical tail instead of a V-

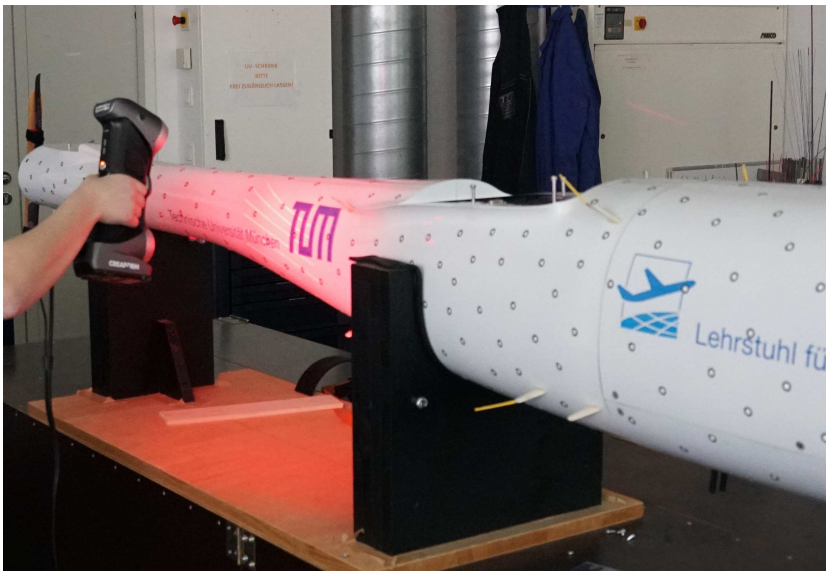


Figure 4.3: 3D-Scanning of IMPULLS

tail, it does not suffice to only scan half of the UAV. The vertical tail has to be scanned on both sides. The reason for that is the actuated flap which prevents mirroring. The picture in figure 4.3 shows an exemplary scan of IMPULLS with the HandySCAN 300.

4.1.4 Post-processing of the Scan Data

After the whole UAV was scanned, the data has to be edited. As of then, the data is noisy. This can be traced back to false reflections of the laser light. These reflections occur especially at sharp edges like the leading or trailing edge of the wing and have to be removed. In VXModel all isolated parts can be selected and deleted afterwards. Also, all holes have to be filled. To do this, VXModel provides algorithms for no, low and high curvature. The low curvature algorithm can fill holes located at the trailing edge or the wing up- and downside. The high curvature algorithm is useful for holes at the fuselage or the leading edge of the wing.

After all scans are enhanced, they have to be aligned to form the UAV. This is also a functionality of VXModel. After all scans are imported in one session, they can be aligned using a "Best-Fit"-algorithm. For this algorithm to work, the scan parts have to overlap in some area. After all symmetric parts are aligned, the scan can be mirrored. Then the vertical tail can be aligned if it exists. Once all scans are in the right position, the scans can be merged to form the complete UAV. Before the export, the scan should be upsampled. More data points ensure a more accurate result and better recognition of the hinge line but also result in a longer computation time. In this thesis, clouds of 10 to 15 million points were used for the calculation depending on the size of the UAV. Lastly, the cloud must be exported by choosing

"Point Cloud ascii" as export option. This generates a TXT-file with a list of all 3D-points in the model. The separate 3D-scans and the complete UAV-scan of the DG-800 S and IMPULLS which were performed in the course of this thesis are presented in section 5.1.

4.2 Shell Script

A shell script was implemented to enable easy use for the operator. Necessary inputs are requested and then passed to the different routines. The operator has to take care to place the point cloud into the correct folder, in order for every tool to find its input files. The script starts a sub-routine developed by Çetin (Çetin, 2020) for the conversion of the point cloud (see section 4.4.1) and executes then the main program (see sections 4.4.2 to 4.4.5).

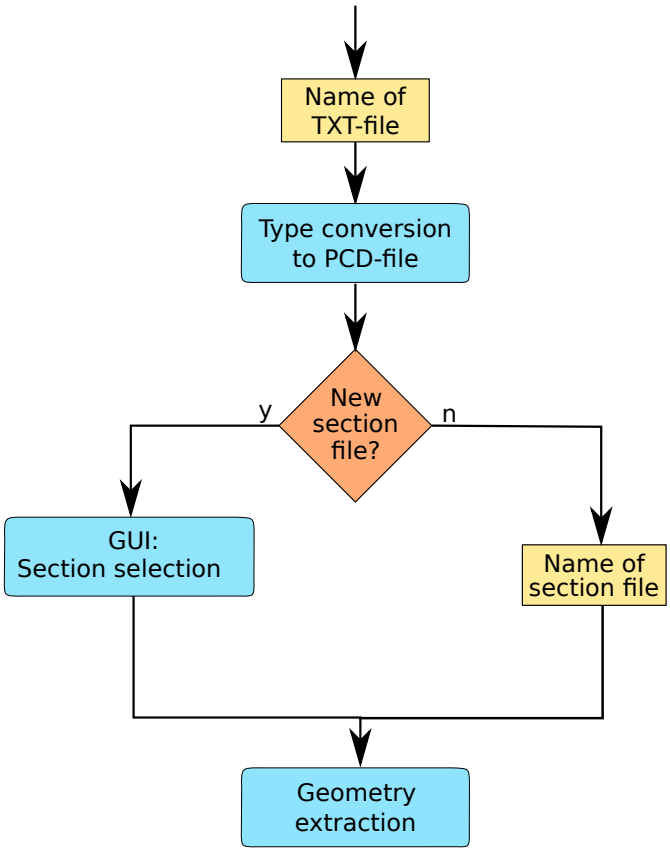


Figure 4.4: Functionality of the shell script

The inputs required by the section generation and the routines started by the shell script are shown in figure 4.4. User inputs are colored yellow, routines blue and inquiries orange. If an already existing section file is used, the software can extract all geometries automatically. If no files exist or new sections should be generated, the user has to enter "y" during the inquiry

of the shell script (see Fig. 4.4). For the generation, the UAV has to be aligned in a known coordinate system. Therefore the GUI (see 4.3) can only be executed after two steps of the main program. The shell script just sets a flag that the GUI routine will not be skipped.

4.3 Section Selection via Graphical User Interface

The implemented GUI enables the easy selection of wanted sections for fuselage, wing and tail. As mentioned in section 4.2, the GUI can only be executed after the alignment of the point cloud has been completed. However, it operates fully independent to the rest of the program. In addition, the section generation file is introduced. This file can be created by the GUI, but can also be written manually. It forms the starting point of the geometry extraction.

4.3.1 Section Generation File

The section generation file consists of five or six rows depending on the tail geometry. If the UAV has a V-tail, only five rows are needed, otherwise six because an additional part has to be sectioned. The first row can have two values, "h" or "v". "v" points to a V-tail, "h" to a horizontal and vertical tail and thus specifies the number of rows to come. If the first line is a "h" the file contains a total of six lines and in case of a "v" five. To be able to extract the wings as well as the tail, the UAV has to be split in half between wing and tail. This splitting distance is specified in the second row. After that, the actual sections are generated. The name of the section is followed by the section distances. Depending on the alignment of the UAV, the tail or the wing is sectioned after the fuselage. It is recommended to select the sections for the first time via the GUI. Then, only the distances have to be changed or added. The two kinds of different section files can be seen in figure 4.5. Sections are generated parallel to the XZ-plane in case of the fuselage and to the XY-plane in case of the vertical tail. Sections of the wing and horizontal tail are generated parallel to the YZ-plane and rotated to the dihedral (see Fig. 4.9). Further details to the section generation are given in section 4.4.4 and 4.4.5.

4.3.2 Functionality of the Graphical User Interface

The GUI was implemented using the SFML library (see section 3.3). The section generation file is created one row after another. The first row is filled with the result of an user inquiry which has to be answered in the shell. Then a window is opened where the UAV is drawn from different points of view depending on the section type. For the fuselage sections the UAV is drawn from a top down view and for the wing from a front view. Sections can be selected and deselected via cursor click. Left mouse click selects a section and right click deselects. Additionally, the UAV can be zoomed in and out using the mouse wheel. The already selected

```
h
-800
fuselage 1140 980 800 500 180 50 -900
horizontal_tail 70 150 300 410
wing 120 135 700 1450 2000 2875 2915 2975
vertical_tail -200 -135 -90 75 170
```

```
v
977.702
fuselage -1450 -925 -510 -95 600 1175 1485
wing 220 575 1150 1180 1770 2350 2400 2450
horizontal_tail 125 260 300 470 505
```

Figure 4.5: Example section generation files for an aircraft with horizontal and vertical tail (left) and with V-tail (right) with different orientations

sections are shifted accordingly, such that they stay at the same position on the UAV. Also, new sections can be selected in the zoomed view. This enables a more precise selection of the sections. It is scaled and translated to the non-zoomed view and saved in its original size in the section generation file. All selected sections are saved when the window is closed. Then, the following pop-up windows enable the selection of sections for the remaining parts. It is important for the sectioning process that the first section is the one nearest to the fuselage. Otherwise, it is not possible to extract the sweep of the wing and tail (see section 4.4.6).

The first view is a top down view on the UAV. Here, any distance between wing and tail can be selected. It is only used to split the UAV in half and has no importance apart from that. In case of a UAV with vertical tail, it could be interesting to cut near the tail, such that the vertical tail can be extracted easier. Examples for the separate steps of the GUI are shown in figure 4.6. The red lines represent the sections which will be generated.

The implementation of a section selection via text file and GUI brings along some advantages compared to the preceding toolchain. First, the sections can have different distances to each other instead a fixed one. As a result, discontinuous breaks in the wing and tail geometry (as shown in the bottom right picture of Fig. 4.6) can be recognized and subsequently extracted. Also, areas of special interest can be sectioned more often than others. In the predecessor work of Çetin, the fuselage data could sometimes not be extracted because the fuselage section ran through the wing. With the GUI the operator knows where for example the wing is positioned and can choose sections just before and after. Another advantage of the GUI is the manual wing and tail selection. As mentioned in section 2.2.2, the operator must input geometry data like the width of the fuselage. This preserves a cut through the fuselage.

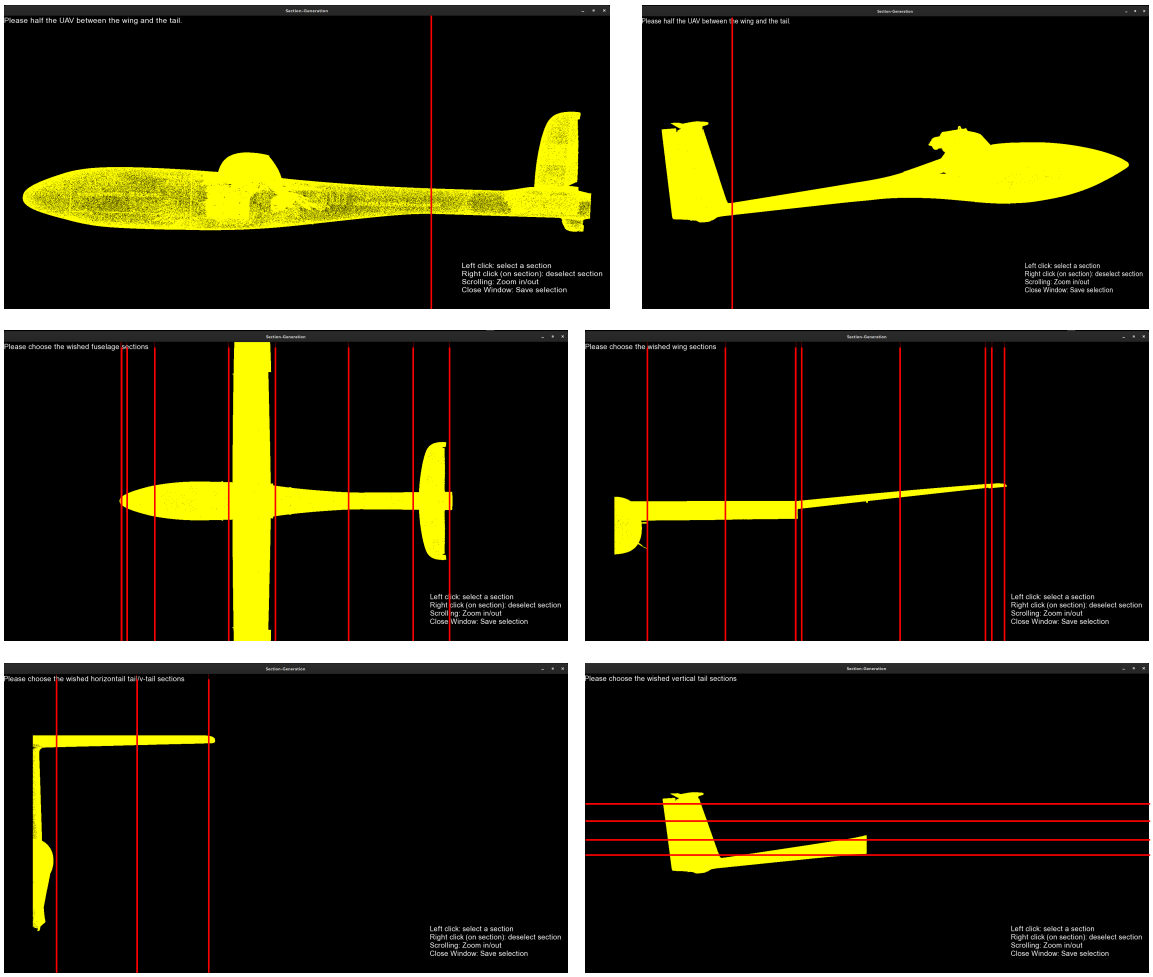


Figure 4.6: Separate steps of the section selection via GUI

Avoiding this cut and sectioning only wing or tail is now easily possible with the new GUI. Furthermore, the section generation file has to be created only one time. Having done that, the already existing file can be used again, which reduces the time. The only task of the operator is the start of the shell script.

4.4 Geometry Extraction

In the following, the main program is presented. Figure 4.7 shows the different sub-routines which are used for the geometry extraction. The GUI described in section 4.3 can either be skipped or executed. This can be selected in the shell script (see 4.2). Apart from the type conversion, all remaining program routines are written in C++ and use the PCL and GSL libraries (see section 3). The section generation routines are called multiple times depending on the number of sections chosen in the section generation file.

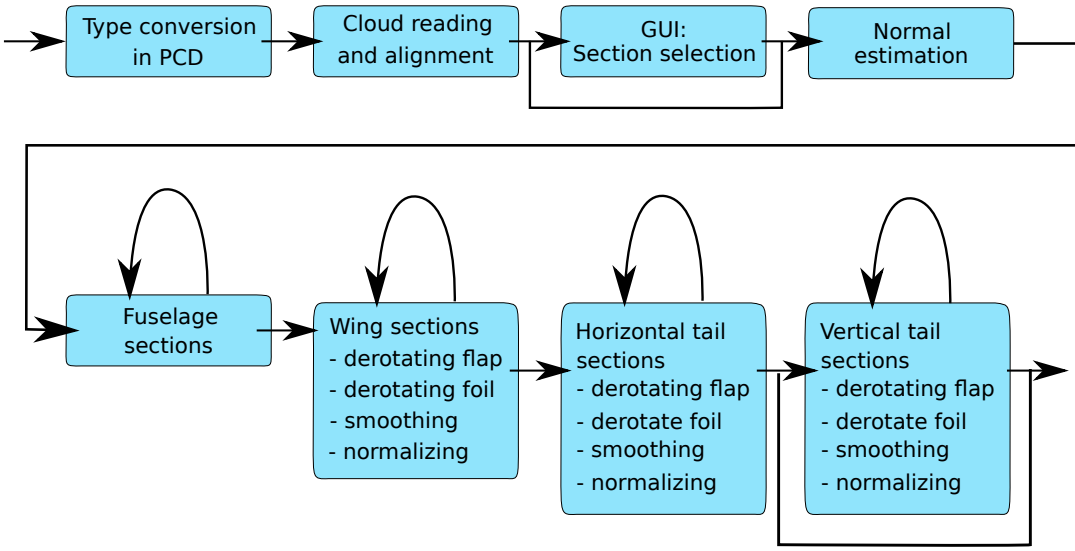


Figure 4.7: Sub-routines of the geometry extraction

4.4.1 Type Conversion in PCD File

This routine was taken from Çetin's predecessor toolchain and is therefore written in Python. The point cloud exported in section 4.1 is a TXT file with a list of all points in X-,Y-, and Z-direction. PCL uses a custom data type for the processing of point clouds (see section 3.2). For converting a TXT file to a PCD file, only the header line has to be added. In the subroutine, Çetin's program reads the TXT file, counts the number of points for the width and size of the point cloud and copies the data into a new file.

4.4.2 Cloud Reading and Alignment

The alignment of the UAV also comes from the predecessor toolchain of Çetin. The alignment is a prerequisite for the usage of the tools of the downstream workflow. With the scan of the target model, VXShot or VXScan choose a coordinate system for the model. This is needed as a reference point for the scanning process but the origin is located outside of the object and the orientation is unknown. It is assumed that the UAV is symmetric. It is reasonable that the origin must be located on the symmetry axis for a correct extraction of the geometry. This is normally not the case and the point cloud has to be aligned to achieve this.

Here, the principle behind the alignment is explained shortly, for further information see (Çetin, 2020). The method is based on bounding boxes. These are cuboids which surround the point cloud completely. Now, the smallest box which fits around the UAV is chosen. It is called a Object Oriented Bounding Box (OOBB). The goal is then to transform the point cloud that the center of the OOBB becomes the center of the general coordinate system and its axes the

new main axes. The cloud is therefore translated and rotated. This functionality is provided by PCL via one of its subroutines. After the transformation, the complete orientation of the UAV is still unknown. Nevertheless, the orientation of the axes are known. The Y-axis runs parallel to the fuselage and the Z-axis parallel to the vertical tail. For a right-handed system, the X-axis points to the wing, but there is no information if the UAV is for example upside down (see figure 4.8). These cases have to be caught in the implementation (see section 4.4.5).

