Hochschule Rhein-Waal
Rhine-Waal University of Applied Sciences
Faculty of Communication and Environment
Prof. Dr.-Ing. Rolf Becker

Westfälische Wilhelms-Universität Münster
University of Münster
Remote Sensing and Spatial Modelling Research Group
Dr. rer. nat. Jan Lehmann

Development of a UAV-borne Lidar System and its application in contrasting environments

Master Thesis
by
Marcel Dogotari

Hochschule Rhein-Waal
Rhine-Waal University of Applied Sciences
Faculty of Communication and Environment
Prof. Dr.-Ing. Rolf Becker

Westfälische Wilhelms-Universität Münster
University of Münster
Remote Sensing and Spatial Modelling Research Group
Dr. rer. nat. Jan Lehmann

Development of a UAV-borne Lidar System and its application in contrasting environments

A Thesis Submitted in
Partial Fulfillment of the
Requirements of the Degree of

Master of Science
in
Information Engineering and Computer Science

by

Marcel Dogotari
Grevener Str. 8
48149 Münster

Matriculation Number:
15295

Submission Date
10 June 2021

ABSTRACT

A versatile UAV-borne lidar system has been developed then applied towards microforms mapping in a bog and tree segmentation in a near-natural beech forest. The lidar system is based on an off-the shelf lidar sensor. Georeferencing of its point clouds has been done with a survey-grade navigation unit and further refined with a global-shutter RGB camera. Hardware solutions have been developed for seamless integration of the components. A complete software chain from data acquisition to point clouds generation has been produced. The system's boresight calibration has been carried out with a novel method, which decreased the roll and pitch uncertainty to $0.18°$ and yaw uncertainty – to$1.6°$. The system has been applied to two contrasting environments: a rewetted cut-over bog and a beech forest. Mismatched point clouds acquired over the bog have been successfully aligned and combined to produce 1×1 cm elevation models of the bog's surface to further detect microforms. Regarding the forest, a novel tree segmentation algorithm has been developed. Qualitative assessment of its output revealed that besides mostly correct results, both over- and under-segmentation occurred. While the median tree height was underestimated by 1.5 m, the distribution of heights closesy resembled the ground truth.

Keywords: UAV, lidar, bog microforms, tree segmentation, system integration, boresight calibration.

CONTENTS

List of Abbreviations V

List of Figures VI

List of Tables VII

1 Introduction 1

2 Description of the data acquisition campaigns 3

2.1 Vechta 3

2.2 Niederkamp 4

2.3 Duisburg 5

3 System description 6

3.1 Coordinate systems and georeferencing 7

3.2 Hardware 10

3.2.1 Interface boards 10

3.2.1.1 APX interface board 10

3.2.1.2 Puck interface board 14

3.2.1.3 RGB-Camera interface board 15

3.2.2 Power electronics 17

3.3 Software 20

3.3.1 Data acquisition 20

3.3.2 Data processing 23

3.3.2.1 Lidar processing 23

3.3.2.2 Processing of RGB data 28

3.3.2.3 General considerations for data processing 31

3.3.3 Data visualization 31

4 Boresight calibration 33

4.1 Methods and Materials 33

4.1.1 Boresight calibration algorithm 34

4.1.2 Refinement of heading with photogrammetry 35

4.2 Results and Discussion 36

4.3 Conclusion and Outlook 44

5 Use case 1. *Vechtaer Moor*: Towards microforms detection in a rewetted cut-over bog 45

5.1 Methods and Materials 46

5.1.1 Study area and data collection 46

5.1.2 Processing algorithm 47
5.2 Results and Discussion 48
5.3 Conclusion and Outlook 53

6 Use case 2. *Naturwaldzelle Niederkamp*: Segmentation of trees in a near-natural beech forest 54
6.1 Introduction 54
6.1.1 Natural forest cells in NRW 54
6.2 Literature review 55
6.2.1 Lidar use in forestry 55
6.2.2 Tree segmentation algorithms 56
6.3 Methods and Materials 58
6.3.1 Study area and data collection 58
6.3.2 Segmentation algorithm 59
6.4 Results and Discussion 61
6.5 Conclusion and Outlook 66

7 Reflection and Outlook 69

References 71

Appendices 79

A Schematic diagrams 80

B Lidar coverage vs distance 93

C Miscellaneous 94
C.1 NMEA Sentences 94

Declaration of Authenticity 95

List of Abbreviations

Abbrev.		**Meaning**
AOI	=	area of interest
CHM	=	cannopy height model
CRS	=	coordinate reference system
CSV	=	comma-separated values
DSM	=	digital surface model
DTM	=	digital terrain model
FOV	=	field of view
GCP	=	ground control point
GDAL	=	geospatial data abstraction library
GNSS	=	global navigation satellite system
GPIO	=	general-purpose input–output
GPS	=	global positioning system
GUI	=	graphical user interface
IC	=	integrated circuit
ICP	=	iterative closest point
IFB	=	interface board
IMU	=	inertial measurement unit
IO	=	input–output
IPL	=	image processing library
LED	=	light emmiting diode
lidar	=	light detection and ranging
LTE	=	long term evolution
NRW	=	North Rhine-Westphalia
NUC	=	Intel® Next Unit of Computing
PCB	=	printed circuit board
PPS	=	pulse per second
RAM	=	random-access memory
RANSAC	=	random sample consensus
RGB	=	red-green-blue
RMSE	=	root-mean-square error
ROI	=	region of interest
RTK	=	real-time kinematic positioning
SBC	=	single-board computer
SDK	=	software development kit
SfM	=	structure from motion
SSH	=	secure shell
TVS	=	transient voltage suppressor
UART	=	universal asynchronous receiver-transmitter
UAV	=	unmanned aerial vehicle
UDP	=	user datagram protocol
UI	=	user interface
USB	=	universal serial bus
UTC	=	coordinated universal time
VRS	=	virtual reference station

LIST OF FIGURES

1.1 The lidar system mounted on the DJI Matrice 600 Pro UAV. 2

2.1 Location of *Vechtaer Moor* and the AOI. 3
2.2 Location of *NWZ Niederkamp*. 4
2.3 Location of pedestrian bridge. 5

3.1 Simplified system diagram: main components and data-flows. 6
3.2 Coordinate systems of the puck, APX and UAV. 7
3.3 Photo of the APX IFB. 12
3.4 Diagram of RTK corrections via WiMOD radio link. 13
3.5 Photograph of the puck interface board. 15
3.6 Photograph of the camera's IFB. 16
3.7 Flight time and power draw for DJI M600 Pro with TB47S accumulators. . 18
3.8 Diagram and photograph of the power electronics. 19
3.9 Typical flight campaign procedure. 21
3.10 Use of flying height, heading difference and velocity to segment flight lines. 24
3.11 Result of flight-lines segmentation: Mission 2 from the Vechta campaign. . 26
3.12 Visualisation of stacked-matrices multiplication. 27
3.13 Demosaicing result: `OpenCV` vs. `IDS SDK`. 29
3.14 Workflows for demosaicing of RGB images. 31
3.15 Screenshot from the GUI, showing one of the three most recent images. . 32

4.1 Ground control points. 34
4.2 Pose error over time – Missions 2–4 in Vechta campaign. 37
4.3 Characterization of `Metashape`–detected boresight errors. 38
4.4 Boresight error for various angles – Vechta with yaw refinement. 39
4.5 Boresight error for various angles – Niederkamp 1. 39
4.6 Boresight error for various angles – Niederkamp 2. 40
4.7 Boresight error for various angles – Duisburg. 40
4.8 Boresight error for various angles – Combined. 41
4.9 Boxplots of DTM differences. 42
4.10 Difference between DTMs generated from flight lines 1 and 2. 43

5.1 Orthomosaic of AOI and location of the 20×20 m ROI. 46
5.2 Point heights in input files. 48
5.3 DSM produced from the original, misaligned, point clouds. 49
5.4 Smooth DTMs before ICP: top-view. 50
5.5 Smooth DTMs before ICP: side-view of point clouds from the east. 50
5.6 Smooth DTMs after ICP: side view of point clouds from the east. 51
5.7 Smooth DTMs after ICP: top-view. 51

5.8 Point heights after translation and rotation of input files. 52
5.9 Low-frequency DTM (left) and high-frequency DSM (right). 52
5.10 DSM produced from the final, aligned, point clouds. 53

6.1 Location of the three core areas within the natural forest cell Niederkamp. 58
6.2 Segmentation results for core area 1 $(50 \times 100$ m). Top–view. 61
6.3 Segmentation results for core area 1 $(50 \times 100$ m). Side–view. 62
6.4 Example of a well–segmented tree. 63
6.5 Example of grave under-segmentation. 64
6.6 Distribution of tree heights in Core Area 1: field–data vs. segmentation. . . 65
6.7 Core Area 1. Multiple flight lines combined. 66
6.8 Distribution heights in point cloud of Core Area 1. 67

LIST OF TABLES

4.1 Results of boresight calibration for each mission. 38

6.1 Summary of tree heights in Core Area 1: field–data vs. segmentation results. 65

1. INTRODUCTION

Bog surfaces relate vital information about their condition, species composition and functions (Couwenberg et al., 2011). Current approaches to capture relevant microforms with high-resolution SfM[1] or low-density airborne lidar[2] are limited either in accuracy or resolution (Korpela et al., 2009; Luscombe et al., 2015; Lovitt et al., 2017). A novel approach using high-density UAV[3] lidar to map microforms in a bog that is currently being restored through rewetting was developed.

Different plant communities form characteristic microforms, such as lawns, hummocks and hollows. By detecting the microforms, the underlying vegetation types and functions of the bog, such as carbon fixation or methane production can be mapped (Lehmann et al., 2016). Also, mapping microforms and topography support restoration efforts (Dargie, 2003; Raabe, Kleinebecker, Knorr, Hölzel, & Gramann, 2018). During restoration through rewetting, one of the knobs to influence which communities are established is the water table. By adjusting its level, optimal growth conditions for certain species are created at different locations. It is therefore paramount to be able to describe the spatial distribution of the diverse microforms and the bog's topography.

For instance, if numerous positive key species were identified at a higher general elevation, the water table could be raised to provide them with optimal growth conditions, eventually sacrificing some “not-as-valuable” plant communities at a lower elevation. This is just a speculative example of an action based on knowledge of elevation profiles and microforms distribution. In practice, the decisions to be made are more complex. Nonetheless, the usefulness of accurately describe the structure of a bog's surface is evident.

So far, bog surfaces have been captured with low-density airborne lidar and SfM using high-resolution RGB[4] images. Lovitt et al. (2017) found that both methods overestimate the true elevation with RMSEs[5] of 40 and 84 cm respectively. UAV-borne lidar sensors have the potential to significantly improve these results. In contrast to lidars flown at higher altitude, these provide considerably better point densities. Compared with SfM techniques, UAV-lidar provide higher global accuracy.

To the best of my knowledge, no airborne lidar mapping of bog microforms has been conducted at an altitude lower than 250 m (Korpela, Haapanen, Korrensalo, Tuittila, & Vesala, 2020). This thesis describes the development of a UAV-borne lidar system and its application to capturing the structure of a bog's surface. The system uses an off-the-shelf lidar sensor (Velodyne LiDAR, Inc., 2018a), a survey-grade GNSS[6]-IMU[7] unit

---

[1] structure from motion

[2] light detection and ranging

[3] unmanned aerial vehicle

[4] red-green-blue

[5] root-mean-square errors

[6] global navigation satellite system

[7] inertial measurement unit

(Trimble Applanix, 2016) and a global shutter RGB camera (IDS GmbH, 2020d). Further hardware and software, as well as a boresight calibration method were developed in-house.

In order to verify the system's universality, it was further used to segment trees in a forested area. This way, its usability for mapping features ranging from sub-metre microforms to 40 m tall trees has been investigated.

The herein presented system was developed within the SPECTORS project (Becker & Mosler, 2019) and has so far been presented at an international conference (Dogotari et al., 2019). A photograph of the system in its current state mounted on a carrier UAV is shown in Figure 1.1.

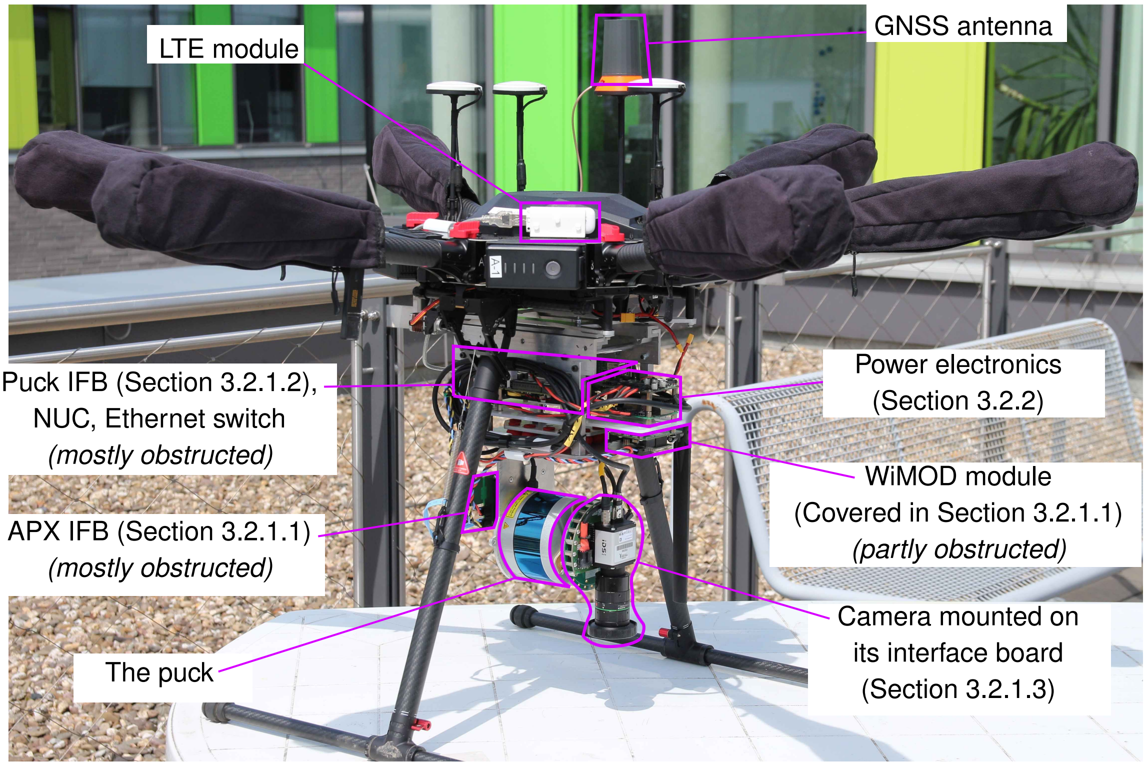


Figure 1.1: The lidar system mounted on the DJI Matrice 600 Pro UAV.