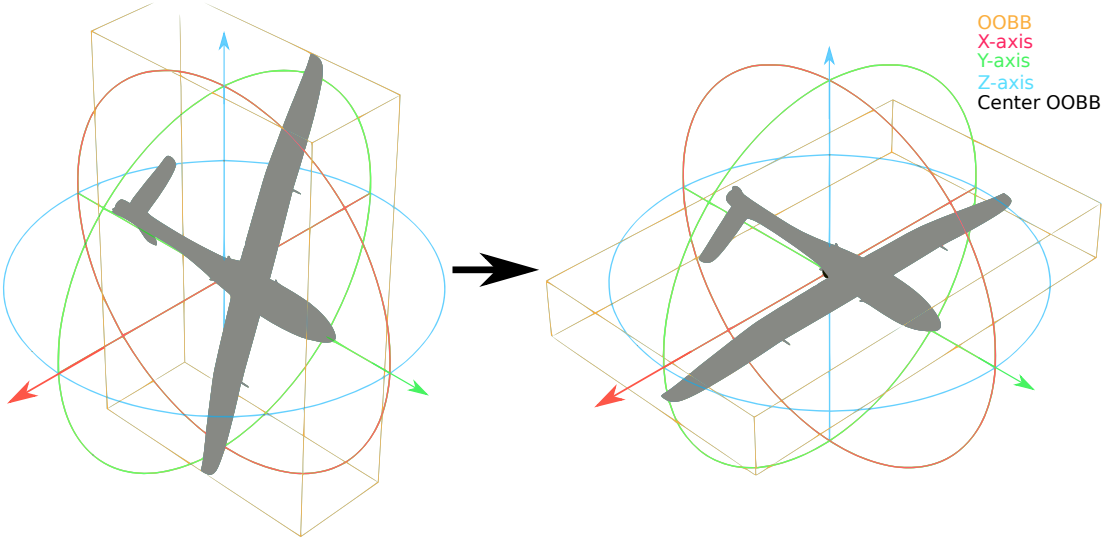


Figure 4.8: Exemplary alignment of a point cloud via OOBB

4.4.3 Normal Vector Estimation with PCL

Normal vectors are estimated for each point in the cloud. Thus for a denser point cloud more computation time is needed. Since the point cloud consists of several million points, this subroutine takes the longest to execute together with the alignment and is the main factor of the whole computation time. For the estimation of the normal vectors, PCL provides different options. In this thesis, an approach with a Moving Least Square (MLS) algorithm was utilized. It is also used for surface reconstruction and therefore has an additional smoothing algorithm implemented (see (Rusu and Cousins, 2011)). For the computation, the points in the cloud are ordered via a K-d tree. Then, a nearest neighbor search is performed. This is a method of searching for either a specific number of data points which are closest to the selected point or in a defined radius around the point. In this case, a plane is constructed to which the normal vector is orthogonal. A search with defined radius is executed. The computed normal vector is then stored in a new point cloud together with the corresponding point.

4.4.4 Extraction of the Fuselage Geometry

The fuselage is the easiest part of the sectioning. This has several reasons. First, the fuselage is a round body. In ADEBO this body is approximated using several ellipses which are translated (see section 3.5), whereas XFLR5 explicitly recommends not to define a fuselage (see section 3.6). Because of that the fuselage sections have to be generated but must not be changed afterwards. Second, for cutting a specific area of the fuselage, no transformation of the aircraft is needed. As mentioned in section 4.4.2, the Y-axis runs parallel to the fuselage. The cutting plane for the fuselage is therefore just the XZ-plane translated to the cutting distance. Then, no further alterations are needed for the extraction of the geometry.

The points of the sections are saved as a standard point cloud TXT file consisting of 2D-points in ASCII. Until now, these files are not used in the toolchain, but could gain importance in the future (see section 7). The width, height and center of the sections are computed additionally and written in several TXT files. In each file, one line consists of the corresponding value of one section. This means that the width of the second section in the section generation file (see section 4.3.1) is stored in the second row of the fuselage width file (see tables 5.3 and 5.1).

4.4.5 Extraction of the Airfoil Geometry

The geometry of wing, horizontal and optional vertical tail is extracted using the same method. It is executed iteratively and consists of five steps:

1. Sectioning
2. Flap derotating
3. Airfoil derotating
4. Smoothing
5. Normalizing

The results are several normalized and ordered airfoils consisting of about 160 points and geometry aspects like dihedral, twist and chord length which are saved in several TXT files. XFLR5 (version 6.47) is able to work with airfoils containing up to 250 points (Deperrois, 2019), XFOIL (version 6.9) is able to work with even more points (Drela and Youngren, 2000). During the smoothing process, the airfoil is up- and then downsampled to 160 points even if XFLR5 and the other tools could also use profiles with more points. The reason for this is the further processing of XFLR5 and the other airflow simulation tools. After the loading of

the data points, a spline curve is fitted to the points of the airfoil and enables an up-sampling and redistribution of the entered data points. This redistribution of the foil has to be done regardless of whether it contains 250 or 160 data points to improve the results. The lower point density enables XFLR5 to modify the foil as well as possible.

Sectioning

In contrast to the fuselage, the wing and horizontal tail usually do not run parallel to the defined X-axis (see section 4.4.2) and can therefore not be sectioned by a translated unit plane like the YZ-plane. The reason for that is the dihedral Γ of the wing and tail, which symbolizes a rotation around the Y-axis. For correct foil extraction, the sectioning plane has to be orthogonal to the wing. Figure 4.9 shows the definition of the dihedral and its relation to the sectioning plane.

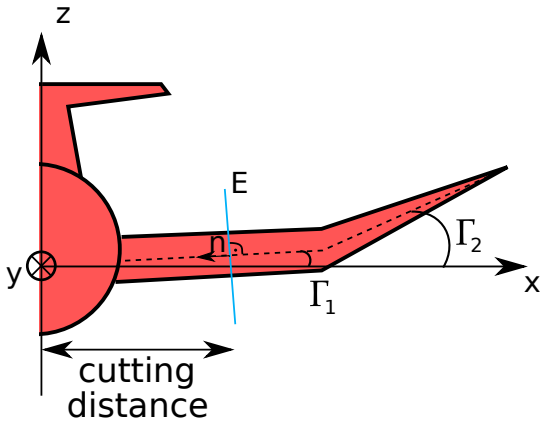


Figure 4.9: Sectioning of the wing

Since the dihedral is unknown and also has to be extracted, it is important to find the correct angle of the sectioning plane. Çetin's method proved to be time consuming and not very efficient (see section 2.2.3). An approach with the aid of surface normal vectors was chosen for this task.

As explained above, the sectioning plane has to be orthogonal to the wing and tail and runs also parallel to the Y-axis since the dihedral is an indicator for the Y-rotation of the wing. A plane can be constructed using two non-parallel vectors and one point of the plane (see equation 4.1).

$$E = (\vec{e_y} \times \vec{n_s}) \circ (\vec{X} + \vec{OP}) \tag{4.1}$$

$\vec{e_y}$ symbolizes the unit vector in y-direction, $\vec{n_s}$ the normal of the surface and $\vec{OP}$ the point of the plane. For the construction of the sectioning plane only the surface normal was unknown in the original point cloud. After the computation of the normal vectors using MLS (see section 4.4.3), the normal vector of the sectioning plane can now be calculated directly. Unfortunately, the normal vector calculation is only an estimation and therefore not always correct, even when using the complex MLS-algorithm. The reason for that is that for example small surface defects of the scanning process can sometimes not be smoothed completely if the defect is too big. It is therefore not sufficient to use just one normal vector at the desired cutting distance but rather it is necessary to form an average of several normal vectors. In the program, a small stripe is sectioned of the wing in the desired cutting distance. Then, all normal vectors of the new point cloud are summed up and form $\vec{n_s}$ of equation 4.1.

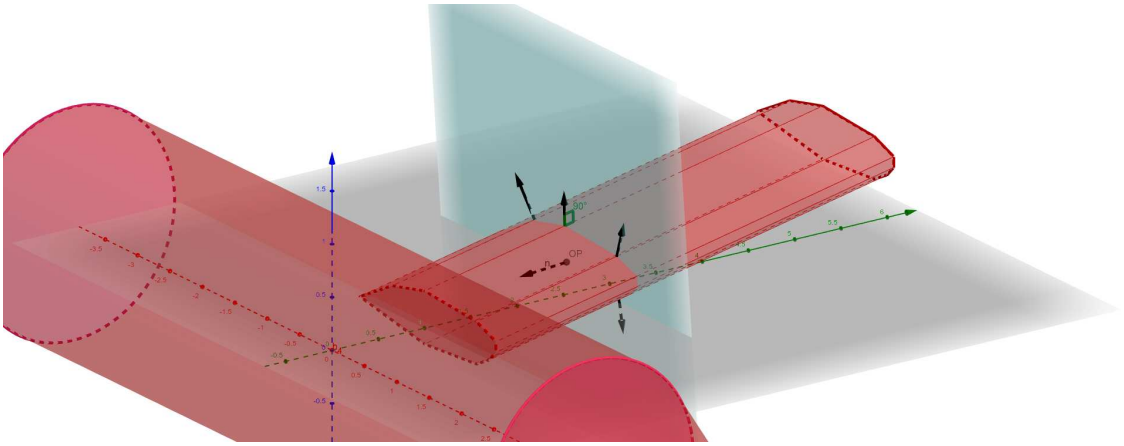


Figure 4.10: Simplified graphic of the plane construction with surface normal vectors for the wing

In figure 4.10, the whole process is shown. The dihedral Γ can then be calculated through the normal vector of the plane (referred to as $\vec{n}$ in Fig. 4.10). It is the angle between the normal vector $\vec{n}$ and the XY-plane (see figure 4.9) and is computed with equation 4.2.

$$sin(\Gamma) = \frac{|\vec{n} \circ \vec{e_z}|}{|\vec{n}|} \tag{4.2}$$

During the export of the mesh to the point cloud (see section 4.1) all surface and link information between the separate points is lost. The MLS-algorithm provides an even arrangement of the points, but does not guarantee that an airfoil can be generated using only one plane.

Instead, a small stripe is filtered out of the point cloud and then projected onto the calculated plane. This method was introduced by Çetin (Çetin, 2020) and has been improved in this thesis.

PCL enables filtering of point clouds, however it is only possible to filter in the direction of one of the main axes. To apply the filter the wing has to be transformed. The easiest transformation of the point cloud is a rotation of Γ. This ensures the alignment of the wing to the X-axis. Due to the transformation the cutting distance has changed. If the UAV has several different dihedrals (see figure 4.9), it is not possible to compute the new cutting distance with equation 4.3. Aside from the dihedral of the current section, the remaining wing geometry, like a second dihedral or the position of a bend is unknown. A new cutting distance after a bend however would depend on the different dihedrals as well as on the position of the bend and is therefore not computable.

$$d_{new} = \frac{d_{old}}{cos(\Gamma)} \tag{4.3}$$

For this reason, a different approach was chosen. One point of the section in the old cutting distance is selected. The point changes its coordinates during the transformation. The new cutting distance then is the new X-value of the selected point (see figure 4.11).

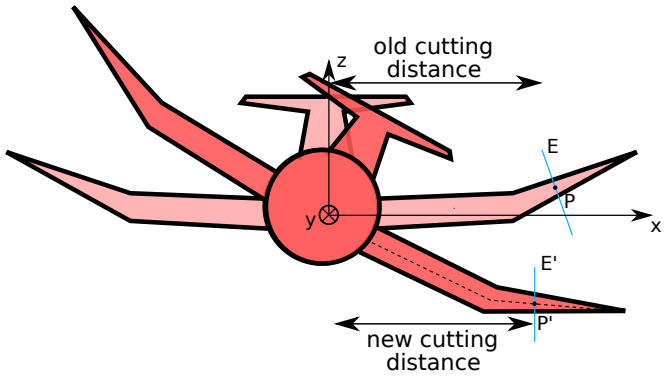


Figure 4.11: Rotation of the aircraft and finding the new cutting distance

The section generation is based on Çetin's method (Çetin, 2020). At the new distance, a filter removes all points outside a given minimum and maximum distance - in this case the new cutting distance is translated to a defined distance d (see Fig. 2.4). Then all points in the filter region are projected on the plane defined in equation 4.1 (in Fig. 2.4 named S) as described in section 2.2.2). The projection is then transferred to the next sub-routine.

For the vertical tail, the routine is simpler. Like the fuselage, it is aligned to a main axis, in this case the Z-axis. So, for the sectioning only the XY-plane is translated and used as the sectioning plane. Then, the airfoil is created by the projection as explained above (see figure 2.4).

Flap derotating

As mentioned in section 4.1, the UAV was scanned with actuated and deflected flaps because the flaps tend to move during the scanning process and are difficult to adjust completely to the rest of the wing. For the computation of the foil in XFLR5, XFOIL or ADEBO a continuous airfoil without flap is needed. So, the flap has to be derotated before the smoothing and normalizing of the airfoil.

For a good airflow, an airfoil has to have a continuous profile. Due to the actuated flap, this is no longer the case: The actuated flap provides a discontinuous bend at the hinge line. With the normal vectors computed in section 4.4.3, the hinge line can be found and the flap can be derotated. If the foil section does not have a flap, the routine recognizes this and skips the current foil. The method for the derotation can be divided into several routines:

1. Finding trailing edge
2. Finding the hinge line
3. Derotating the flap

Finding the trailing edge of the foil is the key for the correct recognition of the hinge line. This proved to be difficult due to the low resolution of the trailing edge during the scan process (see section 4.1). Without knowing the exact orientation (see section 4.4.2) a method had to be developed which extracts the position of the trailing edge only through general aspects of an airfoil. The selected approach uses the fact that an airfoil is wider at the leading edge than at the trailing edge. A nearest neighbors search of the foremost and furthest point of the section is performed and the maximum distance between all neighbor points is computed. The end with the smaller distance is the trailing edge (see Fig. 4.12).

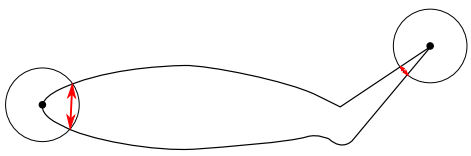


Figure 4.12: Search for the trailing edge via nearest neighbors search, the red lines represent the maximum distances

An iterative routine is now started to find the hinge line. This routine is the reason for the actuation of the flap in the direction of the hinge line (see section 4.1) and not in an arbitrary direction. A discontinuous bend exists only at the hinge line, on the other side there is a gap covering. The algorithm iterates through all points of the upper part of the foil (if the hinge line is on the top side) until the hinge line is found. If the flap is actuated upwards, that means the hinge line is on the top side and the upper part has to be examined and vice versa. The search algorithm works with a comparison of the curvature which can be expressed via the computed normal vectors (see section 4.4.3). If the angle between the normal vectors of two neighboring points is greater than seven degrees, a candidate for the hinge line is found. The upper border of seven for a hinge line recognition was determined by several test during the thesis. This search process is the reason for the search of the trailing edge beforehand. At the leading edge the curvature and therefore the difference in the normal vectors is quite high and would be recognized as the position of the hinge line. At the trailing edge this problem does not arise if one makes sure that only one - the upper or lower part - is searched through. During the loop an average normal vector of the flap is computed similar to the normal calculation for the sectioning. An average normal vector of the foil before the hinge line is also computed. Then, the angle between these two normal vectors is then the required rotation angle α. The complete search process is visualized in figure 4.13.

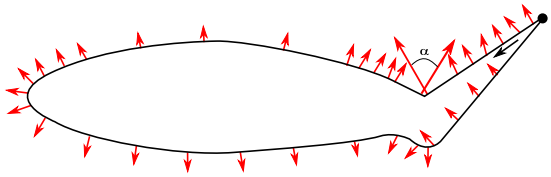


Figure 4.13: Method for the hinge line search

The last step of the routine is the actual derotation. For this, the rotation angle α and the position of the hinge line is needed. The foil is split at the hinge line into two separate point clouds: the flap and the foil without the flap. Then, the flap is rotated around the X-axis with α. The orientation of the flap is unknown. As a result, the flap is rotated in positive and negative direction. To select the right rotation an average normal of the now rotated flaps is computed and compared to the average normal of the foil (see above). The gap covering is removed by deleting all points ahead of the hinge line. Finally, the foil is concatenated. Figure 4.14 visualizes the complete routine.

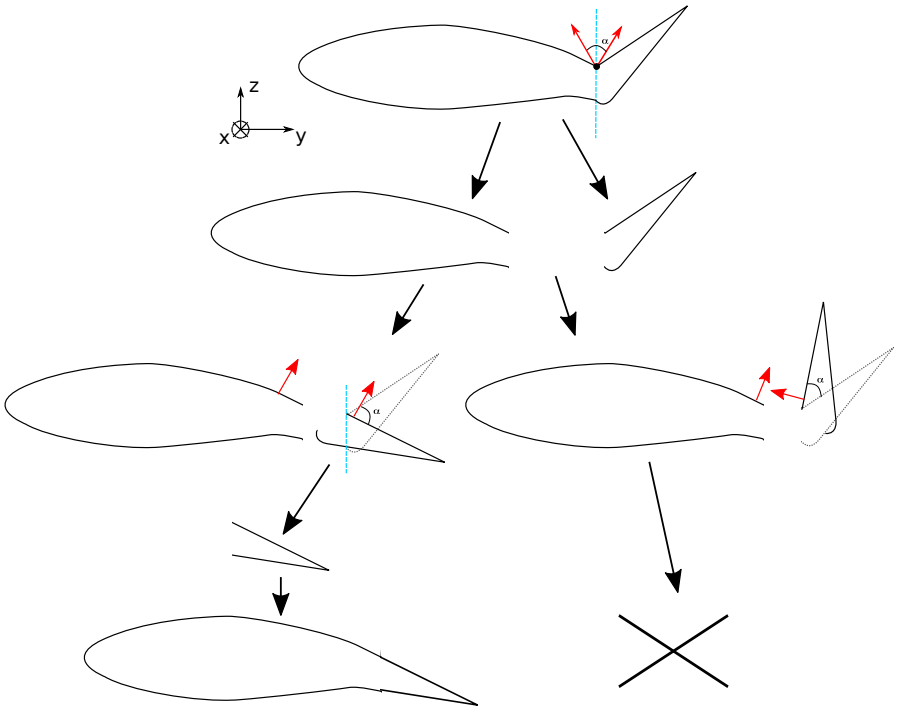


Figure 4.14: Flap derotation method

Airfoil derotating

After the flap is derotated, the foil has to be derotated. In this step, the twist angle Θ of the airfoil and therefore of the wing can be computed. A line from the trailing edge to the leading edge is created. The angle between the line and the XY-plane is the required twist angle. The airfoil is then rotated by this angle (see Fig. 4.15).

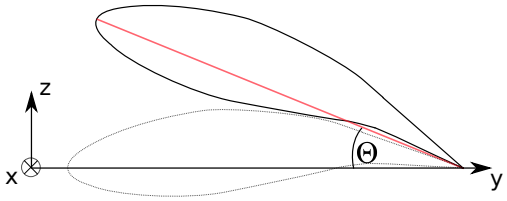


Figure 4.15: Derotation of the airfoil and extraction of the twist angle Θ

Smoothing

To exclude scanning defects for the calculation of the polars the airfoil is smoothed. The implemented method fits a polynomial to the upper and the lower side of the foil. So far, the sectioned and derotated airfoil is available as a non-ordered point cloud. For the fitting function a monotonous increasing set of 2D-data points is needed. That means, the airfoil has to be split into an upper and lower half and then ordered. A simplified skeleton line of the airfoil is computed for this task. The middle of the foil is computed in equidistant steps and connected

to its neighbors to form the skeleton line. The algorithm tests for each data point in the airfoil if it is located above or below the skeleton line and therefore partitions them into two different clouds. These can be ordered using a standard function of the vector class in C++. Figure 4.16 shows an extracted airfoil profile and its computed skeleton line in red.

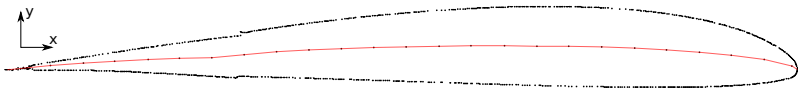


Figure 4.16: Splitting the airfoil in upper and lower half

Now, the GSL library (see section 3.4) is utilized. At first, a cubic spline is interpolated on the 2D-data. The spline is computed using a spline fitting method which was introduced by Steffen (Steffen, 1990). For a standard cubic spline fit, third-order polynomials are fitted onto partial segments of the data. The first and second derivatives remain continuous. This method is an easy way to smooth a curve, however, the spline tends to oscillate especially if the fitted points are not spaced equidistantly. This oscillation problem occurred also during this thesis. For this reason, the spline fitting method of Steffen was chosen. In his method, only the first derivative is continuous. In addition, every cubic function is in its interval monotonous. That means, that maximum and minimum can only occur at the given data points. With this method, an oscillation of the spline can be prevented.

It is important that this spline fit is an interpolation method. All data points remain unchanged. It can only be used for upsampling or distributing the given data points. In this thesis, both options are utilized. The reason for this is the subsequent polynomial fitting for the smoothing. The airfoil is upsampled to 800 data points with a Chebyshev node distribution. Especially the distribution after Chebyshev provides a stable polynomial fitting without oscillation. The reason for that is, that the error function of the polynomial fit has its minimum with the node Chebyshev distribution (Stewart, 1996). Equation 4.4 shows the X-value distribution for a stable and converging polynomial fit.

$$x_k \; = \; cos(\frac{2k-1}{2n}\pi), \; k \; = \; 1, ..., n \tag{4.4}$$

For the fitting process two different polynomial degrees were chosen depending on the maximum difference to the Y-axis. This means that mostly the lower part of the airfoil is fitted with a polynomial of order 10 and the upper part with one of the order 16. The selection process of

the two degrees is explained in section 5.3.2). The last step is to downsample the two parts and then concatenate them. The lower part is copied inversely for this task. So, the leading edge of the lower parts comes just after the leading edge of the upper part.

Normalizing

The airfoil extraction is finished after a normalization process. The leading edge is located in the origin and the trailing edge at $(1, 0)$. Also the orientation is important, but with regard to the airfoil still unknown. Three options have to be considered.

- The foil is upside down.
- Trailing edge and leading edge are exchanged.
- The foil is not ordered from the trailing edge to the leading edge and back.

The first option can be examined using the skeleton line (see Fig. 4.16). The skeleton line of a symmetric airfoil is a line from leading to trailing edge. Then, it is irrelevant which side is the top. By non-symmetric foils with camber the skeleton line is always curved to the top side. So, when the middle of the extracted points of the line is below the origin point, the foil is upside down and has to be mirrored at the X-axis (see figure 4.16).

The other options are processed with the position of the trailing edge. The airfoil is now ordered and distributed symmetrically. It suffices to select the foremost and furthest point in the airfoil and then go an even number of points to the middle. Then, the minimum distance is computed to find the trailing edge similar to figure 4.12. If the position of trailing edge is smaller than the position of the leading edge, the foil has to be mirrored at the y-axis. Finally, the foil has to be ordered originating of the trailing edge and normalized to one. To achieve this, the trailing edge is shifted to the first position and every point is divided through the chord length of the foil. During the result generation the problem occurred that the bad resolution of the trailing edge results in a non-converging airflow computation. The smoothing of the polynomial does not suffice to improve on this fact. To avoid this problem, all points near the trailing edge are removed before the saving. Figure 4.17 shows an exemplary output of the sectioning routine.

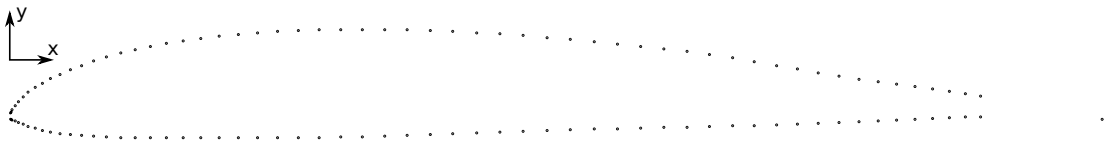


Figure 4.17: Airfoil of figure 4.16 after flap derotation, smoothing and normalizing

4.4.6 Extraction of the wing and tail characteristics

The routines described in section 4.4.5 are used for the extraction of the airfoil for wing and tail. To compute the vertical tail with the same routines as the horizontal tail and the wing the created section is rotated by 90 degrees around the Y-axis. Thus, the vertical tail airfoil has the same orientation as the projected wing and horizontal tail sections and can be passed to the same routine.

Similar to the fuselage geometry extraction (see section 4.4.4), some TXT files are generated. Besides the saving of the airfoil after the post-processing (see section 4.4.5) in these files are stored the remaining geometrical aspects of interest: The dihedral Γ in degrees, the position of the hinge line in percent, the chord length in millimeter and the twist Θ in degrees are saved. Additionally, the offset of the separate foils is computed. This is the absolute distance between the leading edge of the first section and the current section (see tables 5.2 and 5.4). The leading edge is found with the same method as pictured in figure 4.12. Then the positions of the two leading edges are compared. With this calculation, the sweep Λ of the wing and tail can be computed with equation 4.5.

$$\Lambda_i = arctan(\frac{y_i - y_0}{x_i - x_0}),\ i = 0, ..., n \tag{4.5}$$

The position of the first leading edge is also saved to this file. No distinction is made whether the first section is the section which is nearest the fuselage. This is the reason why the sections should be ordered from the fuselage to the tip when creating the section generation file (see section 4.3.1). Figure 4.18 visualizes this routine.

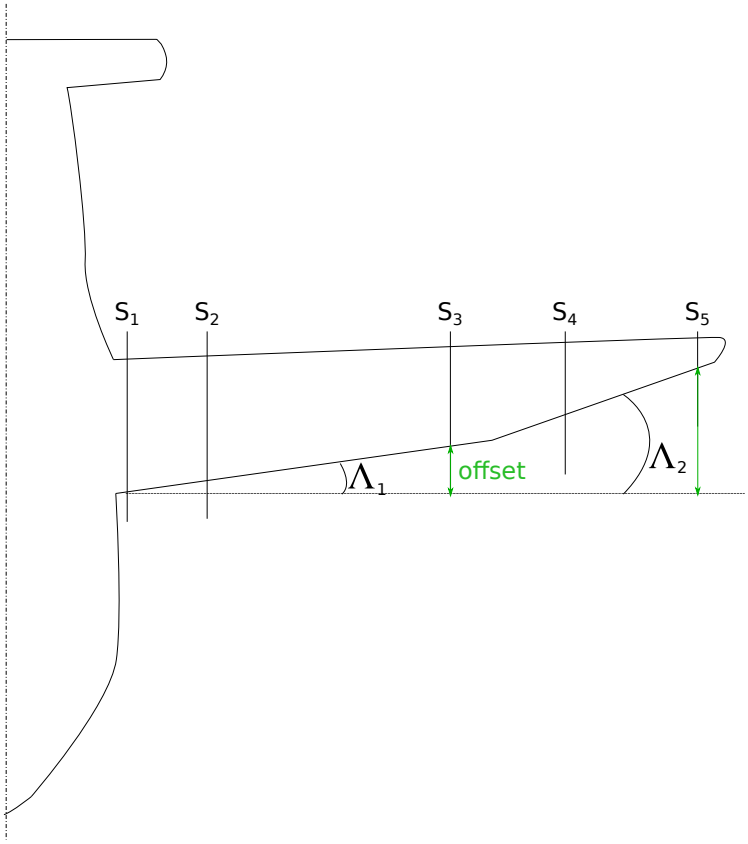


Figure 4.18: Computation of the offset and the sweep Λ

5 Results

In the following the generated results are presented and discussed. The 3D-scans of IMPULLS and the DG-800 S are presented (see section 5.1). They form the basis for all results afterwards. The airfoils generated with the method described in section 4.4.5 are presented and compared to the available original profiles of IMPULLS (see section 5.2). In this thesis, a study of fitting functions was conducted (see section 5.3). The results of this study justify the selection of the smoothing approach explained in section 4.4.5. In this study, two different aspects were evaluated. The fitting function itself and the polynomial order of the target function. Finally, the aircraft models generated in ADEBO (see section 5.5) and XFLR5 (see section 5.6) are discussed.

5.1 Scans

The scans were generated with actuated flaps and a resolution of $1\ mm$ as mentioned in section 4.1. The UAV was scanned in several parts. The separate parts were the fuselage, the wing and the tail. The fuselage was also split in three chunks due to its size. The results are presented in the following.

5.1.1 3D-Scan of IMPULLS

The UAV IMPULLS was scanned in five parts: The front, middle and back part of the fuselage, the V-tail, and the wing. In figures 5.1 and 5.2 the extracted surface data is presented. Outliers and noise were removed beforehand as described by Çetin and Busch ((Çetin, 2020), (Busch, 2020)).

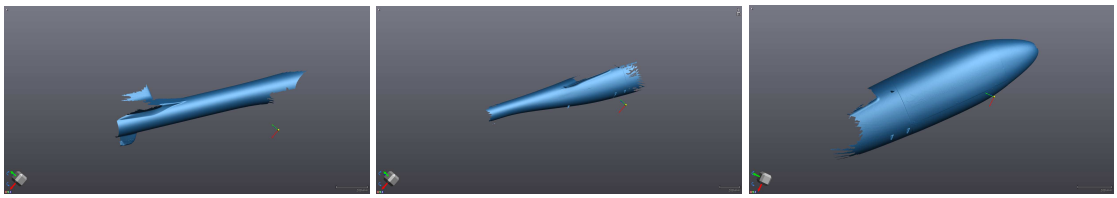


Figure 5.1: 3D-scans of the fuselage of IMPULLS: back (left), center (mid), front (right)

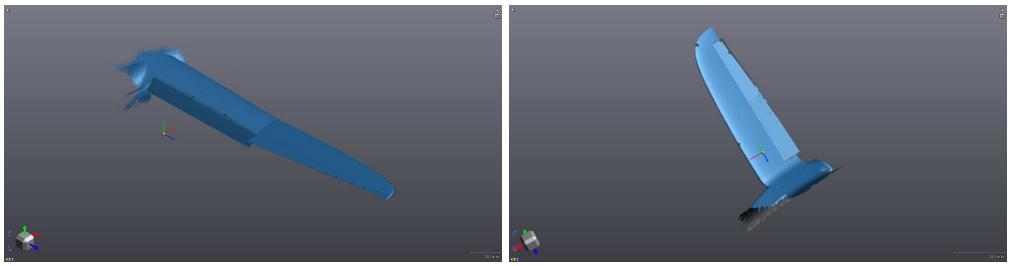


Figure 5.2: 3D-scans of the wing (left) and v-tail (right) of IMPULLS

These scans were combined using a "Best-Fit"-algorithm and then mirrored as discussed in section 4.1. The complete UAV can be seen in figure 5.3.

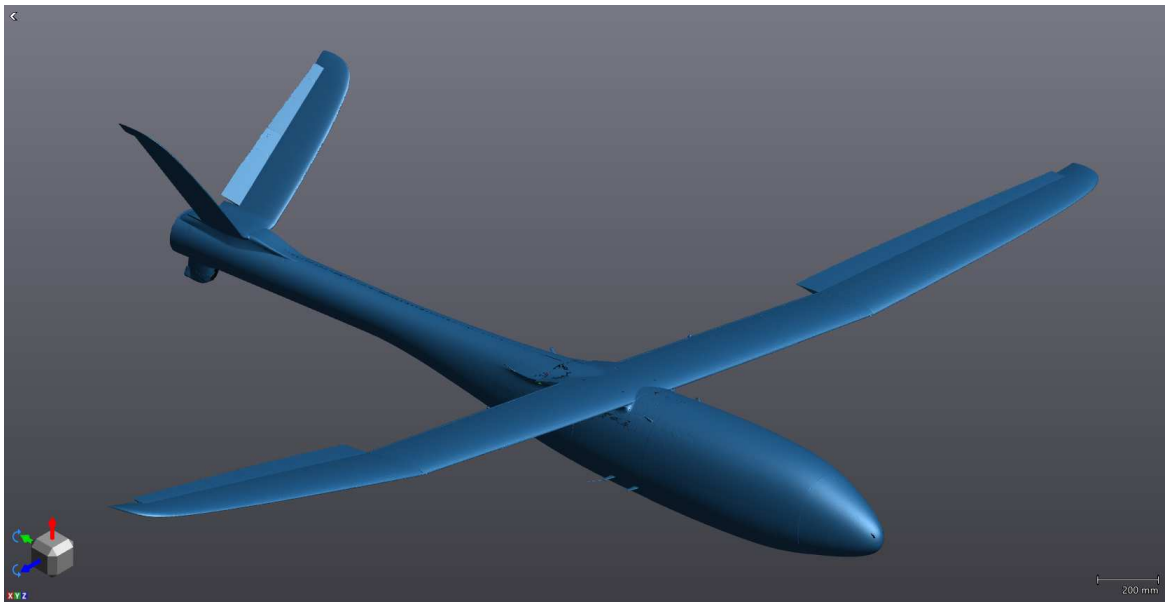


Figure 5.3: Combined and post-processed 3D-scan of IMPULLS

5.1.2 3D-Scan of the DG-800 S

In case of the DG-800 S, the 3D-scan created by Çetin (see (Çetin, 2020)) already existed, but it had been created with the flaps aligned to the rest of the wing. Consequently, the wing and the tail were scanned with actuated flaps, the fuselage was taken from the already available scan (see section 5.4).

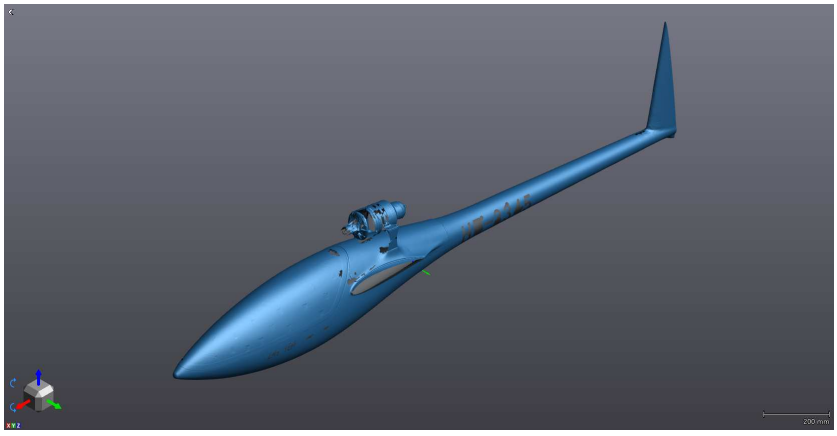


Figure 5.4: 3D-scans of the fuselage of the DG-800 S created by Çetin

The scans which were generated for this thesis are shown in figure 5.5 and 5.6. As mentioned in section 4.1, the vertical tail had to be scanned from both sides because it is not symmetric when the flaps are actuated. The remaining UAV was mirrored, then the combined vertical tail was fit to the fuselage. The complete UAV is presented in figure 5.7.

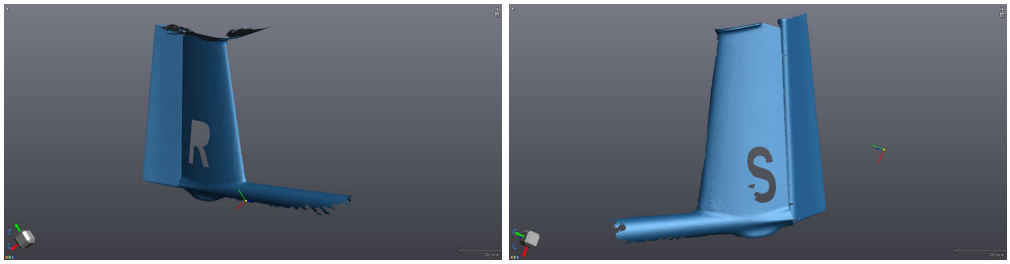


Figure 5.5: 3D-scan of the vertical tail of the DG-800 S from both sides

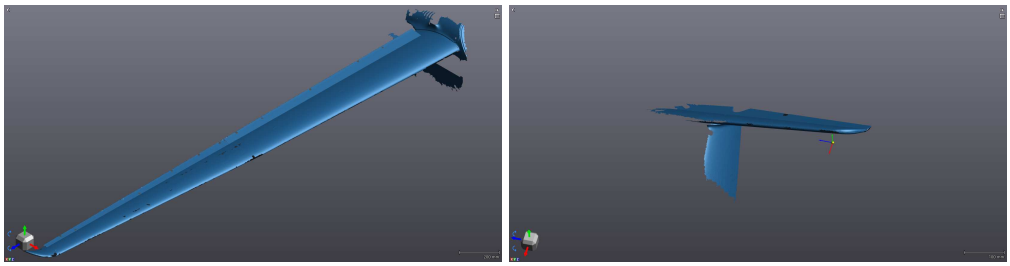


Figure 5.6: 3D-scan of the wing (left) and horizontal tail (right)

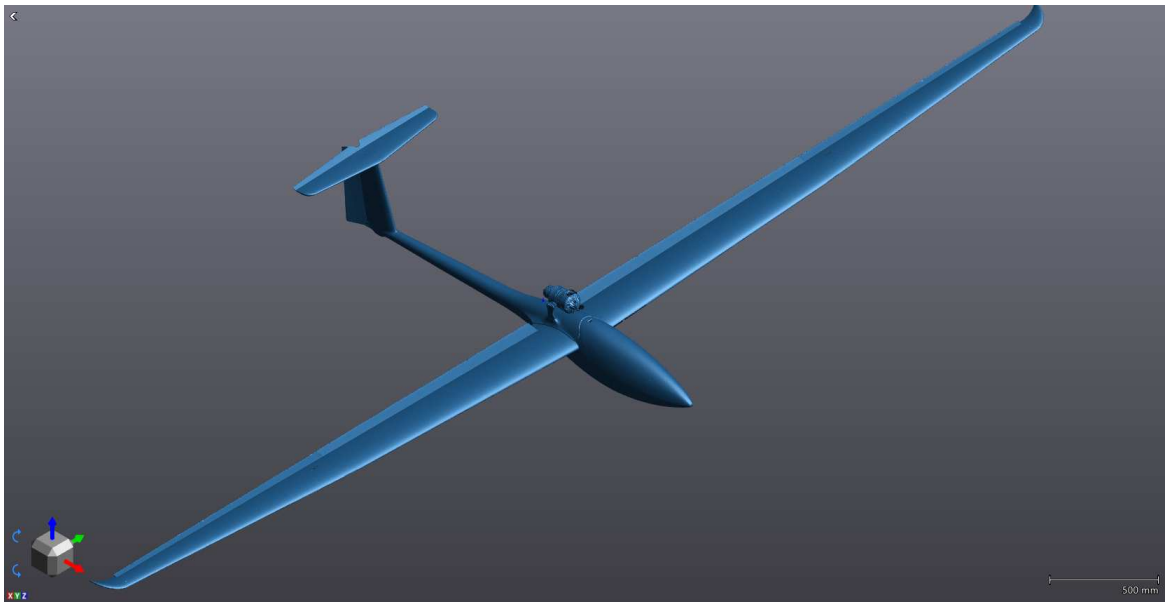


Figure 5.7: Combined and post-processed 3D-scan of the DG-800 S

5.2 Airfoil Results

In this section, the airfoil extraction method described in section 4.4.5 is validated. Exemplary airfoils are presented and then compared to the original airfoil in terms of curvature, total difference and polars. A foil with derotated flap and one without flap was chosen for the UAV IMPULLS. This enables analyzing the accuracy of the derotation algorithm.