The rest of the paper is structured as follows: Chapter 2 presents the acquisition campaigns for the data used throughout the thesis. Chapter 3 establishes relevant coordinate systems, describes the system and explains its hardware and software implementation. Chapter 4 goes into further details regarding the boresight calibration of the system. Chapter 5 motivates the detection of microform in bogs then describes an algorithm for fusing data from misaligned flight lines, using the bog *Vechtaer Moor* in Lower Saxony, Germany as an case study. It also presents the current results and discusses future implementations. Chapter 6 deals with the segmentation of trees in a near-natural beech forest in a natural forest cell in the German state of North Rhine-Westphalia. Chapter 7 provides closing remarks on the entire system, as well as the two use cases.

2. DESCRIPTION OF THE DATA ACQUISITION CAMPAIGNS

Data from three flight campaigns (two in Niederkamp and one in Vechta) was extensively used throughout this thesis. These campaigns are described in detail in Chapters 6 and 5. But references to the corresponding datasets were inevitable in Chapters 3 and 4. In order to aid understanding the earlier chapters without reading the later ones first, a short description of the sites and the flight campaigns is given in this section. Moreover, data from a fourth campaign in Duisburg had been used for the system's boresight calibration. No further analysis was performed on this data but for the sake of completeness, a short description of the Duisburg campaign follows at the end of the current Chapter.

2.1. VECHTA

The flight campaign Vechta was conducted on 19.08.2020 over a small AOI[8] in the *Vechtaer Moor* (the Bog Vechta). The Bog Vechta is situated in the districts of Vechta and Diepholz in the state of Lower Saxony and covers more than 20 km^2. The area of interest that was flown over lies completely within the district of Vechta and has an area of 10 hectares. The location of the bog, as well as that of the AOI is presented in Figure 2.1.

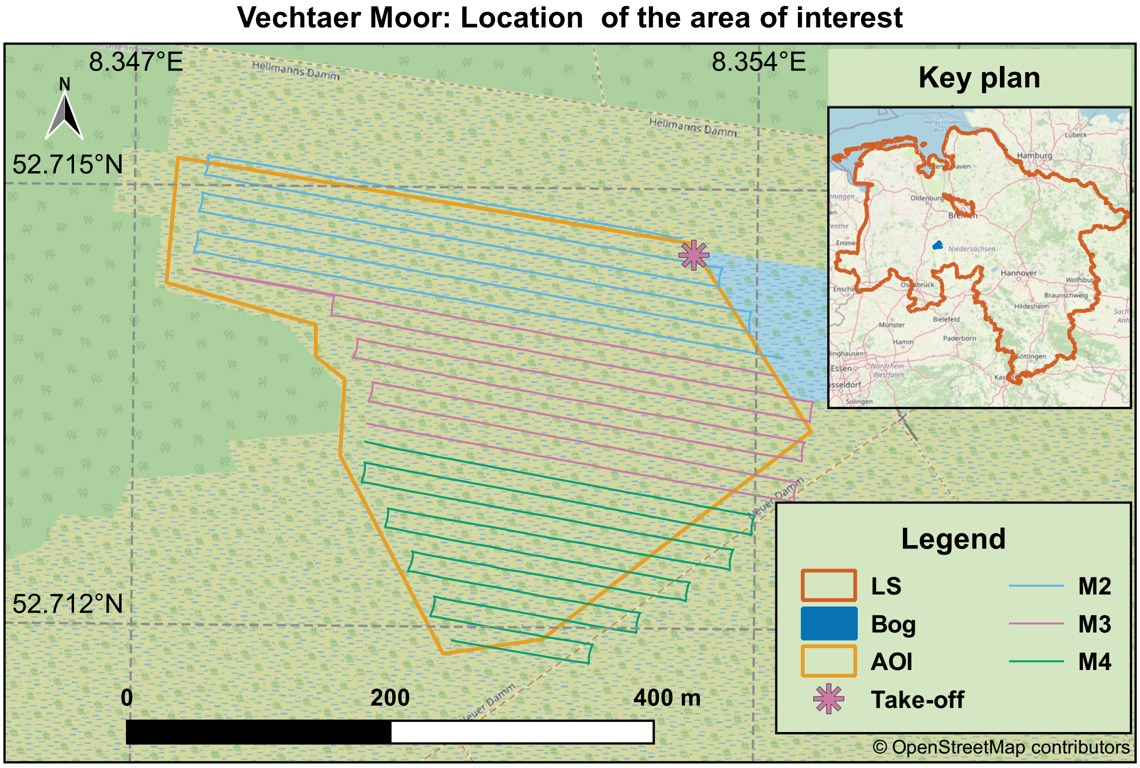


Figure 2.1: Location of *Vechtaer Moor* and the AOI. Top right: position of the Bog Vechta within Lower Saxony (*LS*). Main map: Outline of the AOI, paths of the flights: missions two, three & four.

---

[8] area of interest

Three missions had been flown in the Vechta flight campaign: M2, M3 and M4. An initial flight M1 along the same route as M2 had been flown beforehand, but its data was discarded, since no heading alignment (Jaeger, 2017) of the GNSS-IMU unit had been performed prior to data acquisition. The analysis of the data from the Vechta flight campaign is presented in Chapter 5.

2.2. NIEDERKAMP

Campaigns Niederkamp 1 and 2 were conducted over a natural forest cell, or *Naturwaldzelle (NWZ)* in Niederkamp, which is part of the town Kamp-Lintfort (District Wesel, NRW[9]). The cell has an area of 8.2 hectares. The general location of the *NWZ* is presented in the Figure 2.2.

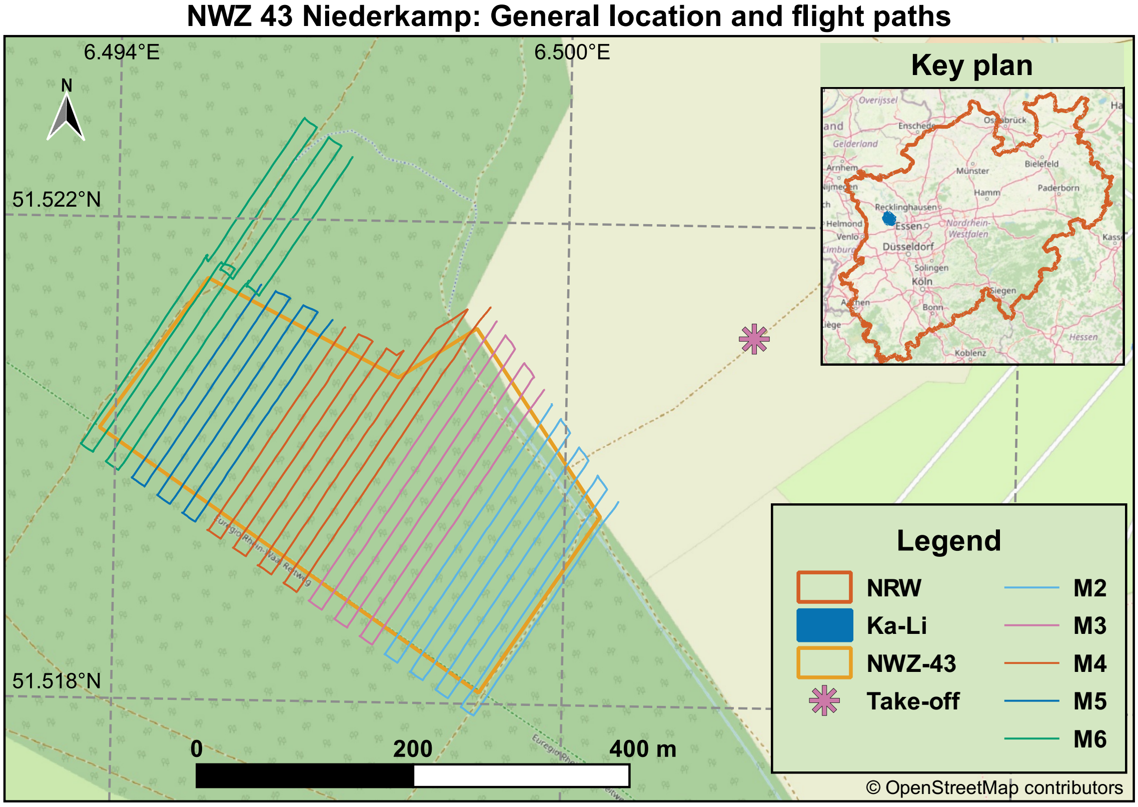


Figure 2.2: Location of *NWZ Niederkamp*. Top right: position of Kamp-Lintfort (*Ka-Li*) within North Rhine-Westphalia (*NRW*). Main map: Outline of the natural forest cell Niederkamp (*NWZ-43*) and paths of five flights: missions two, three, four, five and six. The take-off point from the field next to the forest is also shown.