5.2.1 Airfoil Results of the wing of IMPULLS

For the comparison of the generated airfoils to the original airfoil, sections at $220\,mm$ from the root of the wing and $2400mm$ from the tip were chosen. The former generated a foil containing a derotated flap and the latter one without. The foils are compared in terms of curvature, total difference to the original foil and by polars.

Comparison by curvature

The curvatures of the two extracted airfoils do not differ from each other. The curvature is generated by the smoothing via polynomial fit (see section 4.4.5). Since the airfoil was fitted with a polynomial degree of the same degree for the upper and lower side of the airfoil, the curvature is also the same. For the upper side of the airfoil, the curvature diverges hardly from the curvature of the original foil. The lower half diverges significantly from the original airfoil. In figure 5.8 the curvature is shown. There, the light blue and orange line are the curvatures of the original foil and the other two represent the extracted section.

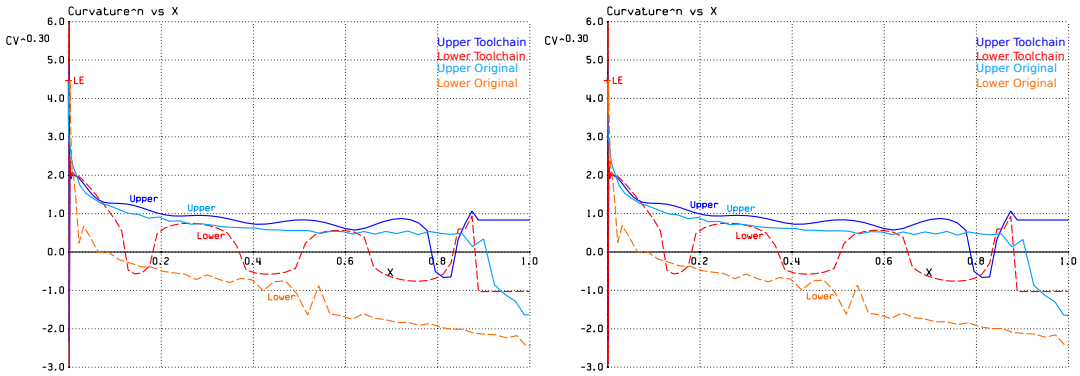


Figure 5.8: Curvature comparison of the generated sections at $220mm$ (left) and $2400\,mm$ (right) to the original airfoil profile

Comparison by total difference

Two routines are used to compute the total difference of the two foils. The first one utilizes a spline interpolation implemented in the GSL library to upsample and redistribute the data points to a Chebyshev node distribution (see section 4.4.5). When applied to the original and

the scanned airfoils, it is guaranteed that every point is compared to the corresponding point of the other airfoil. The second routine plots the foils and the difference between them and was created by Çetin (Çetin, 2020). The plot consists of two parts. The upper part is a plot of the compared airfoils and the lower part shows the total difference between them. The goal was to achieve the accuracy of wind tunnel tolerances as they are presented in (Kulfan and Bussoletti, 2006). They can be described with equation 5.1. y and x are the values of the airfoil data points and c represents the chord length. If the tolerances are met, it can be assumed that the two airfoils behave in the same manner in airflow simulations and flight testing.

$$\text{error} = \frac{|y_{scan} - y_O|}{c} < \begin{cases} 4 \cdot 10^{-4} & \text{if } x/c < 0.2 \\ 8 \cdot 10^{-4} & \text{if } x/c > 0.2 \end{cases} \tag{5.1}$$

Figure 5.9 shows the resulting error between the extracted airfoil and the original airfoil. The average difference between the two airfoils is $2.65 \cdot 10^{-3}$. It is evident from figure 5.9 that the error increases in direction of the tail. This can be traced back to the derotation of the flap. Unfortunately the accuracy of wind tunnel tolerances could not be achieved. The error is about one magnitude larger than the tolerances (see equation 5.1) permit. However, the results are based on scans which are subject to a relatively low scan resolution of $1\,mm$ and also manufacturing tolerances of the UAV.

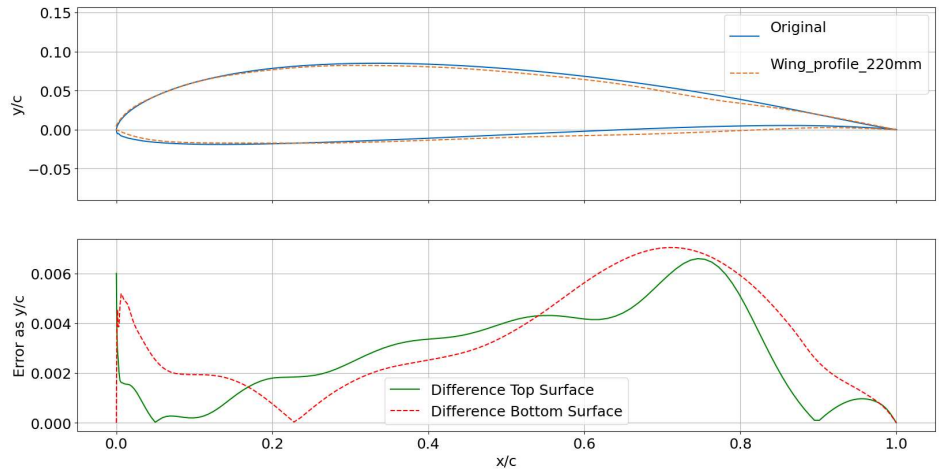


Figure 5.9: Total airfoil differences between the section at $220\,mm$ with derotated flap and the original airfoil

The foil derotation has some impact on the result as can be seen in figure 5.10. The average error is about two thirds of the error with a derotated flap and amounts to $1.7 \cdot 10^{-3}$. It should be noted that the bottom surface hardly differs from the original foil. Nevertheless, overall this section also cannot meet the wind tunnel tolerances.

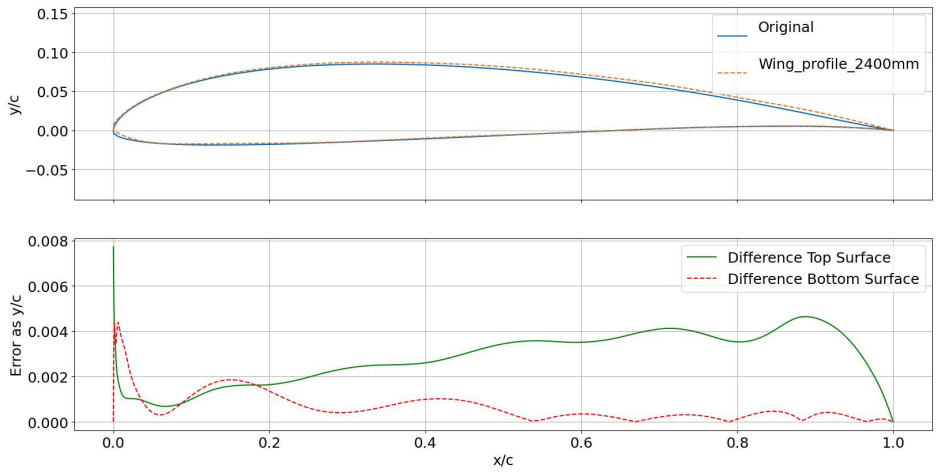


Figure 5.10: Total airfoil difference between the section at $2400\ mm$ and the original airfoil

Comparison by polars

An airflow simulation using XFLR5 (see section 3.6) was executed to compute polars of both the generated and the original airfoil. In the process, the extracted foils converged most of the time even for a high AoA. An analysis with an AoA from -4 to 20 degrees was performed with different exemplary Reynolds numbers. The computation mostly converged over the complete AoA range. However, the resulting polars are always less than those of the original airfoil, but they share the same trend (see Fig. 5.11 and 5.12). The maximum of the $c_{l,\alpha}$ curves are also at the same AoA. It is interesting to see that even though the airfoil section at $2400\ mm$ had a lower total difference to the original airfoil, the results gained with the section at $220\ mm$ are closer to the polars of the original foil.

Overall, the polars with a low Reynolds number ($\sim 3 \cdot 10^4$) are really close to the original polars. As can be seen in figure 5.11 and 5.12, the lowest polars differ only slightly with a maximum deviation of less than 0.1. The polars of middle range Reynolds numbers from 10^5 to 10^6 differ significantly from the original, but in the high Reynolds area this difference decreases again.

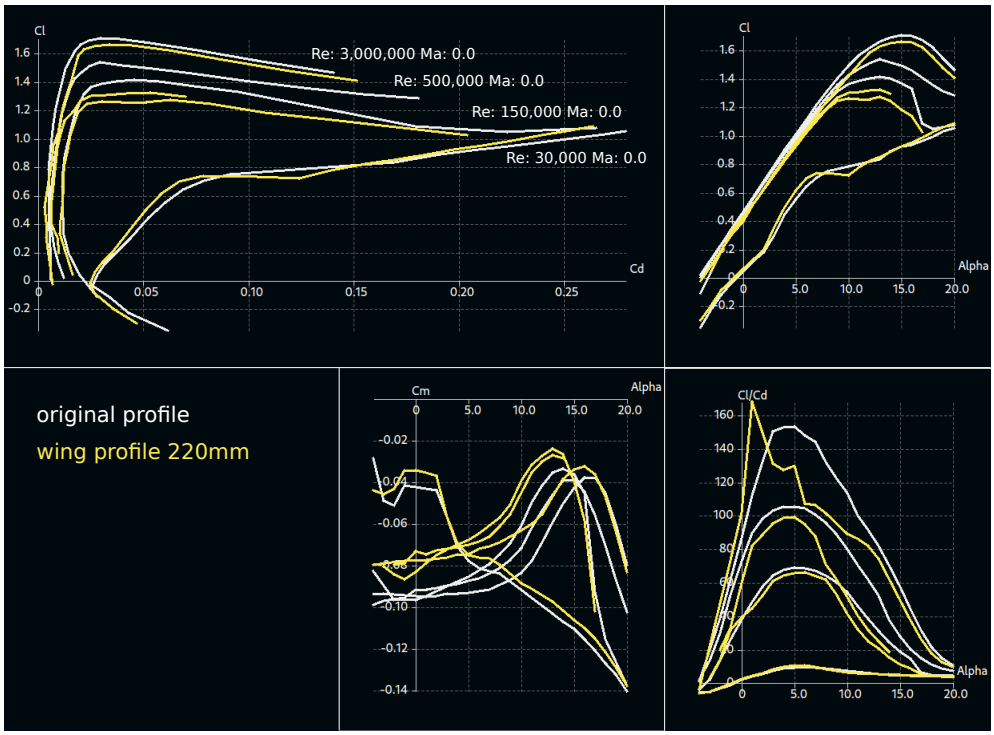


Figure 5.11: Computed polars of the original airfoil (white) and the airfoil at $220\ mm$ (yellow)

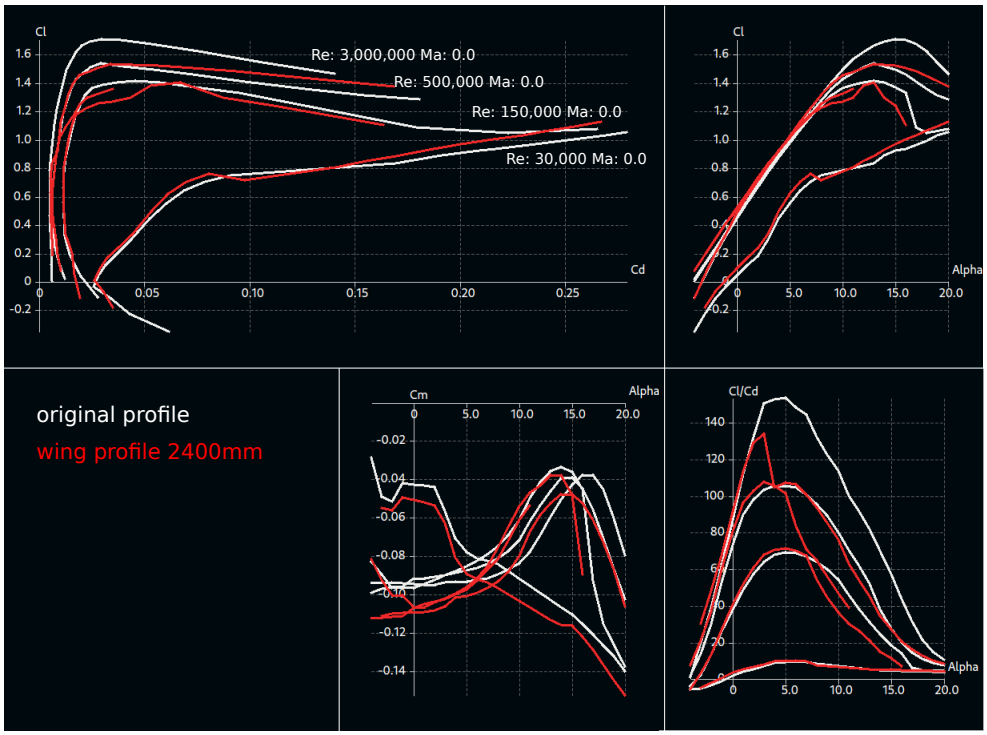


Figure 5.12: Computed polars of the original airfoil (white) and the airfoil at $2400\ mm$ (red)

5.2.2 Airfoil Results of the V-tail of IMPULLS

For the tail only one section was selected for the comparison. The toolchain has still some areas for improvement which are specified in chapter 6. Until now, it was therefore not possible to extract a converging airfoil without a flap derotation. The selected airfoil is also compared in terms of curvature, total difference and polars.

Comparison by curvature

In figure 5.13 the curvature is compared. The orange and light blue lines correspond to the original airfoil. It can be seen that the curvature of the extracted airfoil is almost continuous, in contrast to the original foil. The continuous airfoil is the one which generates a better airflow.

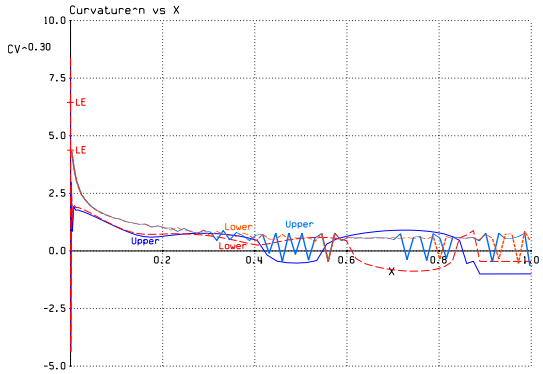


Figure 5.13: Curvature comparison between the V-tail section at $260\ mm$ and the original airfoil

Comparison by total difference

The comparison of the total difference of the extracted to the original airfoil shows that the flap was not derotated completely (see Fig. 5.14). The airfoil with the derotated flap of the wing of IMPULLS also shows a deviation in direction of the trailing edge. It can be assumed that the flap was not completely derotated. The average error of the V-tail amounts $8.15 \cdot 10^{-3}$.

Comparison by polars

Even if the tail extraction is worse than the one for the wing, the polars again have the same tendency as the original foil. At low Reynolds numbers such as $3 \cdot 10^3$ the $c_l - c_d$-curve completely coincides to the one of the original foil. However, the extracted airfoil only converges to significant lower AoAs and at lower Reynolds numbers when compared to the wing airfoil (see Fig. 5.15).

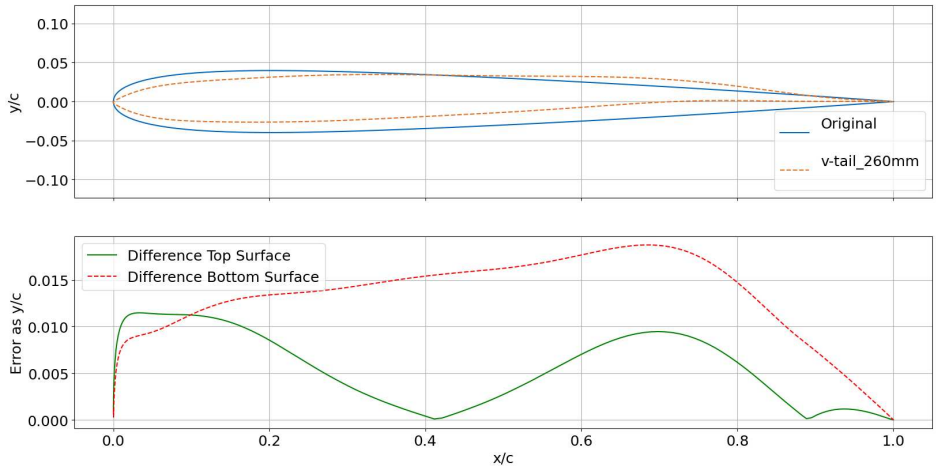


Figure 5.14: Comparison by total difference between the V-tail section at $260\ mm$ and the original foil

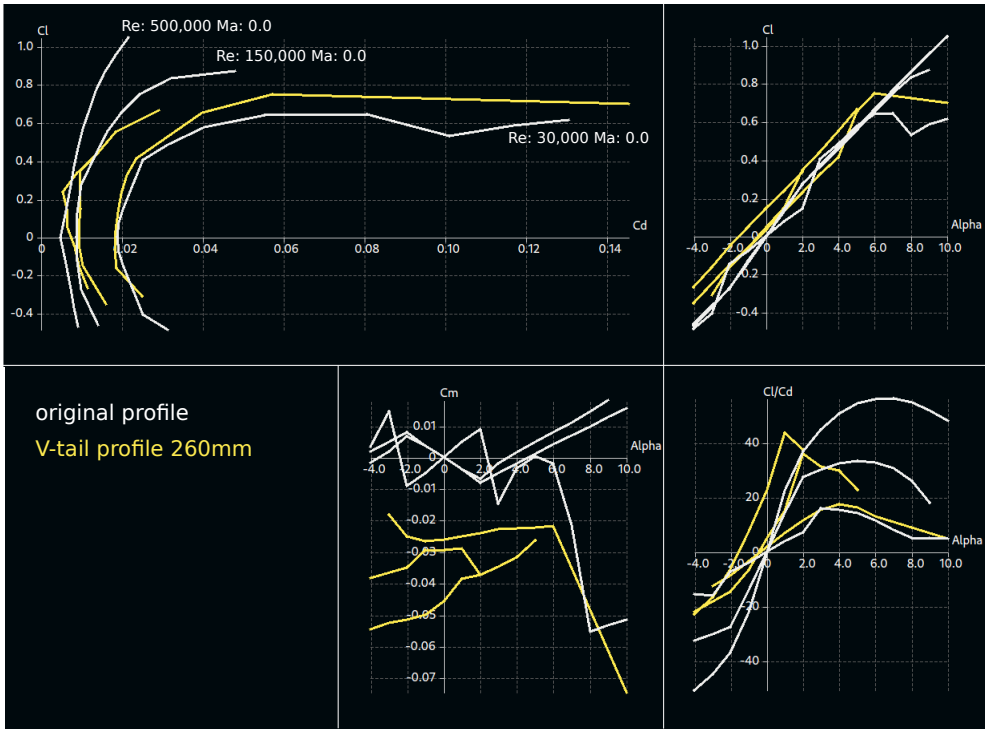


Figure 5.15: Comparison of the polars of the V-tail airfoil at $260\ mm$ (yellow) and the original airfoil (white)

5.2.3 Result of the DG-800 S

As mentioned in chapter 1, for the DG-800 S, the airfoil geometry is unknown. Thus, the extracted airfoils cannot be compared to the original profiles. Selected foils of the wing, horizontal and vertical tail are evaluated instead in terms of curvature and polars. The wing can also be compared to an airfoil extracted by Çetin (Çetin, 2020). The extracted airfoils are presented in figure 5.16.

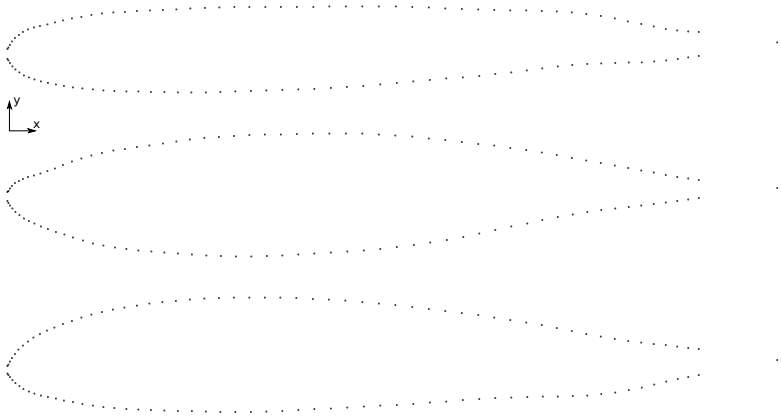


Figure 5.16: Exemplary extracted Airfoils of the horizontal tail (top), vertical tail (mid) and the wing (bottom) of the DG-800 S

Curvature of the extracted foils

Looking at the curvature of the horizontal tail and wing, it can be seen that the lower curve has many inflection points and oscillates, in contrast to the upper curve. This could point to a polynomial degree that is too high (see Fig. 5.17). A distinction between two different polynomial degrees is computed during the smoothing process to optimize the fitting. To achieve a better polynomial fit it could be convenient to fit the polynomial with more than only two different degrees. This is also verified by comparing the curvature of the vertical tail to the curvature of the wing and horizontal tail, where the lower curve barely oscillates.

Polars of the extracted foils

Figure 5.18 shows the computed polars of the extracted airfoils. As can be seen, the polars converge to a high AoA, similar to the wing airfoil of IMPULLS. The computation was executed for an AoA ranging from 4 to 20 degrees.

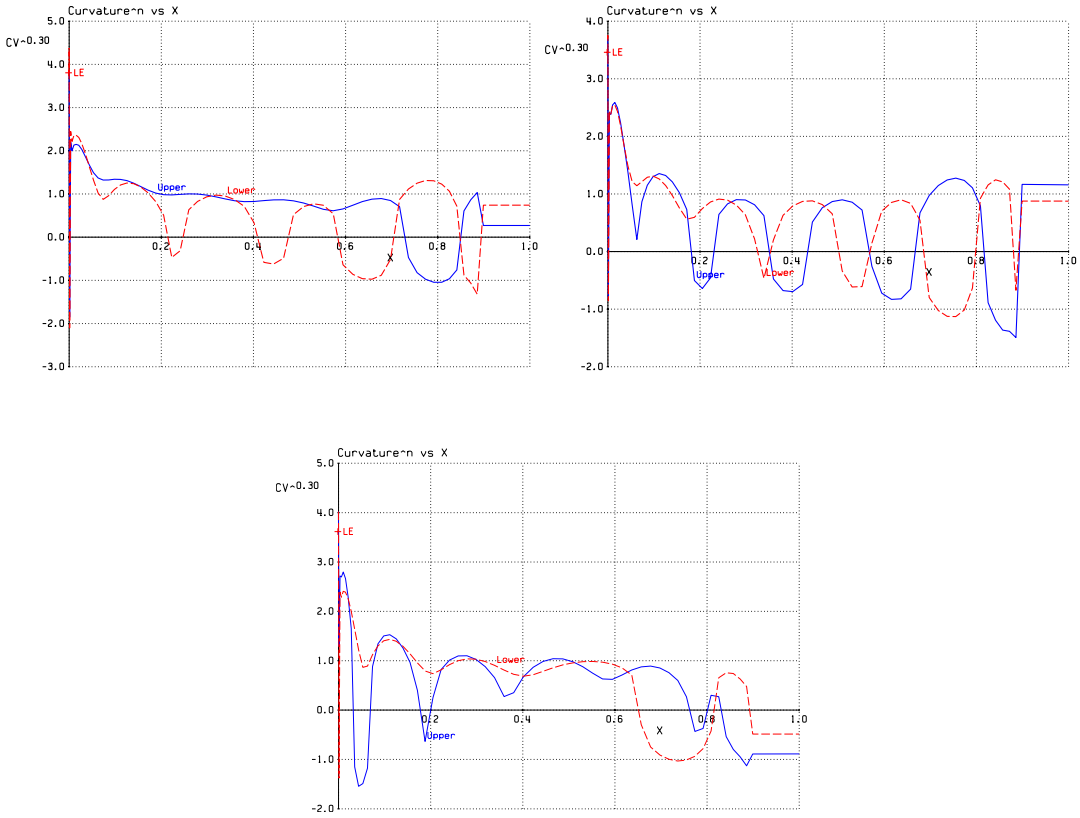


Figure 5.17: Curvatures of the extracted airfoils of the wing (top left), the horizontal tail (top left) and the vertical tail (bottom) of the DG-800 S

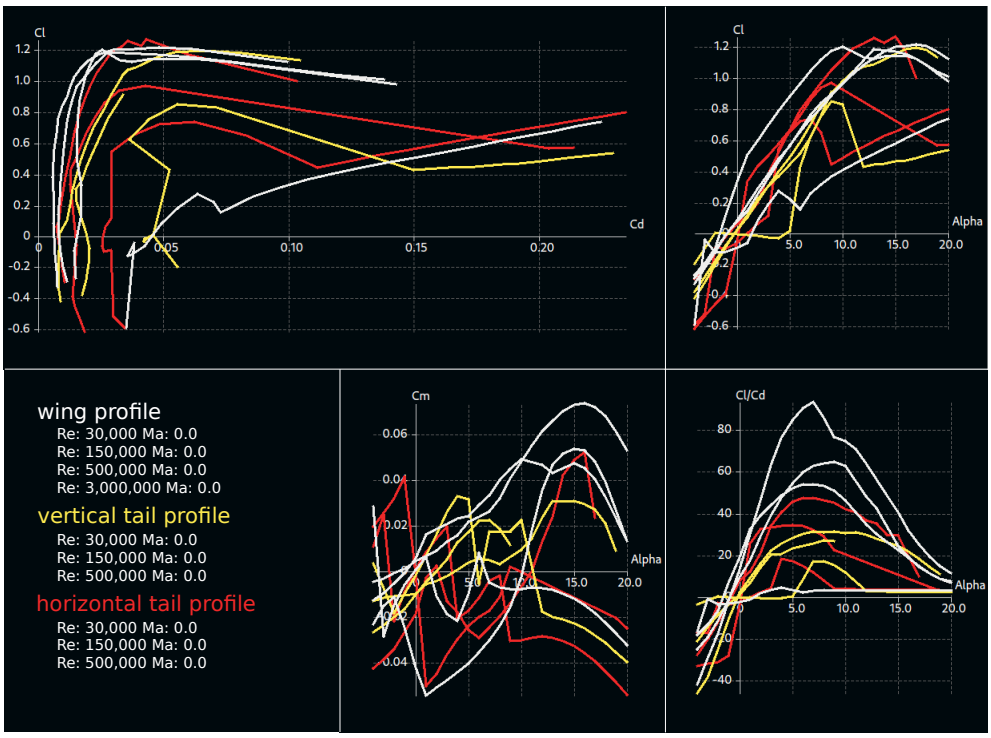


Figure 5.18: Polars of the the extracted foils of the wing (white), the horizontal tail (yellow) and the vertical tail (red)

Comparison to the predecessor toolchain by total difference

The extracted foil was compared to an airfoil extracted by Çetin (see Fig. 5.19). The outlier at the lower part of the airfoil was removed by the smoothing process. At the trailing edge, that the flap was not completely derotated as mentioned in section 5.2.2. If the flap is aligned, it is possible that the two foils match well in direction of the trailing edge.

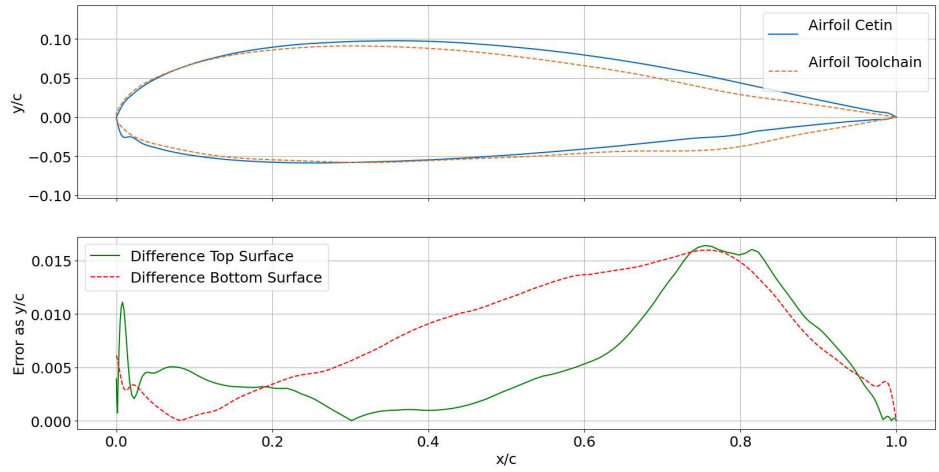


Figure 5.19: Total difference between an airfoil of the DG-800 S extracted by Çetin (Çetin, 2020) and the current toolchain with derotated flap

5.3 Case Study about fitting functions

Especially the UAV IMPULLS was used for a case study, where the goal was to find how the scan data can be improved. The reason for that is the known geometry of the UAV. The used airfoils were available for this thesis and could therefore be compared to the airfoil extracted by the toolchain. It was tested how the results are influenced by different fitting functions (see section 5.3.1) and what impact different fitting degrees have on the results (see section 5.3.2). For this task the curvature, absolute difference and polars were compared.

5.3.1 Case Study of different fitting functions

First, different fitting functions were examined. The original scan data is compared to a spline fit with Steffen's method (see section 4.4.5), the polynomial fit used in this thesis and a mixture of spline and polynomial fit. In the hybrid, the target was to prevent significant deviations of the original scan data but still accomplish a smoothing of the airfoil. The airfoil section at $220\,mm$ of the wing of IMPULLS was used to compare the extracted foils to the original airfoil. As in section 5.2.1, the foils are compared in terms of curvature, total difference and polars. The results for the polynomial fit can be found in the figures 5.8 on the left, 5.9 and 5.11.

Comparison by curvature

Figure 5.20 shows the different curvature plots. The editing of the scan data improves the curvature. Also the spline fit leads to a smaller oscillation even though the GSL algorithm is an interpolation method. The combination of spline and polynomial fit improves the result significantly. Nevertheless, the curvature of the polynomial is by far the best result in terms of the curvature (see Fig. 5.8).

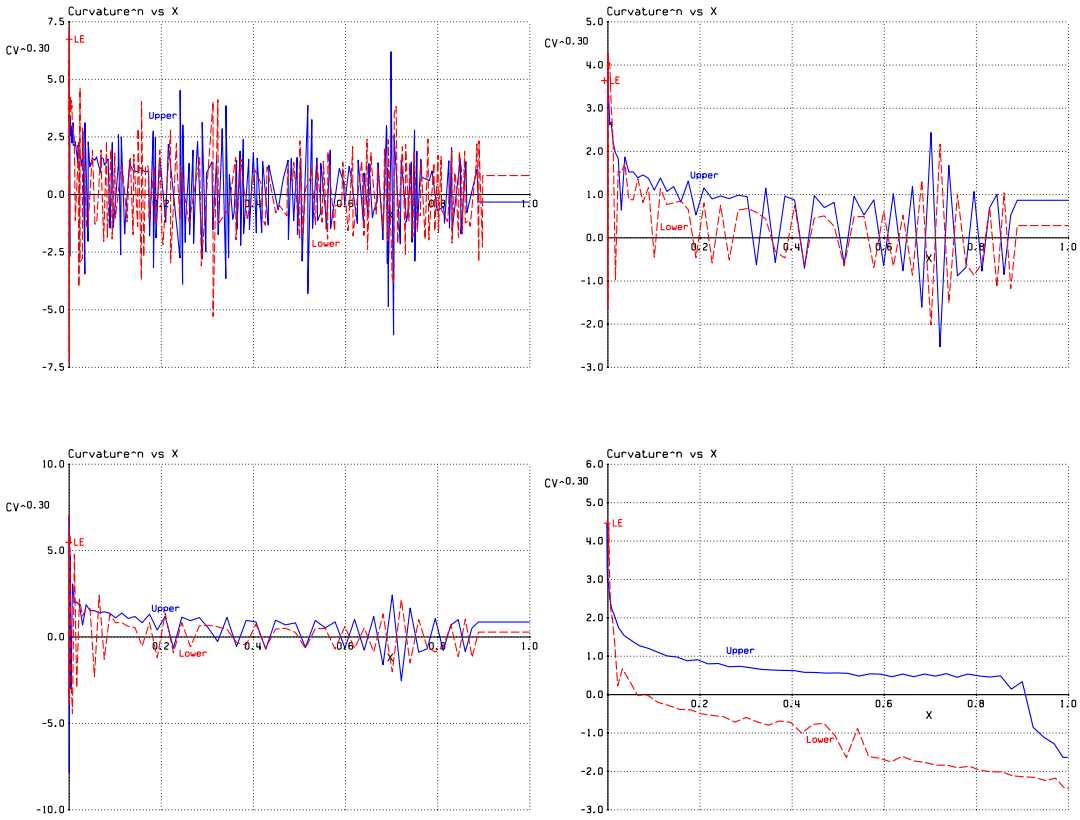


Figure 5.20: Curvatures of the original scan data (top left), the spline interpolation (top right) and the combination of spline and polynomial (bottom left) and for comparison the curvature of the original airfoil (bottom right)

Comparison by total difference

The comparison by total difference to the original foil shows that the polynomial fit matches best. The average error could be decreased from $4.6 \cdot 10^3$ for the scan, to $3.99 \cdot 10^3$ for the spline, to $3.97 \cdot 10^3$ for the hybrid to $2.6 \cdot 10^3$ for the polynomial. Thus, the smoothing of the foil using the polynomial approach is suited best for approximating the original airfoil. In figure 5.21 the total difference is plotted. It can also be seen that the spline fit and the hybrid provide a smoothing of irregularities and decrease the error of the top surface. However, only the

polynomial fit achieves a significant error reduction for both the bottom and the top surface (see Fig. 5.9). It can be assumed that an improvement of the scan data would also decrease the error of the extracted airfoil.

Comparison by polars

The polars of the hybrid and the original scan data reach the lowest lift coefficient from all fitting methods. The computed coefficients of the spline and the polynomial approximate the original airfoil significantly better. The highest c_l - value was computed by the airfoil fitted to the spline. However, this value is above the maximum value of the original foil. In addition, the maximum of the $c_l - \alpha$ curve of the spline is shifted about several degrees of the AoA. In contrast to this, the maximum of the polynomial fit is almost at the same AoA as the one of the original airfoil (see Fig. 5.22). Overall, it should also be mentioned that the polar of the low Reynolds numbers is represented very well by every fitting method and even the non-edited scan data matches almost perfectly.

Conclusion

The polynomial fitting method selected in this thesis approximates the original airfoil best when compared to the other fitting methods in case of the curvature and the total difference to the original airfoil. In terms of the comparison of the polars, it becomes clear that the fitting process could be improved in the future because it does not generate the lift coefficient of the designed profile. Nevertheless, the same behavior and tendency as the original airfoil can be observed.

5.3.2 Case Study of different polynomial degrees

Since the fitting via polynomial is a promising approach to smooth the airfoil (see section 5.3.1), a second study was executed. In this study, the goal was to find the optimal degree for the polynomial fit. High order polynomials oscillate easier and low order polynomial can only map the curve with significant deviation. So, a middle ground between these two extremes had to be found. As mentioned in section 4.4.5, two different orders were selected for the fitting process. The first, a polynomial of a low order for curves with only a low maximum Y-distance and one of higher order for curves with a high maximum Y-distance. This means that for a non-symmetric airfoil the upper part is fitted using a high order polynomial and the lower part using a low order polynomial.