The campaign Niederkamp 1 took place on $30.11.2020$ and consisted of a single flight: mission M2 in Figure 2.2. This data was only used to perform the boresight calibration of the system, which is thoroughly explained in Chapter 4.

The campaign Niederkamp 2 consisted of five flights: the same M2 as in Niederkamp 1 and M3–M6 as seen in Figure 2.2. It took place on $18.12.2020$ and its data was used for conducting the analysis outlined in Chapter 6. The labelling of the flight lines started at 2 because one dummy data acquisition was performed in the laboratory prior to both campaigns and that data was preserved and kept its label. The last mission (M6) went

---

[9] North Rhine-Westphalia

beyond the targetted area to collect further data from a similar part of the forest. This was facilitated because it was the last flight of the day and the UAV's accumulators still had a considerable amount of charge left.

2.3. DUISBURG

Another flight campaign had been carried out in cooperation with the surveying company Planungs- und Vermessungsgesellschaft ANSPERGER mbH (PVA) from Kamp-Lintfort, who were a partner in the SPECTORS project. The campaign took place in Duisburg and the object of interest was a pedestrian bridge that crosses railroads on the territory of the steel concern thyssenkrupp Steel Europe AG (TKS). This bridge was chosen because PVA scanned it from the inside and from underneath with a terrestrial laser scanner but also needed some information from above to create a more complete model of the bridge. The location of the bridge is shown in Figure 2.3.

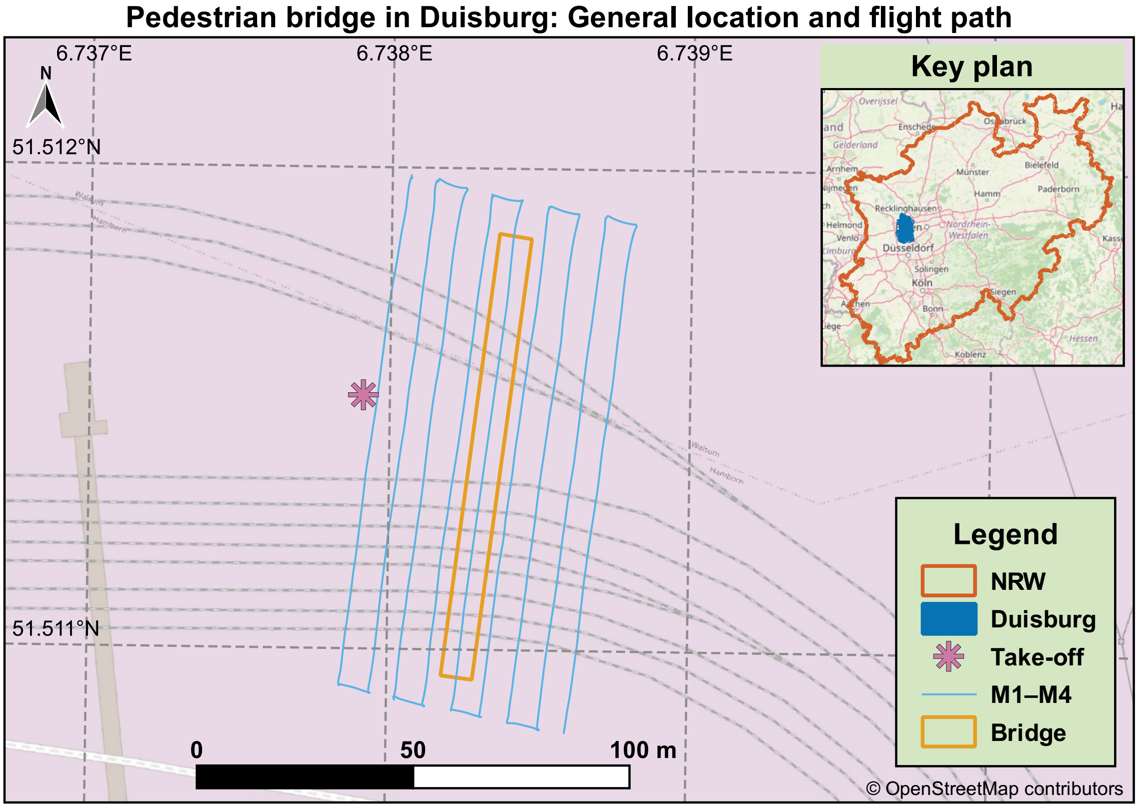


Figure 2.3: Location of pedestrian bridge. Top right: position of Duisburg within North Rhine-Westphalia (*NRW*). Main map: Outline of the pedestrian bridge, flight path and take-off location.

Four flights have been conducted along the path shown in Figure 2.3. The speed has been varied between 2 and 5 m/s and the camera shutter interval was also changed to obtain a few different front overlap ratios. The lidar dataset acquired during this campaign was used when performing the boresight calibration of the system (Chapter 4). No further analysis of the data had been carried out in the context of this thesis.

3. SYSTEM DESCRIPTION

In this section, first a high-level overview of the system architecture is presented. Next, the relevant coordinate systems are established and details of hardware and software implementation are given.

The current paper presents a UAV-borne laser scanner system. At its core, the system has a Velodyne VLP-16 Puck LITE lidar sensor (called puck in the rest of the paper). The puck uses 16 laser–diode pairs to conduct ≈ 300 thousand time-of-flight measurements of its surroundings each second. The positions of the laser returns are provided by the puck in its own reference frame (Velodyne LiDAR, Inc., 2019b, p. 53). To convert these measurements to meaningful georeferenced point clouds, the position and orientation of the puck relative to the Earth needs to be known as well. A high-grade GNSS-IMU unit from Applanix – the APX-15 RTK[10] (simply APX from now on) – is used for this purpose. The APX records the pose of the lidar unit in a world reference frame and data from both sensors are combined to obtain the longitude, latitude and elevation of the lidar readings. Both these data streams are recorded on a SBC[11]. A second-generation NUC[12] has been chosen for this task. The APX gets its RTK corrections via an LTE[13] USB[14] stick. Alternatively, a backup radio for operation in remote areas has also been implemented. Furthermore the system features a 12-megapixel global-shutter RGB camera used for generating photogrammetric products. Its raw images are also saved on the NUC. The system, weighing 3.3 kg and requiring up to 45 W of power, is flown on a DJI Matrice 600 Pro UAV. A simplified system diagram, complete with data flows is shown in Figure 3.1.