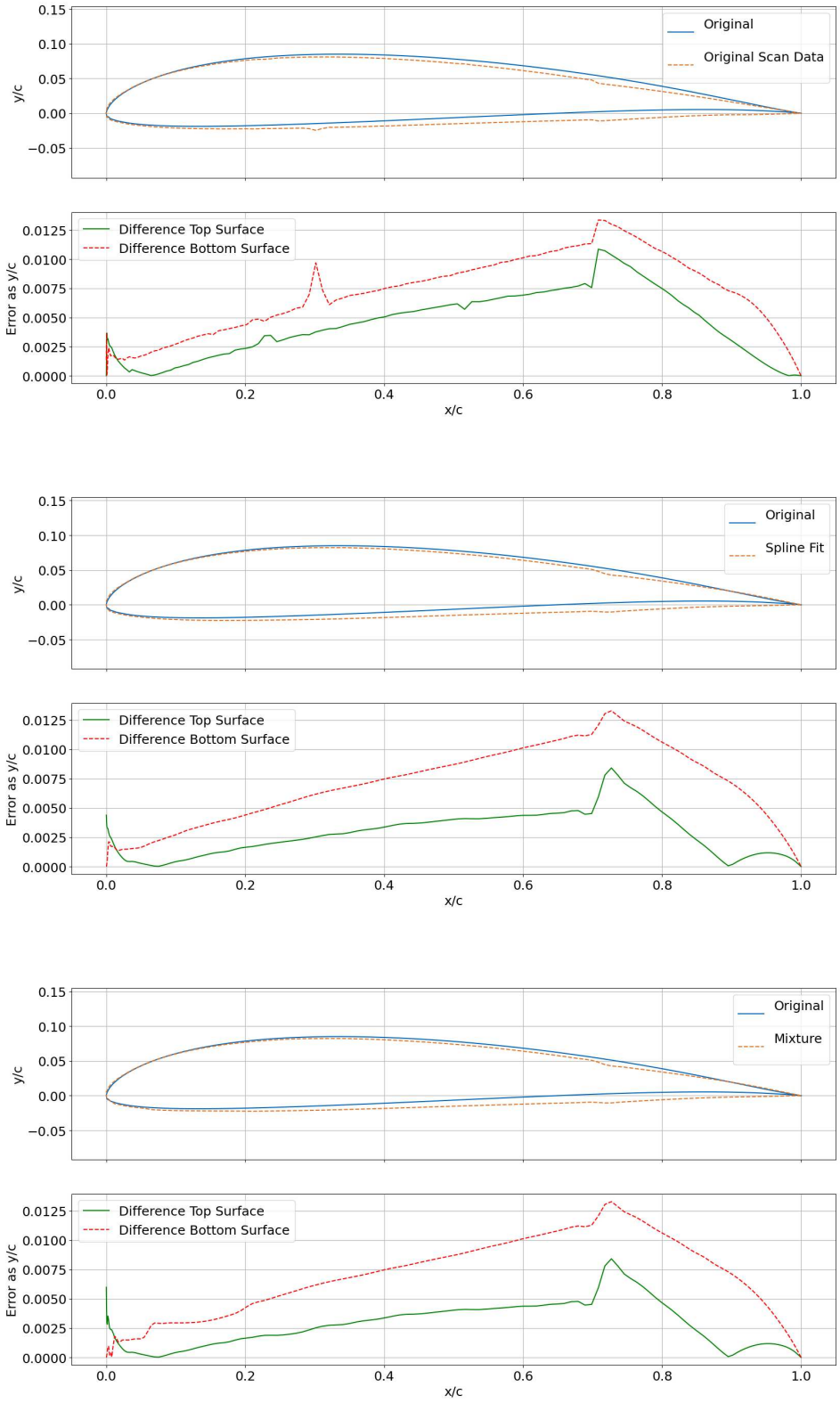


Figure 5.21: Comparison of the original scan data (top), the spline interpolation (mid) and the combination of spline and polynomial (bottom) to the original foil

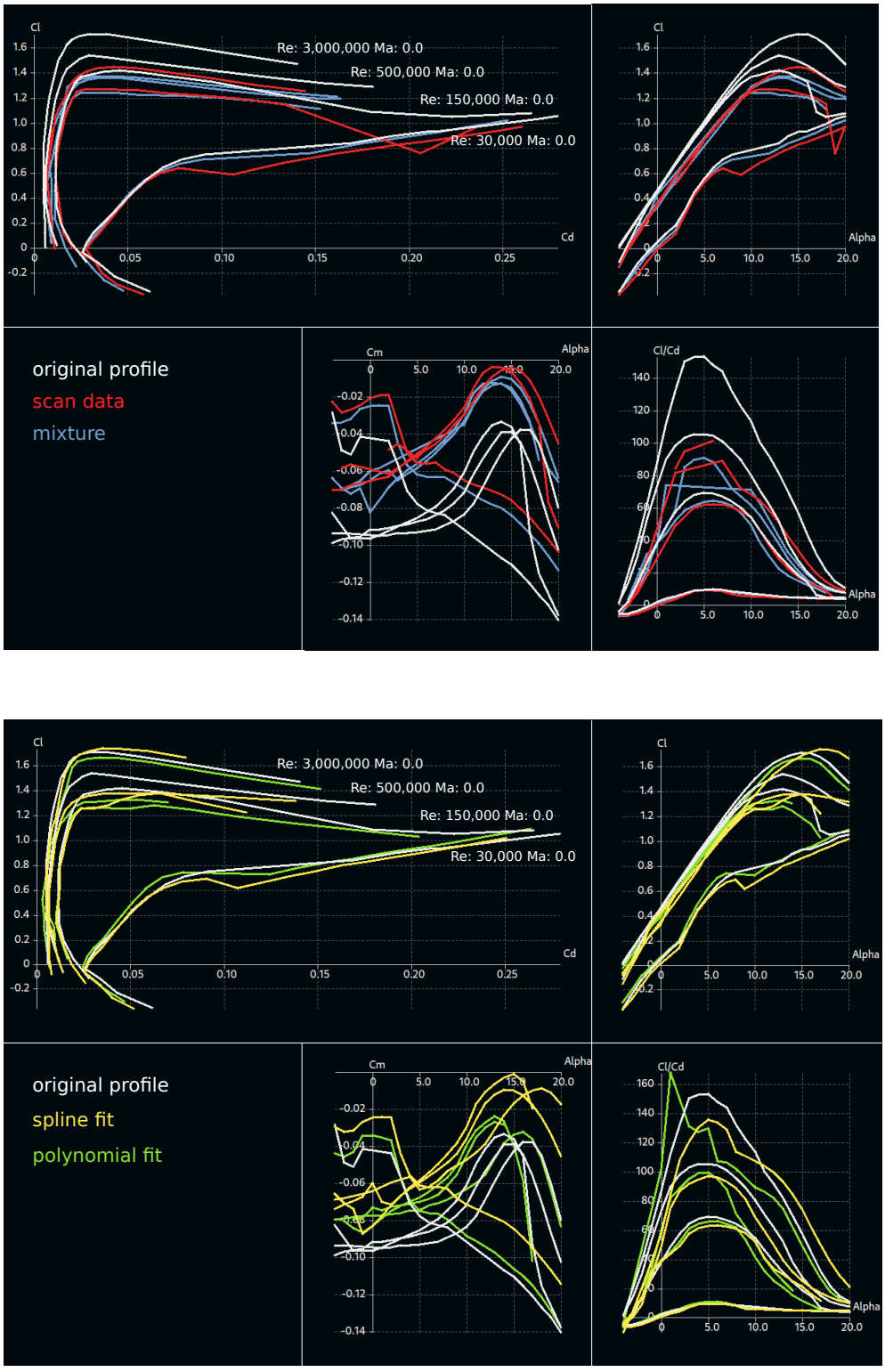


Figure 5.22: Comparison of the computed polars of the scan data (red) and mixture (light blue) in the top and the spline (yellow) and the polynomial fit (green) in the bottom with the original airfoil (white)

In the study, the impact of the top and the bottom order was examined separately and the three best results were chosen. Then, each degree used for the bottom polynomial was combined with each degree used for the top polynomial and polars for the foils were computed. In the end, the two degrees with the best combined result were selected. For the data used in this thesis, these were degree 16 for the top and degree 10 for the bottom of the airfoil.

Polars with different top degrees

Exemplary polars with different top degrees are presented in figure 5.23. The polars of every fitting degree represent the polar of the original airfoil in the low Reynolds number range with no significant deviation. The lowest degree presented in figure 5.23 is 11. It has a significantly lower lift coefficient when compared to the original foil. Nevertheless, the same tendency as the original foil can be observed. In case of the highest order polynomial with a degree of 25 the c_{l_α}-value is the same as at the original airfoil. However, it does not converge for high AoA. The evaluation of the top degree found that a polynomial fitting degree ranging from 15 to 17 provides the best results for the lift-coefficient.

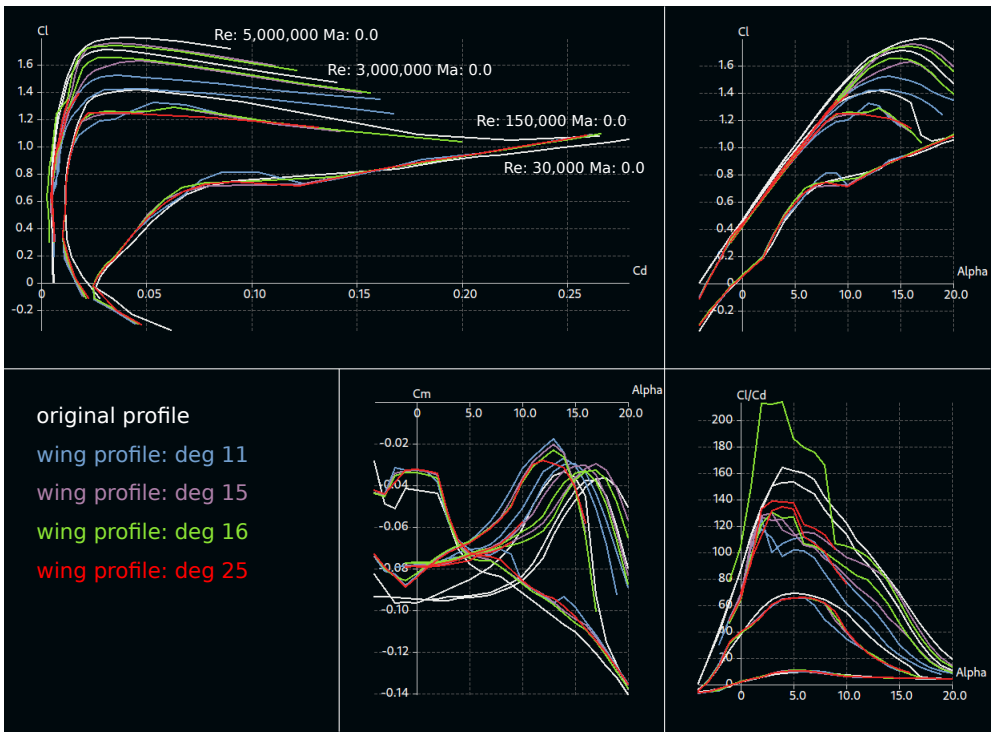


Figure 5.23: Comparison of the polars of different fitting degrees for the top of the airfoil

Polars with different bottom degrees

The degree of the bottom polynomial also has barely an impact on the polars of low Reynolds numbers. In case of the bottom degree, the degree has an inverse impact on the quality of the result. High-order polynomials result in significantly lower lift-coefficients while low-order polynomials tend to diverge (see Fig. 5.24). The best results could be reached with a degree between 9 and 11.

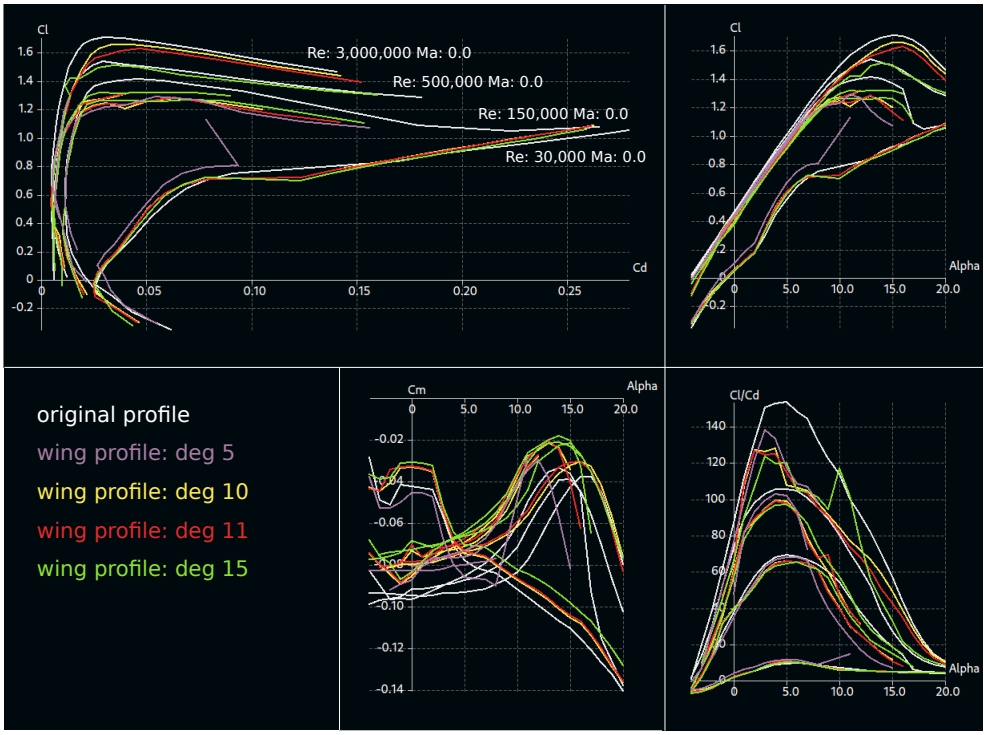


Figure 5.24: Comparison of the polars of different fitting degrees for the bottom of the airfoil

Polar comparison of the best top and bottom degrees

Nine foils were computed and evaluated during this study. In figure 5.25, three exemplary foils of this examination are presented. It was observed that the airfoil with the top degree of 16 and the bottom degree of 10 best approximates the original airfoil. In case of the others, the lift coefficient is either lower or the foil does not converge to high AoA.

5.4 Results of the extracted geometry

Apart from the airfoil, several TXT files were generated during the computation process. The position of the center, width and height of the fuselage were saved. In case of the lifting surface the chord length, dihedral, twist, offset and flap position are evaluated. In addition, the distance between two generated sections is saved as span partition. The toolchain has some weaknesses which are specified in section 6. These lead to sometimes unrealistic

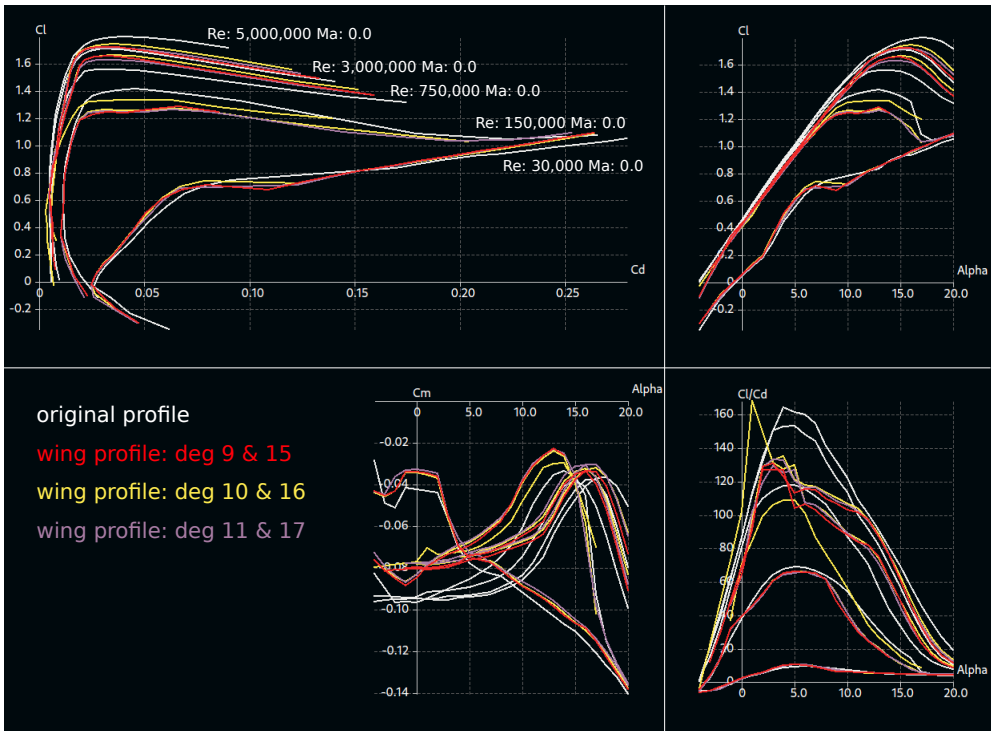


Figure 5.25: Comparison of the polars of different selected fitting degrees for the bottom and top of the airfoil

and defective results which are presented in this section. To generate the aircraft models in ADEBO (see section 5.5) and XFLR5 (see section 5.6) the faulty data had to be excluded. For the sake of academic transparency, the entire data is presented in this thesis. For presentation reasons the results are rounded to two decimals.

5.4.1 Results of IMPULLS

The results of the fuselage are presented in table 5.1 and the results of the lifting surfaces in table 5.2. Additionally, the position of the leading edge of the first section of the wing and tail is saved. For IMPULLS the wing's first section is at $(220, -481.9, 12.48)$ and for the V-tail at $(125, 1209.66, 130.68)$. These can be entered during the model generation to specify the position of the lifting surfaces.

5.4.2 Results of DG-800 S

The following tables present the data extracted from the scan of the DG-800 S (see tables 5.3 and 5.4). Just like IMPULLS, for the DG-800 S the position of the wing and tail was extracted. The first extracted leading of the wing is at $(120, 495.62, 20.42)$, for the horizontal tail $(70, -925.02, 222.55)$ and the vertical tail at $(0, -840.60, -4.15)$.

Section distance [mm]	Plane height z [mm]	Half-axis x [mm]	Half-axis z [mm]
0	133.55	60.00	-70.53
525	140.49	170.69	-177.04
940	159.62	168.37	-169.73
1355	172.21	136.26	-136.90
2050	165.95	75.7	-74.44
2625	175.12	78.589	-85.23
2935	198.87	81.21	-75.65

Table 5.1: Geometry results of the fuselage of IMPULLS

Section name	Chord length [mm]	Dihedral [°]	Twist [°]	Offset [mm]	Flap position [%]	Span partition [mm]
wing profile 220mm	363.25	0.00	7.21	0	70.78	220
wing profile 575mm	345.37	0.06	7.69	1.23	74.71	355
wing profile 1150mm	332.92	1.08	7.52	8.91	70.56	575.10
wing profile 1180mm	335.04	6.11	8.04	9.56	70.1	30.17
wing profile 1770mm	283.42	5.86	0.47	30.62	84.7	593.09
wing profile 2350mm	185.74	5.77	4.61	92.15	70.1	582.95
wing profile 2400mm	168.02	5.47	5.78	108.58	0	50.23
wing profile 2450mm	143.31	4.9	5.66	136.56	0	50.18
v-tail profile 125mm	230.65	46.07	0.5	0	71.6	180.17
v-tail profile 260mm	216.00	46.31	0.85	15.7	75.69	195.42
v-tail profile 300mm	208.70	46.43	1.45	23.58	88.95	58.04
v-tail profile 470mm	149.95	50.02	3.42	229.08	17.82	264.57
v-tail profile 505mm	130.84	51.99	0.94	241.8	10.67	56.83

Table 5.2: Geometry results of the lifting surfaces of IMPULLS

Section distance [mm]	Plane height z [mm]	Half-axis x [mm]	Half-axis z [mm]
0	5.4	14.93	13.14
160	5.35	66.57	81.7
340	16.05	98.02	121.75
640	27.04	104.85	134.41
960	1.06	94.92	132.93
1090	69.68	55.29	60.14
2040	77.57	26.41	139.36

Table 5.3: Geometry results of the fuselage of the DG-800 S

5.5 ADEBO Models

The extracted data was entered in ADEBO to generate a model of the complete aircraft. The aircraft models are created with one airfoil selected for each lifting surface. No further computations or tests were executed. The dynamic behavior was examined using XFLR5 (see section 5.6) instead. ADEBO was only used for a geometric study.

Section name	Chord length [mm]	Dihedral [°]	Twist [°]	Offset [mm]	Flap position [%]	Span partition [mm]
h. tail profile 70mm	174.46	1.39	5.13	0	21.99	70.02
h. tail profile 150mm	157.43	1.34	4.93	8.20	20.2	80.02
h. tail profile 300mm	128.54	1.44	5.12	23.43	18.77	150.05
h. tail profile 410mm	97.79	0.73	2.93	44.61	18.67	110.01
wing profile 120mm	297.72	2.89	6.58	0	0	120.15
wing profile 135mm	303.39	1.88	4.25	0.14	15.53	15.01
wing profile 700mm	270.99	3.05	5.80	3.5	19.36	565.8
wing profile 1450mm	238.54	3.14	5.96	7.76	17.82	751.13
wing profile 2000mm	195.23	3.23	2.76	31.75	0	550.88
wing profile 2875mm	105.75	2.00	8.34	72.79	62.74	875.54
wing profile 2915mm	106.06	8.72	7.19	80.67	0	40.472
wing profile 2975mm	89.19	69.10	13.29	201.29	92.00	168.21
v. tail profile -200mm	270.22	0	3.31	0	35.69	200
v. tail profile -135mm	285.21	0	2.19	8.29	32.5	65
v. tail profile -90mm	269.26	0	4.23	8.28	15.57	45
v. tail profile 75mm	230.45	0	0.44	64.76	28.79	165
v. tail profile 170mm	213.12	0	0.88	96.59	31.42	95

Table 5.4: Geometry results of the lifting surfaces of the DG-800 S

5.5.1 ADEBO model of IMPULLS

For the ADEBO model the wing section at $220\ mm$ and the V-tail section at $260\ mm$ were chosen. These airfoils are presented in section 5.2.1 and 5.2.2. The fuselage of the UAV was sectioned seven times, the wing eight times and the V-tail five times. The distances of the generated sections and all extracted data which was included in ADEBO can be seen in tables 5.2 and 5.1. A plot of the aircraft model of IMPULLS is presented in figure 5.26.

5.5.2 ADEBO model of the DG-800 S

For the DG-800 S, the airfoils presented in figure 5.16 were selected for the model generation. The fuselage was sectioned seven times and the extracted data can be seen in table 5.3. The wing was sectioned eight times, the horizontal tail four times and the vertical tail five times. The corresponding geometries can be found in table 5.4. The ADEBO model is pictured in figure 5.27.

Through a comparison of the ADEBO model presented in figure 5.27 to the model created by the predecessor toolchain (see Fig. 2.6), it can be seen that the result was much improved on during this thesis. More sections were generated which approximate every lifting surface of the UAV. Also the bending at the tip of the wing was extracted. However, the partition between the last two sections is still faulty and should be fixed in the future.

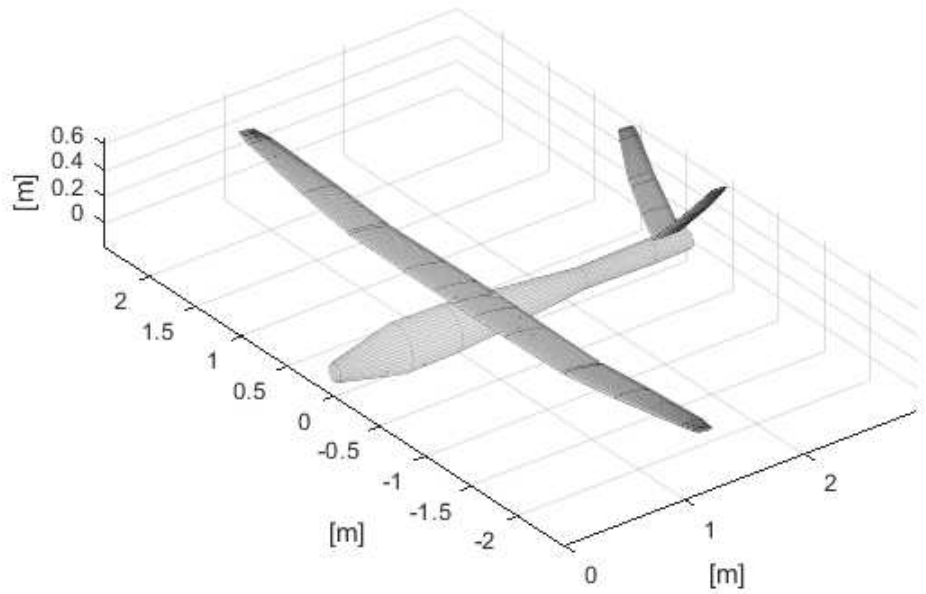


Figure 5.26: ADEBO model of IMPULLS

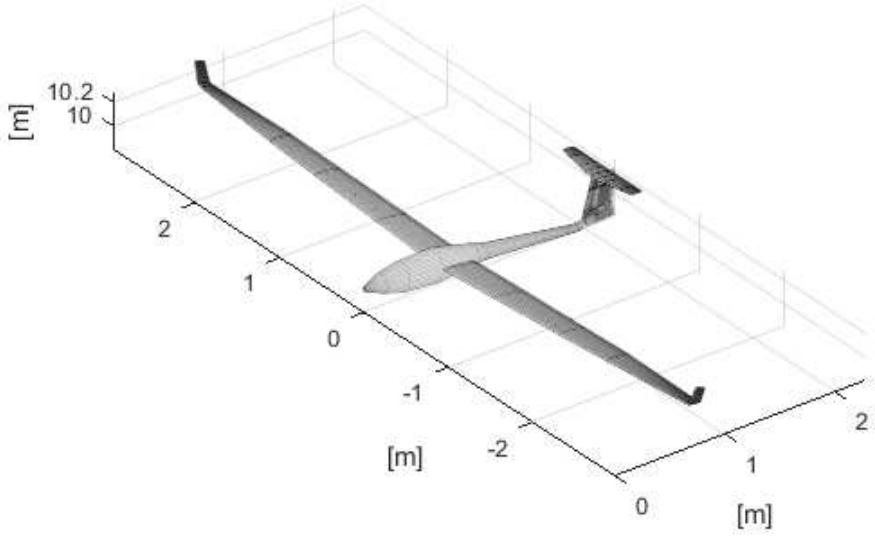


Figure 5.27: ADEBO model of the DG-800 S

5.6 XFLR5 Models

For dynamic evaluation, models of both UAVs were generated in XFLR5 (see section 3.6). XFLR5 is able to compute polars not only for the airfoil but also for the complete lifting surface. Unfortunately, it was impossible to compute the $C_{l,max}$ for either UAV. Thus, the AoA for a stall remains unknown. This could be improved in the future. Polars with an AoA from -10 to 2 degrees with regard to the axis of the fuselage were computed. Higher AoAs could not be interpolated because the lift coefficients of the airfoils were to low. With a better derotated flap it may be possible to compute higher c_l-values and therefore polars with higher AoA for the aircraft model.

5.6.1 XFLR5 model of IMPULLS

In figure 5.28 the generated XFLR5 model of IMPULLS is presented. Airflow simulations were computed using velocities ranging from $10\ m/s$ to $50\ m/s$, because the UAV was designed for a cruise speed of $19\ m/s$. The model has a wing area of $1.51\ m^2$ with a span of $5\ m$. The designed wing area amounts $1.548\ m^2$, the generated model distinguishes only 3% to this. Also, c_{l0} and c_{d0} can be read with the computed polars. The axis distances to the point where the $c_l - c_d$ curve has a vertical tangent are the coefficients. c_{l0} is about 0.3 in all polars. The drag coefficient depends on the velocity and is about 0.0118 in case of the cruising speed of $20\ m/s$ c_{d0} (see Fig. 5.29).

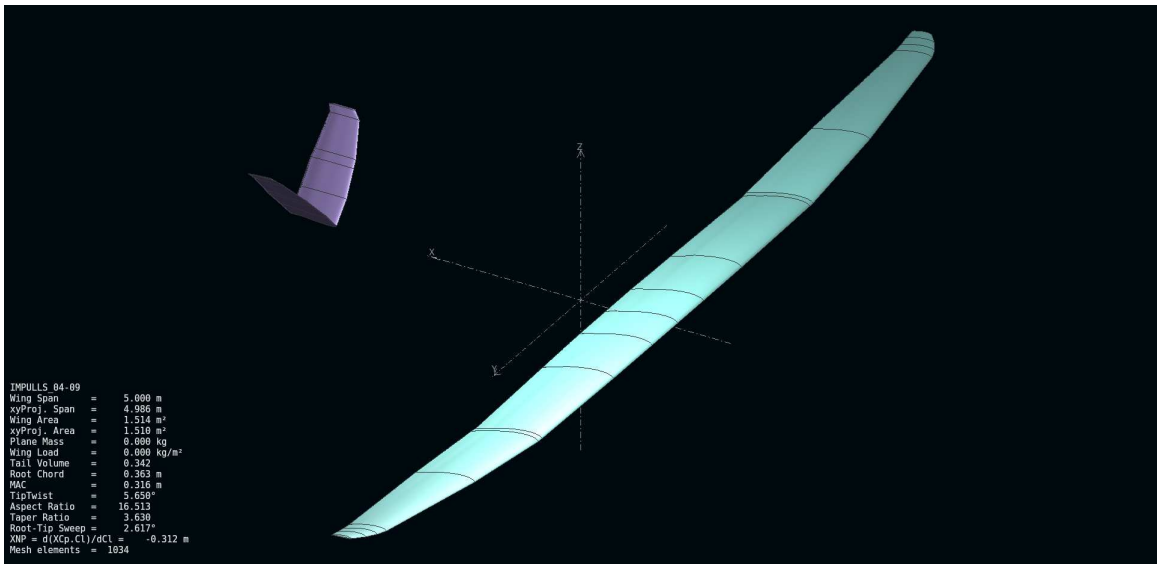


Figure 5.28: XFLR5 model of IMPULLS

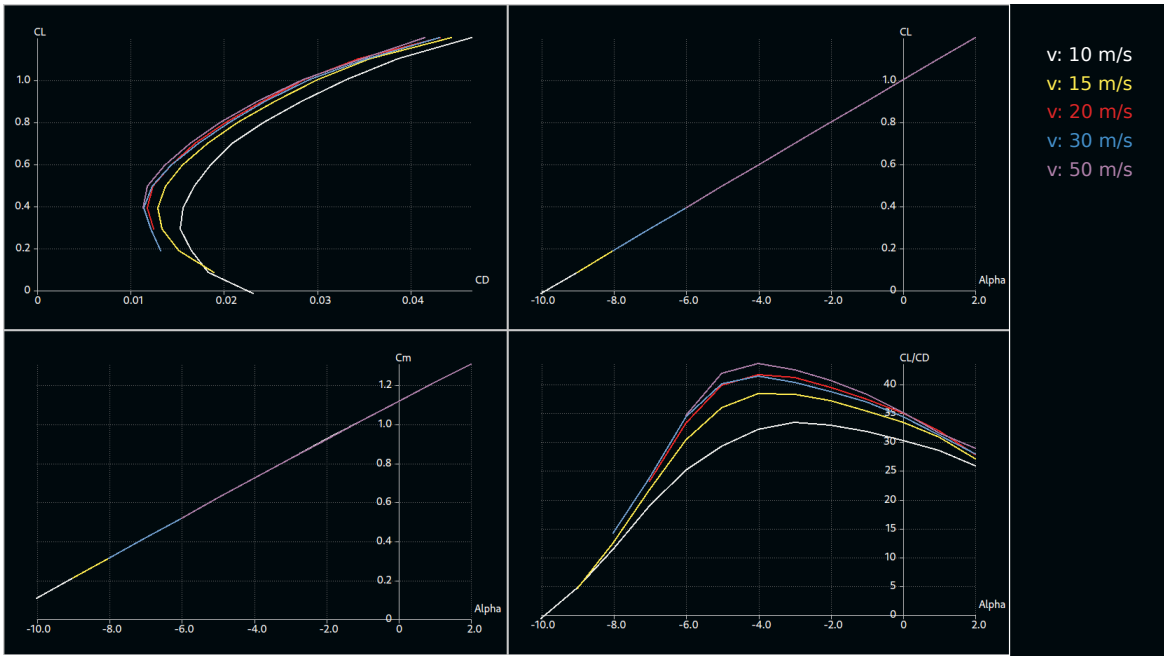
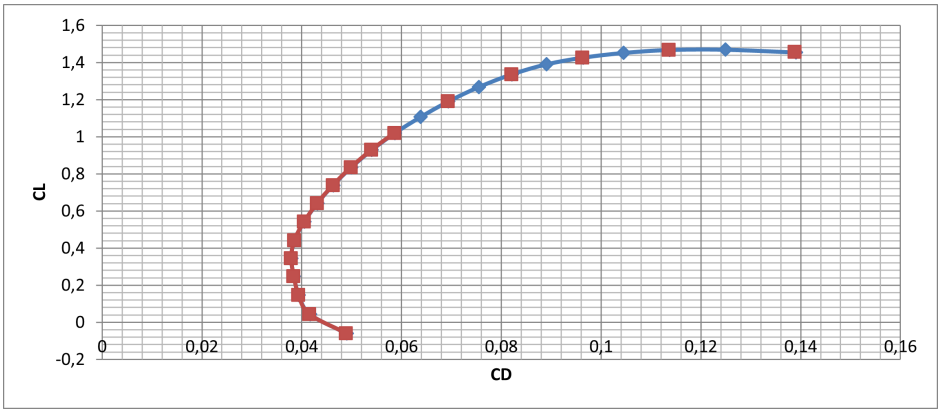


Figure 5.29: Polars of the XFLR5 model of IMPULLS

There are polars available from the designing process of IMPULLS (see Fig. 5.30). When compared to the polars of the created model in XFLR5, it can be seen that the $c_l - \alpha$ curve matches almost exactly the polar in figure 5.30. Also the c_{l0} is the same as designed. Only the c_{d0} coefficient does not match. It amounts to 0.039 which is two times the computed value. However, the polars were computed with a fuselage. The fuselage is the main factor for c_{d0}, so this value can not be compared with the XFLR5 model.