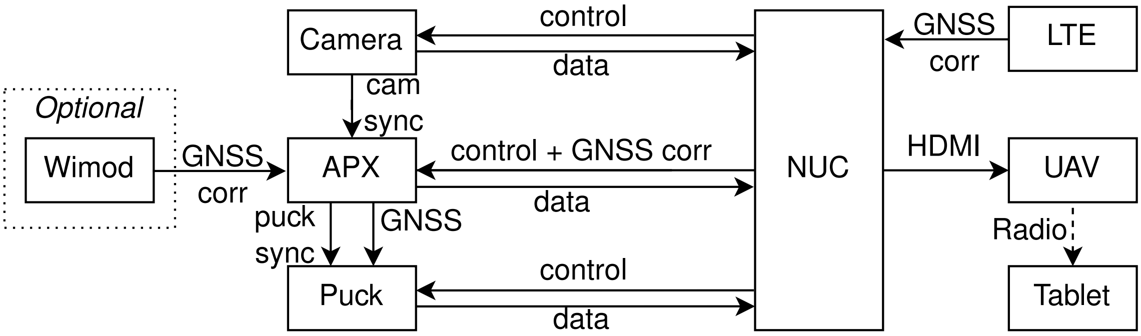


Figure 3.1: Simplified system diagram: main components and data-flows. The NUC acts as a control hub and data storage for all sensors. *(See Sections 3.2.1 & 3.3.1.)* The APX generates synchronization signals for the puck and saves timestamps for camera shots; GNSS corrections are delivered to the APX either via LTE or a custom radio link. *(See Sections 3.2.1.1–3.2.1.3.)* The NUC–HDMI–UAV–Radio–Tablet path represents a video stream sent from the NUC to the UAV operator. *(See Section 3.3.3.)*

---

[10] real-time kinematic positioning

[11] single-board computer

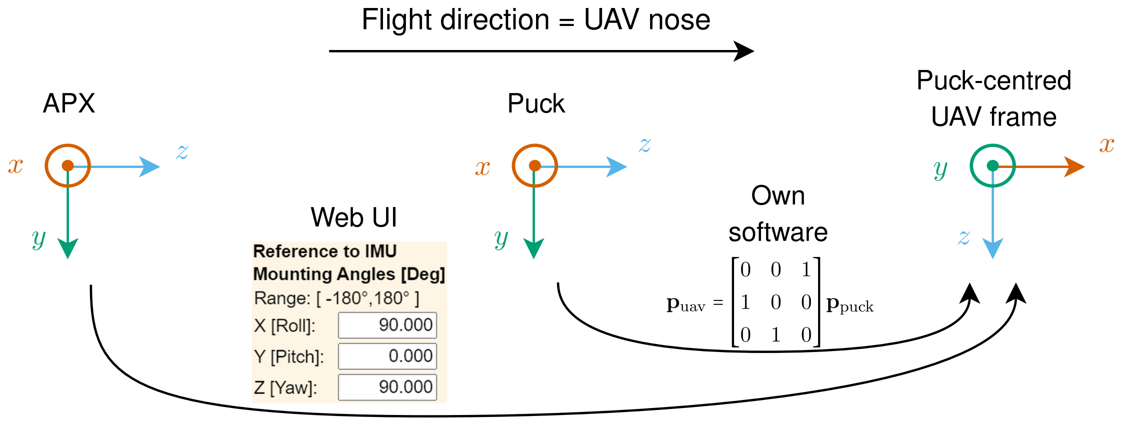
[12] Intel® Next Unit of Computing

[13] long term evolution

[14] universal serial bus

3.1. COORDINATE SYSTEMS AND GEOREFERENCING

This section shortly presents the coordinate systems of the relevant components (lidar unit, GNSS-IMU device, UAV frame) and outlines how they relate to each other, as well as to locations on the Earth. Parallelly, the transformations necessary for georeferencing the lidar returns are presented. These are mainly rotations and they depend strongly on how the particular components are mounted. Therefore, Figure 3.2 displays the native coordinate systems of the puck, APX and UAV in their current mounting positions.



Reference to IMU Mounting Angles [Deg]
Range: [-180°,180°]
X [Roll]: 90.000
Y [Pitch]: 0.000
Z [Yaw]: 90.000

$$\mathbf{p}_{\text{uav}} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{p}_{\text{puck}}$$

Figure 3.2: Coordinate systems of the puck, APX and UAV. Transformations from the APX frame to the puck-centred UAV frame are automatically done by APX, after applying the appropriate settings in its web UI. The puck to UAV transformations are done in own software. *(Details below.)*

The “puck-centred UAV frame” in Figure 3.2 is an imaginary construct. The orientation and naming of its axes are consistent with the convention used by Nonami et al. (2010), Concurrently, this frame has its origin at the puck’s centre, regardless of the UAV onto which the system might be mounted. The advantage of using such a construct is that the roll, pitch and heading angles output by the APX have a well-defined meaning. Moreover, transformations between the puck and this frame are simplified to just one rotation.

The APX is used to relate the lidar ranging measurements to point locations in the real world. Therefore it was configured to output the position and orientation of the puck’s centre relative to a world reference frame. By precisely positioning the puck’s centre instead of any other point in the system, the georeferencing is significantly simplified. To enable this function, the procedure documented by Applanix Corporation (2019b, pp. 17-23) was followed. It consisted of measuring mounting angles between APX, puck and UAV, as well as lever arms between APX, puck and GNSS antenna. These were measured in CAD and confirmed in real life, then configured in APX’s web UI[15].

The APX outputs the orientation as three angles: roll, pitch and true heading. These represent rotations of the APX’s target reference frame relative to a world frame. The angles refer to simple rotations about x, y and z respectively. The roll, pitch and heading can be interpreted more intuitively when the orientation of the UAV’s frame is regarded instead of that of the puck. For instance consider a yawing motion: it would be correctly seen in heading in the case of the UAV, but might be confused with pitch in the puck’s native system. Therefore, the aircraft’s reference frame was configured as the output system in

---

[15] user interface

APX's settings menu. As a result, the APX's output is such as would be measured by a GNSS-IMU unit situated in the puck's centre but with the IMU's x axis pointing to the UAV's nose and the y-axis—to the UAV's "right wing". Besides making the output angles more interpretable, using the UAV's frame for APX's output also ensures that the data processing can be more readily adapted to systems with other mounting positions for the lidar and GNSS-IMU units: Only this initial step would need to be adapted for a different system, but successive steps should work as they are.

Now given that the APX outputs its data in the puck-centred UAV frame, it is important to transform the puck data to this exact frame before the two data-streams can be combined. The respective procedure is described below.

Every laser return of the puck has its position described by the following spherical coordinates in the puck raw data: range R, elevation ω and azimuth α. Consider the vector $\mathbf{p}$ that connects the origin $O\left(0,0,0\right)$ of the puck's coordinate system to a laser return $P\left(X,Y,Z\right)$. Then R represents its length: $R = \|\mathbf{p}\|$. The azimuth α is the angle between the y axis and the projection of $\mathbf{p}$ onto the xy plane ($\mathbf{p}_{xy}$). The elevation ω is then the angle between $\mathbf{p}$ and $\mathbf{p}_{xy}$. Following these definitions (Velodyne LiDAR, Inc., 2019b, pp. 53-54), the Cartesian coordinates of P in puck's reference frame are:

$$\mathbf{p}_{\text{puck}} = \begin{bmatrix} X_{\text{puck}} \\ Y_{\text{puck}} \\ Z_{\text{puck}} \end{bmatrix} = R \begin{bmatrix} \cos\omega \sin\theta \\ \cos\omega\cos\theta \\ \sin\omega \end{bmatrix} \tag{3.1}$$

Now consider the relative orientation of the UAV's and puck's coordinate systems. It takes two rotations around the axes to map one onto the other. For instance, the puck's coordinate system can be rotated to that of the UAV by first rotating around its original x axis by $-\frac{\pi}{2}$, then around the new z' axis by $-\frac{\pi}{2}$. However if one wants to change the location of the laser returns from the puck's coordinate into that of the UAV, the inverses of these rotations have to be used. So if the position of the point P in puck's reference frame (Equation 3.1) is considered, then its coordinates in the UAV's frame are:

$$\mathbf{p}_{\text{uav}} = \begin{bmatrix} X_{\text{uav}} \\ Y_{\text{uav}} \\ Z_{\text{uav}} \end{bmatrix} = \mathbf{R}_{z'}\left(\frac{\pi}{2}\right)\mathbf{R}_{x}\left(\frac{\pi}{2}\right)\mathbf{p}_{\text{puck}} \tag{3.2}$$

Note that the order of rotations *does* matter and in the equation above $\mathbf{R}_x\left(\frac{\pi}{2}\right)$ is executed *before* $\mathbf{R}_{z'}\left(\frac{\pi}{2}\right)$. The following notation for rotations $\mathbf{R}$ around each of the axes x, y and z by the angle θ are used throughout this paper:

$$\mathbf{R}_x\left(\theta\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}; \quad \mathbf{R}_y\left(\theta\right) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}; \quad \mathbf{R}_z\left(\theta\right) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

The rotation matrices from Equation 3.2 can be expanded and then multiplied as follows:

$$\mathbf{p}_{\text{uav}} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{p}_{\text{puck}} \tag{3.4}$$

$$\mathbf{p}_{\text{uav}} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{p}_{\text{puck}} \tag{3.5}$$

The relationship obtained in Equation 3.5 is used to obtain the coordinates of all lidar returns in UAV's reference frame. At this point both the puck and APX data are in the same coordinate system, so georeferencing the points is trivial. Given APX's output angles roll (γ), pitch (β) and true heading (α), the following rotations calculate the points' coordinates in a north-east-down (NED) system with the origin at the puck's centre:

$$\mathbf{p}_{\text{ned}} = \mathbf{R}_{z''}(\alpha)\,\mathbf{R}_{y'}(\beta)\,\mathbf{R}_{x}(\gamma)\,\mathbf{p}_{\text{uav}} \tag{3.6}$$

It can be verified that the result is indeed in a NED system by assuming a point with the following UAV coordinates: $\mathbf{v}_{\text{uav}} = [x_0\, y_0\, z_0]$ and the following angles: $\alpha = 0, \beta = 0, \gamma = 0$, which mean the UAV's nose is pointing exactly to the north and its right wing to the east. Rotating by zero degrees around each of the axes, leaves the coordinates unchanged, so $\mathbf{v}_{\text{ned}} = [x_0\, y_0\, z_0]$. By consulting the UAV's coordinate system and its current orientation, it becomes apparent the point's coordinates x_0 units towards north, y_0 units towards east and z_0 units down, the same as a NED system.

The following rotation transforms the coordinates obtained in Equation 3.6 from NED to east-north-up (ENU)—a more intuitive coordinate system:

$$\mathbf{p}_{\text{enu}} = \mathbf{R}_{x'}(\pi)\,\mathbf{R}_{z}\left(-\frac{\pi}{2}\right)\mathbf{p}_{\text{ned}} \tag{3.7}$$

Now the georeferenced location of each laser return is calculated by adding the position of the puck to the result from the Equation 3.7:

$$\mathbf{p}_{\text{geo}} = \mathbf{p}_{\text{enu}} + \begin{bmatrix} \text{Easting} \\ \text{Northing} \\ \text{Elevation} \end{bmatrix} \tag{3.8}$$

Where Easting and Northing are obtained by representing the APX-provided puck coordinates in a projected CRS[16] and the Elevation is output directly by the APX as the puck's height above the geoid.

The transformations described in the current section were implemented in software. Further programming details are presented in Section 3.3.2.1.

---

[16] coordinate reference system

3.2. HARDWARE

The hardware integration of the system had the following aims:

- Establish communication interfaces between components;
- Supply power to each component;
- Balance feature richness with ease of use;
- Ensure reliable operation under normal use conditions.

Besides these overall targets, further practical considerations were made. For instance, it was opted to use standard connectors for interfaces which would be routinely exposed to the outside of the system, like USB and Ethernet. But more ruggedized options were chosen for either non-standardized interfaces, or for connections that are only seldom accessed from outside, such as power connections and digital IO[17] between devices. Moreover, the enclosures of the NUC and the Ethernet switch were discarded to enable easier mounting. Because the system was developed as a prototype, weight saving and other optimizations were not considered in the current paper. Some high-level details of hardware design and implementation are presented in the following subsections. The relevant schematics can be found in Appendix A.

3.2.1. INTERFACE BOARDS

This section succinctly presents the overall functions and a few features of the PCBs[18] that were designed to be used at the interfaces of the three main sensors: puck, APX and RGB camera. Furthermore, the motivation as well as some design decisions for each of the IFBs[19] are presented in the following subsections.

3.2.1.1. APX interface board

The APX is a feature-rich board, providing Ethernet and USB among other interfaces. However, to enable a miniature size (67 x 60 x 15 mm) and mass (60 g), the APX exposes all its electrical interfaces only through a 44-pin IO connector (Trimble Applanix, 2016). The manufacturer provides an evaluation board (Applanix Corporation, 2016, p. B-1), that routes all these signals to standard and more user-friendly connectors. While extremely versatile, the evaluation board is over 10 x 15 cm large, taking up more space than even the NUC. Consequently, a much smaller IFB was developed in-house. Its functions were restricted to the minimum necessary:

- Supply adequate power to the APX;
- Provide a communication interface to the NUC;
- Facilitate synchronization of APX and Puck;
- Enable reception of GNSS correction messages;
- Feature a few LEDs[20] for easy debugging on the ground.

Synchronization with the puck The main function of the APX is to record the position and orientation (together called pose) of the puck in a world frame. This is then used to

---

[17] input–output

[18] printed circuit boards

[19] interface boards

[20] light emmiting diodes

compute the georeferenced coordinates of the puck's laser returns. The puck's pose is obtained by recording APX's own pose and applying translations and rotations to account for the sensors' relative mounting. However, since the puck rotates at $5 - 20$ Hz and the dynamics of the system overall are fast-changing, very precise alignment of the data streams is necessary. One way to accomplish this is to have the sensors provide accurate timestamps for their data. A very accurate and readily available time basis is the GPS[21] time, which is computed by the APX (or really any other GNSS-receiver) simultaneously with their position—it is a core component of all GNSS systems. The GPS time differs from the UTC[22] by a fixed offset. Since $31.12.2016$ the offset has been exactly 18 seconds and the earliest possible adjustment might occur on $31.12.2021$ (Bizouard, 2021). The APX measures the GPS time, transforms it to UTC and communicates it to the puck. The procedure is documented by the manufacturers of both sensors (Velodyne LiDAR, Inc., 2019b, pp. 41-49; Applanix Corporation, n.d., pp. 6-8).

The synchronization is implemented as follows: The APX generates a PPS[23] signal, that is a short pulse whose rising edge coincides with the beginning of a UTC second. This pulse is shortly followed by a GNSS message specifying the UTC at the instant of the PPS's rising edge. The puck then uses this information to interpolate the timestamp of each Ethernet packet that it sends to the NUC. To make use of this feature, the PPS signal of the APX, the TX[24] of its RS232[25] port as well as a ground connection had to be wired to an external connector which then goes to the puck via a cable.

Communication interface The APX has a web UI through which all the important settings can be changed, the positioning solution can be checked in real-time and even firmware updates can be made. Furthermore, the sensor's data can be retrieved via Ethernet. Therefore it was crucial to integrate the APX's Ethernet interface into the IFB. Four connections had to be made from the 44-pin connector on the APX to a standard RJ45 connector on the IFB in order to make use of the APX's integrated $10/100$BASE-T[26] Ethernet controller (Applanix Corporation, 2019a, p. 9).