5.6.2 XFLR5 model of the DG-800 S

The DG-800 S was also modeled in XFLR5 (see Fig. 5.31). The wing area of the model is $1.309\ m^2$. Due to the large flight regime of the DG-800 S, airflow simulations using velocities ranging from $10\ m/s$ to $75\ m/s$ were conducted. The polars are presented in figure 5.32. c_{l0}

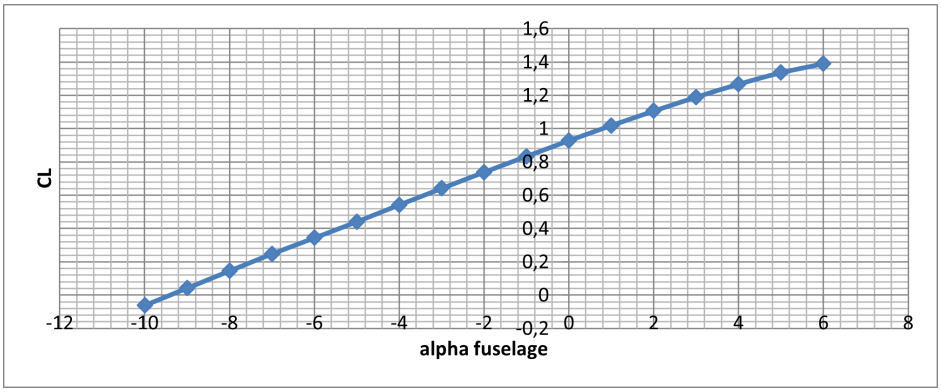


Figure 5.30: Polars of the designing process of IMPULLS

and c_{d0} can also be found here. The c_{l0} amounts to 0.2. For a velocity of $75\ m/s$ the drag coefficient is 0.01 (see Fig. 5.32). Also the $c_{l,\alpha}$ coefficient can be calculated with the polars. It is the gradient of the $c_l - \alpha$ curve and amounts to $0.10671/deg$. Converted to radian measure is this a value of $6.11/rad$. Typically, thin airfoils reach values of $c_{l,\alpha} \sim\ 2\pi/rad$ (Anderson, 2013, S.325). The computed $c_{l,\alpha}$ comes close to this value. There was no model of the DG-800 S available to compare the simulations to. However, flight tests were conducted and the c_{l0} and $c_{l,\alpha}$ computed (see equation 5.2). Unfortunately, these values do not match the computed data. Reasons for that could be a high measurement noise which is based on the measurement method and wind during the tests.

$$c_{l0} = 0.376$$

$$c_{l_\alpha} = 5.63 \tag{5.2}$$

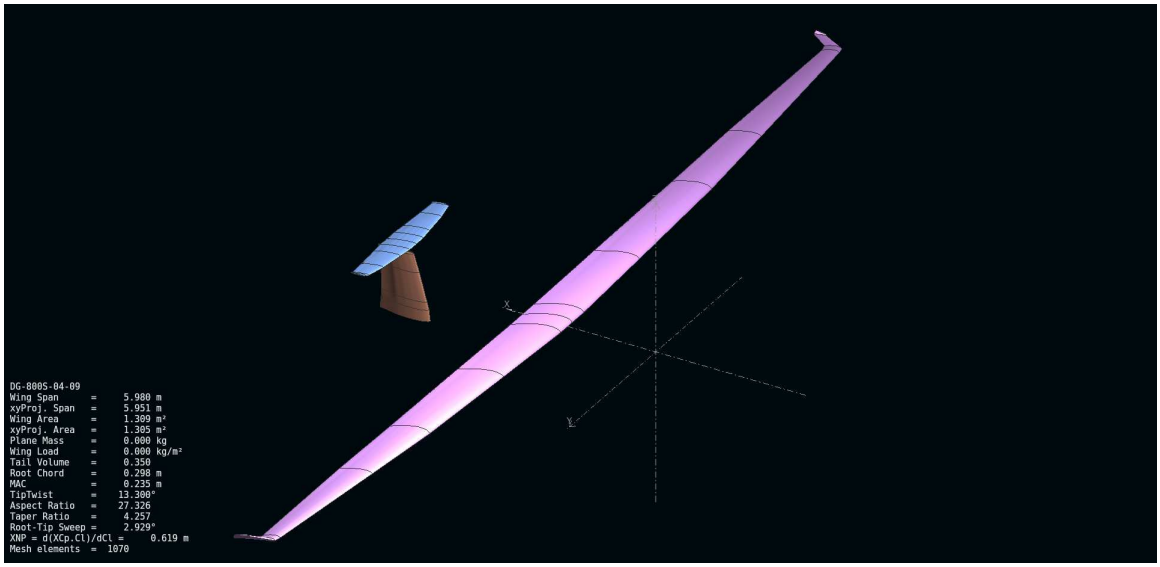


Figure 5.31: XFLR5 model of the DG-800 S

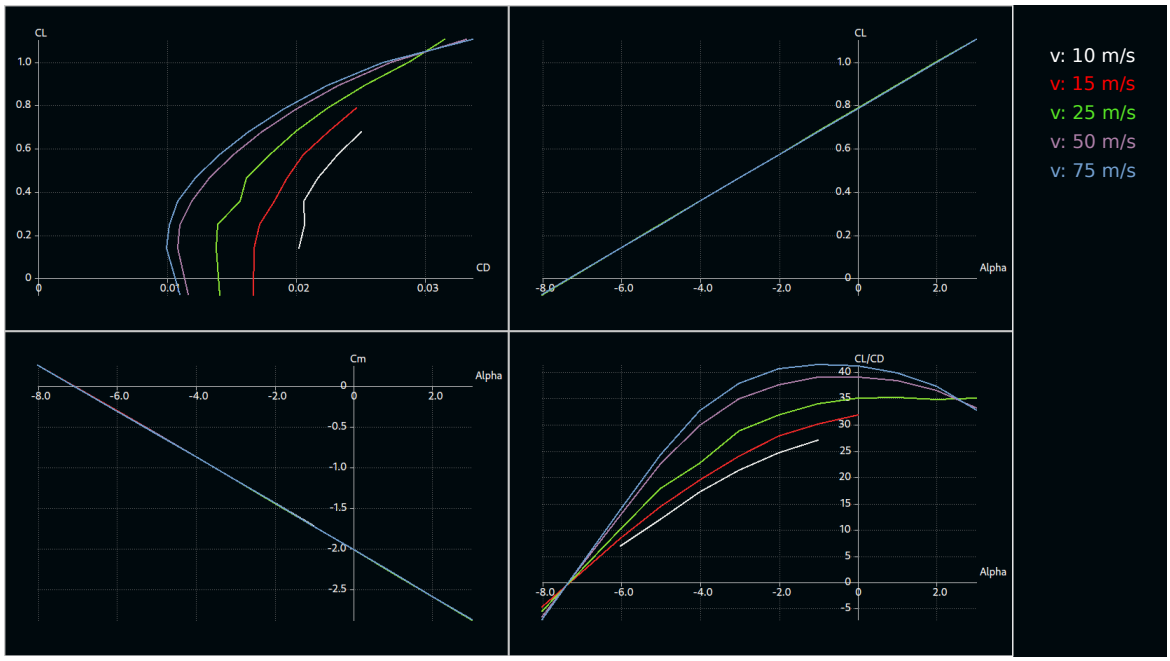


Figure 5.32: Polars of the XFLR5 model of the DG-800 S

6 Conclusion

In this thesis, the toolchain has been improved when compared to the predecessor toolchain created by Çetin (Çetin, 2020). The computation time was reduced from 30 to 45 minutes to 15 minutes maximum for any analysis with an point cloud with maximum 15 million data points. The numbers of sections has almost no impact on the process. The time consuming tasks are the aircraft alignment (see section 4.4.2) and the normal estimation (see section 4.4.3). This means that any desired number of sections can be generated in an appropriate amount of time. Additionally, the vertical and horizontal tail can be extracted using the toolchain. The ease of use was improved. Via a shell script, all required routines are called, the toolchain therefore is fully automated (see section 4.2). Sections can be set in arbitrary distances. Bends can be extracted and also entered in the aircraft model. The sections can be selected via a GUI which provides the possibility to obtain the whole geometry of the aircraft without knowing geometric aspects beforehand. The generated section file prevents invoking the GUI every time which decreases the operation time. Finally, the hinge line and the sweep of the wing and tail can be computed with the derotation method (see section 4.4.5).

In the result presentation, it is shown that the trend of the airfoils as well as the reconstructed aircraft is the same as of the original. Nevertheless, the current toolchain has some weaknesses which could be improved on in the future. As mentioned in section 5.2, the derotation of the flap is not completed so far, several degrees are still missing for a continuous surface. It probably suffices to increase the derotation angle with a defined value such that the curvature in direction of the tail is continued. Furthermore, the flap was sometimes not recognized. The faulty flap positions like 17.82% at section $470\ mm$ of the V-tail in table 5.2 show the airfoils where the flap was not derotated. The reason for this is the method for the trailing edge search. When the scan data has a bad resolution, the leading edge is mistakenly selected as trailing edge and the derotation fails. This faulty recognition occurred especially at the V-tail of IMPULLS and the wing of the DG-800 S. This problem could probably be solved by using a new scan of higher resolution. Generally, the results of the tail were significantly worse than the results of the wing (see section 5.2.2). During this thesis, the good extraction of the wing profiles was preferred. The flap recognition and rotation algorithm could be adapted in future work to improve the tail results.

Also the polynomial fit could be improved further. The curvature of the polynomial can strongly oscillate depending on the airfoil which was fitted (see Fig. 5.17). A distinction with four or five different degrees could generate even better results. Additionally, the polynomial fit oscillated for some airfoils (see Fig. 6.1). Due to the Chebyshev nodes distribution, the polynomial should be stable. However, this could also be because of a faulty spline interpolation beforehand. This problem occurred already during this thesis (see section 4.4.5) and was the reason for the selection of the cubic spline with Steffen's method (Steffen, 1990). However, it is possible that also the spline with Steffen's method oscillates.

Figure 6.1: Exemplary foil with oscillation of the top

In summary, the new toolchain enables the generation of a complete extraction of all geometries and airfoils within a few minutes. The results still contain some numerical errors. However, they are most of the time noticeable even to the untrained eye and can therefore be easily ignored.

7 Outlook

It was tested how the developed toolchain can be applied on different aircraft types. For this question, a jet was scanned and evaluated. A model of the lizard jet was chosen (see Fig. 7.1). The complete and post-processed 3D-scan of the jet is presented in figure 7.2. The flaps were not actuated during the scanning process. It could be tested if an extraction is generally possible, flaps could be extracted using a second scan.

Figure 7.1: Lizard jet

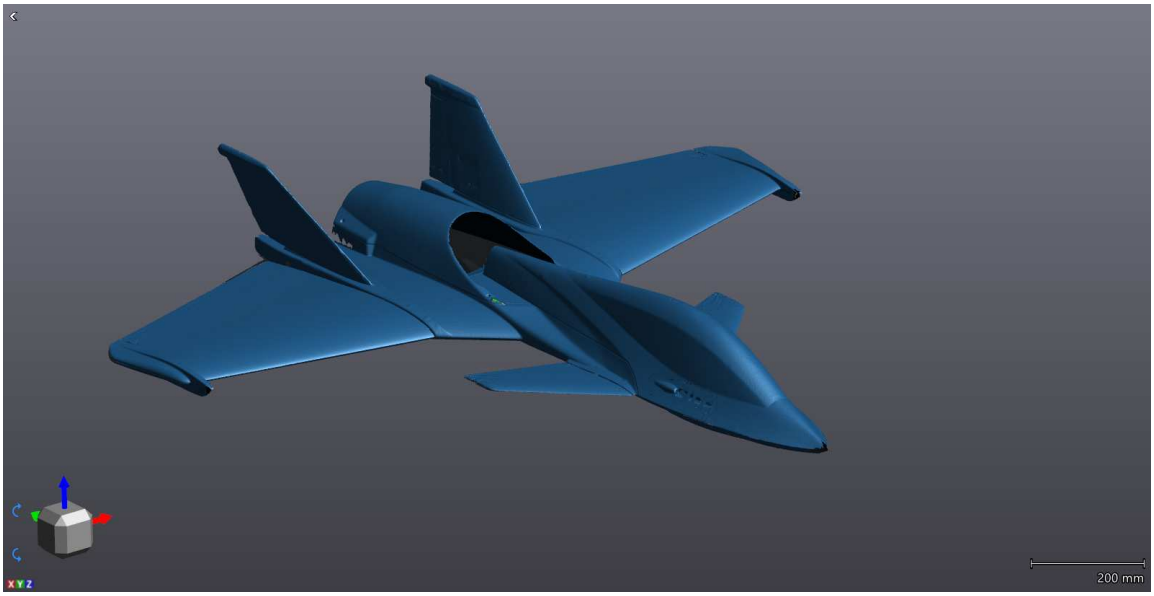


Figure 7.2: 3D-scan of the jet

Some problems occurred during the data extraction. First, the alignment of the UAV failed. The fuselage was aligned with the X-axis and not with the Y-axis as implemented. The reason for that is that the greatest edge of the bounding box is aligned with the X-axis. In case of

gliders or common aircraft the wings are significantly longer than the fuselage. Thus, the wings are aligned to the X-axis. A jet has however a long fuselage and in relation to this only small wings. So, for the extraction with the toolchain the already aligned aircraft had to be rotated about 90 degrees around the Z-axis to be able to process the jet.

Additionally, it was not possible to extract the vertical tail. First, the current toolchain expects a single fin and not a double one. To extract a double fin the UAV has to be split in half. This is so far not implemented because a single fin would also be split in half. Second, the jet has canards. These are located in front of the wing and are interpreted as the horizontal tail in the toolchain. The vertical tail is expected to be in the same area as the horizontal tail. Since in this case the horizontal tail is the canard and the vertical tail is mounted on the wings, the toolchain can not find the vertical tail and this extraction fails.

Nevertheless, sections of the fuselage, wing and canards could be generated. Figure 7.3 presents an exemplary airfoil of the wing and canards and the sections of the fuselage. So, the wing and canards could be reconstructed. A model of the fuselage however would be difficult. As it can be seen in figure 7.3, the fuselage can not be approximated by an ellipsoid which is the preferred and most common method for a fuselage model. A fuselage reconstruction in a CAD software could however be possible. These software are often able to import data points of a TXT file as coordinates. Then, the separate sections could be connected to form a surface or even a volume model.

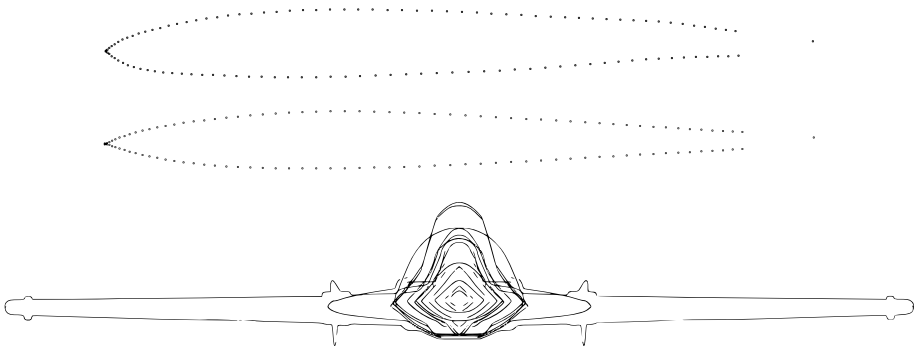


Figure 7.3: Generated sections of the wing (top), the canards (mid) and the fuselage (bottom) of the scanned jet

In summary, a complete extraction of a different aircraft type is still not possible. However, the toolchain is already capable to extract some data. For the remaining parts, the toolchain has to be adapted and be developed further. The possibility for a complete extraction of every aircraft type is real and could be realized in future work.

Bibliography

Anderson, J. D. (2013), *Fundamentals of Aerodynamics*, 5 edn, Mc Graw Hill India. ISBN: 978-0-07-070012-3.

Busch, S. (2020), Entwicklung einer Reverse-Engineering Toolchain für UAV Propeller, Bachelor's thesis, Institute of Aircraft Design - Technical University Munich.

CARF-Models (n.d.), 'DG-800S Overview'.
URL: *https://carf-models.com/de/products/dg-800-s* [Accessed on: 2020-10-04]

Creaform and AMATEK (2018*a*), *HandySCAN 3D Training Script*.
URL: *https://www.creaform3d.com* [Accessed on: 2020-10-04]

Creaform and AMATEK (2018*b*), *MaxSHOT 3D Training Script*.
URL: *https://www.creaform3d.com* [Accessed on: 2020-10-04]

Creaform and AMATEK (2018*c*), *VXmodel Training Script*.
URL: *https://www.creaform3d.com* [Accessed on: 2020-10-04]

Creaform and AMATEK (2019), 'VXElements'. Software. v7.03.
URL: *https://www.creaform3d.com* [Accessed on: 2020-10-04]

Dantsker, O. (2015), Determining Aerodynamic Characteristics of an Unmanned Aerial Vehicle using a 3D Scanning Technique. Proceedings in AIAA Aerospace Sciences Meeting 2015. doi: 10.2514/6.2015-0026.

Dantsker, O. and Vahora, M. (2018), Comparison of Aerodynamic Characterization Methods for Design of Unmanned Aerial Vehicles. Proceedings in AIAA Aerospace Sciences Meeting 2018. doi: 10.2514/6.2018-0272.

Deperrois, A. (2019), 'XFLR5'. Software. v6.47.
URL: *www.xflr5.tech/xflr5.htm* [Accessed on: 2020-10-03]

Drela, M. and Youngren, H. (2000), 'XFOIL Subsonic Airfoil Development System'. Software. v6.99.
URL: *https://web.mit.edu/drela/Public/web/xfoil/* [Accessed on: 2020-10-03]

Fahlstrom, P. G. and Gleason, T. J. (2012), *Introduction to UAV Systems*, 4th edn, John Wiley & Sons, Ltd. ISBN: 978-1119978664.

Galassi, M. e. a. (2018), 'GNU Scientific Library Reference Manual'. Software. v2.6. **URL:** *https://www.gnu.org/software/gsl/* [Accessed on: 2020-10-04]

Gomila, L. (2020), 'Simple and Fast Multimedia Library'. Software. v2.5.1. **URL:** *https://www.sfml-dev.org/* [Accessed on: 2020-10-04]

Herbst, S. (2017), Development of an Aircraft Design Environment Using an Object-Oriented Data Model in MATLAB, Dissertation, Institute of Aircraft Design - Technical University Munich, Munich.

Hornung, M. (2011), UAV Research Platform IMPULLS (Innovative Modular Payload UAS-LLS), Technical report, Institute of Aircraft Design - Technische Universität München.

Kim, S. and Ahn, S. J. (2005), Extraction of Geometric Primitives from Point Cloud Data. Proceedings in International Conference on Control, Automation and Systems 2005.

Kulfan, B. M. and Bussoletti, J. E. (2006), "Fundamental" Parameteric Geometry Representations for Aircraft Component Shapes. Proceedings in AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference 2006. doi: 10.2514/6.2006-6948.

Nielson, G. (2011), *Techniques for Extracting and Modeling Geometric Features from Point Cloud Data Sets with Application to Urban Terrain Modeling*, pp. 175–195. doi: 10.1007/978-3-642-22907-7_10.

Rusu, R. B. and Cousins, S. (2011), 3D is here: Point Cloud Library (PCL), pp. 1–4. Proceedings in IEEE International Conference on Robotics and Automation 2011. doi: 10.1109/ICRA.2011.5980567.

Steffen, M. (1990), 'A simple method for monotonic interpolation in one dimension.', *A&A* **239**, 443–450. doi: https://ui.adsabs.harvard.edu/abs/1990A&A...239..443S.

Stewart, G. W. (1996), *Afternotes on Numerical Analysis*, Society for Industrial and Applied Mathematics. ISBN: 978-0898713626.

Valigi, M. C., Logozzo, S., Canella, G. and de Angelis, F. (n.d.), 'Reverse Engineering techniques: From 3D scanning to the CAD file in the concrete industry', **85 (5)**, 50–57.

Várady, T., Martin, R. R. and Cox, J. (1997), 'Reverse Engineering of Geometric Models — an Introduction', *Computer-Aided Design* **29**(4), 255 – 268. Reverse Engineering of Geometric Models. doi: https://doi.org/10.1016/S0010-4485(96)00054-1.

Wang, Y., Wang, J., Chen, X., Chu, T., Liu, M. and Ting, Y. (2018), 'Feature Surface Extraction and Reconstruction from Industrial Components Using Multistep Segmentation and Optimization', *Remote Sensing* **10**, 1073. doi: 10.3390/rs10071073.

Çetin, K. M. (2020), Development and Implementation of a 3D-Scanning-Toolchain for the Characterization of UAVs, Bachelor's thesis, Institute of Aircraft Design - Technical University Munich.

Declaration of Originality

I hereby declare that this thesis is my own work prepared without the help of a third party. No other than the listed literature and resources have been used. All sources transferred literally or analogously to this work have been labeled accordingly. Additionally, I hereby certify that this thesis has not been underlain in any other examination procedure up to the present.

Munich, October 30, 2020

Lina van Brügge