Transformers (aka magnetics) are crucial for signal conditioning in Ethernet applications. The APX's documentation specifies that "the magnetics are implemented on the board" [i.e. on the APX] (Applanix Corporation, 2019a, p. 9), making further magnetics on an IFB redundant. However the manufacturer's own evaluation board (Applanix Corporation, 2016, p. B-1) does includes an Ethernet transformer as well as a TVS[27] diode array. In order to offer the same level of protection, both were added to the new IFB. They were assembled in a circuit proposed by Marak and Havens (2015, p. 14). This circuit is meant to protect the Ethernet data lines from over-current, transient voltages and electrostatic

---

[21] global positioning system

[22] coordinated universal time

[23] pulse per second

[24] Transmit wire on multiple serial communication interfaces.

[25] A low-speed serial communication protocol, functioning both with and without flow control.

[26] 10BASE-T and 100BASE-T refer to 10 and 100 Mbps Ethernet respectively. Both these standards use two differential pairs: one for transmitting and one for receiving data.

[27] transient voltage suppressor

discharge. Further recommendations for placement and routing of Ethernet components (Microsemi, 2018; Pulse Electronics, 2020) were also followed when designing the IFB.

Power safety features Even though the power electronics that were designed for the system provide extensive protections from various potential power failures (see Section 3.2.2), further safety measures were implemented for the APX. These were deemed necessary because the APX is by far the most expensive component in the system. Furthermore, while one would have to use either my or the manufacturer's IFB to connect to the APX, it would be trivial to bypass the power electronics and therefore the protections described in Section 3.2.2. To this end, an IC[28] featuring "overvoltage, undervoltage and reverse supply protection" (Analog Devices, Inc., 2019) was selected and implemented in the IFB circuitry.

Simplified graceful shutdown Besides the real-time output of position and orientation, the APX supports logging of raw IMU and GNSS measurements to its internal memory. Since the logging takes place continuously, simply powering off the board might lead to data corruption. In order to guarantee the integrity of the data, the manufacturer recommends that a graceful shutdown be implemented whereby the power can only be turned off after sending APX a signal and receiving the board's OK to shutdown (Applanix Corporation, 2019a, pp. 7–8). This effectively gives the board time to safely close any open files before being powered off. The manufacturer's evaluation board implements this by using two slide switches that have to be operated in a simple, yet strict sequence for both powering up and down the board. To minimize the risk of data corruption due to human error, the graceful shutdown feature on our IFB was implemented with just one switch and a few logic gates. As a result, turning on and off the board takes only one flip of a switch.

An annotated photograph of the APX IFB is shown in Figure 3.3.

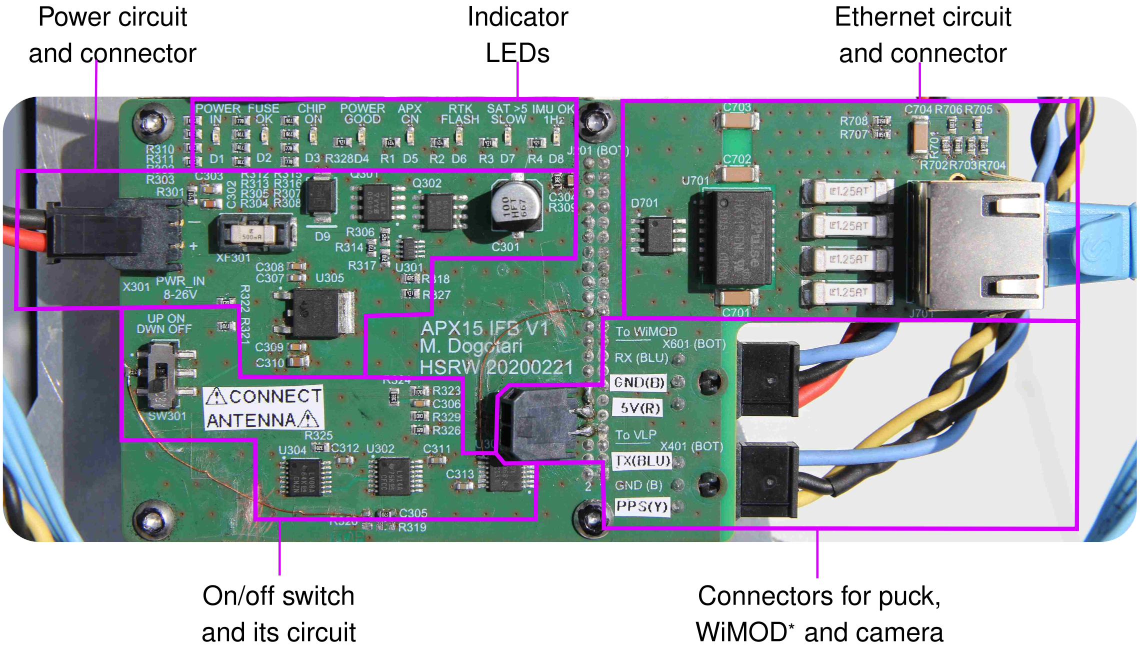


Figure 3.3: Photo of the APX IFB. * WiMOD is the radio interface explained in the next paragraphs.

---

[28] integrated circuit

Radio interface for RTK corrections The RTK function of the APX requires corrections from another GNSS receiver. In RTK terminology, the APX is a rover, while another stationary receiver – a base. Normally, the base is a physical receiver situated nearby, that sends its observations to the rover via a radio link. Alternatively, VRSs[29] can be used. Their "observations" are generated by a network of receivers usually some distance away, that infer what a GNSS receiver would observe near the rover's location.

In the present system, two channels for delivering corrections to the APX have been implemented. By default, corrections from a VRS are received via internet. The LTE module is responsible for providing access to the internet when the system is airborne.

A backup solution was implemented for areas with poor cellular coverage. Two radios were used to establish a link for sending the corrections from a physical base. Radio links are always subject to trade-offs between multiple factors, such as power usage, achievable distance and link throughput. In order to allow flexibility when balancing these factors, WiMODino™ boards (IMST GmbH, 2019) were used. They provide an Arduino®-compatible programming interface for an array of WiMOD™ radio modules that use chirp spread spectrum modulation techniques to obtain high-budget but relatively low-throughput radio links (Seller & Sornin, 2013).

A significant advantage of the WiMODino™ boards is that the modules they carry operate at two different frequencies: 868 MHz and 2.4 GHz, thus covering a range of possible combinations of achievable distance versus channel throughput. Moreover, the electrical and mechanical interface of both boards is identical. Therefore changing the radio module used in a system is straightforward. So it was reasoned that using the WiMODino™ platform would enable swapping off the boards when necessary to e.g. extend the distance from base to rover at the cost of corrections' update frequency.

In the current implementation, two WiMODino™ boards with WiMOD™ iM282A modules (IMST GmbH, 2018a) were used. The iM282A module operates in the highest frequency band of the series: 2.4 GHz, so it features higher data-rate than the other modules, but also the shortest range. For instance, an effective data-rate of 5.08 kbps at a distance of over 5.7 km line-of-sight was documented without any packet loss (IMST GmbH, 2018b).

As for the two WiMODino™ boards used in our system, one got corrections from a stationary base and forwarded them to the other via radio, which passed them onto the APX. The communication with both the base and the rover were realized with UARTs[30]. An "Arduino® shield"-style board for signal level translation and connector mating was designed. The programming of the radio link was done by Vu and Dogotari (2020). A simplified system diagram is shown in Figure 3.4.

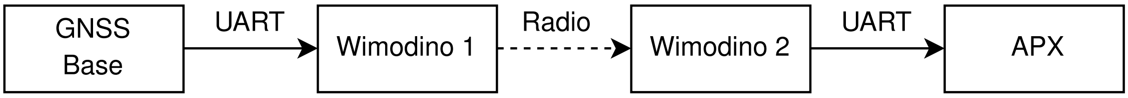


Figure 3.4: Diagram of RTK corrections via WiMOD radio link.

---

[29] virtual reference stations

[30] universal asynchronous receiver-transmitters

A successful proof-of-concept for the design using WiMODinos™ was realised. A computer connected to a VRS acted as the base and the APX achieved RTK fix based on the received corrections. Furthermore, the setup was tested with an available Emlid Reach RS2 base and even though corrections were received and interpreted by APX as being RTCM messages, no RTK fix was obtained. The likely explanation seems to be that the Reach RS2 did not provide the RTCM message 1008 to complement its corrections. This message, which refers to antenna description, seems to be required by Trimble receivers (SNIP Support, 2020; torriem, 2019). Moreover, torriem (2019) suggested the limitation could be overcome by injecting a dummy RTCM 1008 message. However, the proposed method has net been tested for the current work, because operation in LTE-denied environments has not been a necessity so far. Nonetheless, since a recent firmware update, the RS2 supports the RTCM message 1008 natively (Fursa, 2021), so using it with the current radio-link should function properly.

3.2.1.2. Puck interface board

The puck is a fairly ruggedized sensor with all its IO-connections provided through a three-meter long shielded cable. This cable contains an Ethernet interface, as well as further connections for supplying power to the puck and synchronizing it to a GNSS receiver (Velodyne LiDAR, Inc., 2019b, p- 40). By default, the manufacturer delivers the puck with a ready-to-use interface box. However, several potential shortcomings were identified, thus motivating the design of an own IFB. Fortunately, the schematics of the manufacturer's IFB were provided (Velodyne LiDAR, Inc., 2019b, p. 112) and could be used as a starting point for the new design.

Compared to the original interface board, mine has the following main differences:

- The power barrel connector was replaced with a more robust Molex connector;
- The screw terminal array for the sensor-side cable was also replaced with a Molex connector and a suitable plug was fitted to the cable;
- The cable shield was connected to ground. This is recommended by Velodyne LiDAR, Inc. (2018c, p. 15), but not always implemented *(compare Velodyne LiDAR, Inc., 2018b, p. 7 with Velodyne LiDAR, Inc., 2019b, p. 40)*;
- Magnetics, fuses and TVS diodes were added to the Ethernet interface;
- The circuitry for powering a GNSS/GPS device was discarded.

Furthermore the three-meter long cord was shortened to aid cable management. In order to keep the warranty valid, a waiver and instructions for safely cutting the cable were obtained from the puck's manufacturer (Velodyne LiDAR, Inc., 2018c).

The safety features of the original IFB (a 3-Ampere fuse and a TVS diode) were adopted into the new design.

Summarizing, the current IFB provides the puck with a standard Ethernet jack, an array of field-ready connectors and protection mechanisms for potentially vulnerable interfaces. An annotated photograph of the puck IFB is presented in Figure 3.5.

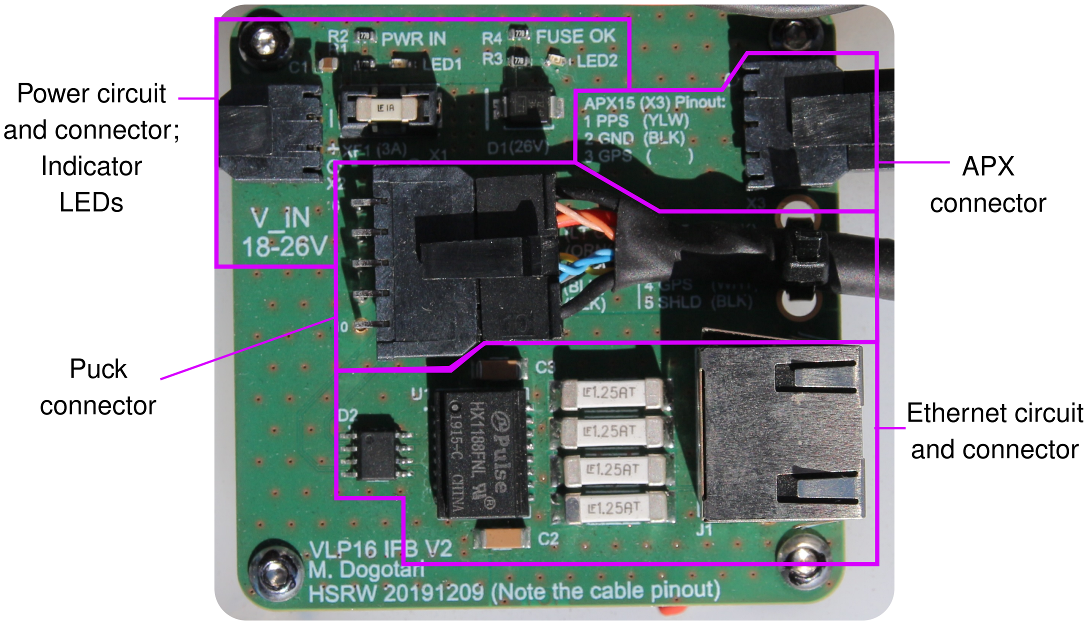

Power circuit and connector; Indicator LEDs
Puck connector
APX connector
Ethernet circuit and connector
VLP16 IFB V2
M. Dogotari
HSRW 20191209 (Note the cable pinout)

Figure 3.5: Photograph of the puck interface board.

3.2.1.3. RGB-Camera interface board

The RGB camera comes with a USB-C port that is used in conjunction with appropriate software (IDS GmbH, 2020b) for programming the camera and retrieving its images. In order to enable accurate synchronization of the images with the GNSS-IMU data, an interface board was developed.

Besides the USB interface, the camera provides an IO connector that exposes standard optocoupler-isolated trigger-in, flash-out functions. The connector also contains two non-isolated GPIOs[31] and a five-volts line to power external devices (IDS GmbH, 2020e). In order to identify where each photo was taken, an output signal had to be generated and sent to the APX, which would forward the time-stamped pose to the NUC. Normally, the existing flash-out function would be used for this purpose. However, to allow greater flexibility and potentially slightly more accurate time-stamping as a result of faster switching speeds, the GPIOs were used to generate this signal instead of the standard flash-out pins. Since the signal needs to travel over yet another cable to the APX IFB, it was further buffered with a standard logic IC to "hide" the cable's capacitance from the camera's driver IC. Moreover, the signals were designed to safely interface with the APX, which has stringent requirements on the acceptable voltage range (Applanix Corporation, 2019a).

While only one GPIO is strictly necessary in the current usage scenario, i.e. stamp each photo, both GPIOs were equipped with buffers in order to facilitate other potential usage scenarios, e.g. interleaving photos with different settings, synchronization with further hardware, etc. In addition, all the pins were routed to an extra connector. Also, a prototyping area was integrated to allow for future hardware modifications. Due to space constraints and in order to simplify swapping the camera, the IFB was actually implemented as two separate PCBs. One is rigidly mounted to the puck and contains the prototyping area, while the other contains the buffering circuitry and serves as the mounting plate for

---

[31] general-purpose input–outputs

the camera. The latter is connected to the camera with an open-ended cable (IDS GmbH, 2017), which was fitted with a Molex connector for ease of use. The two boards feature a circular design with mounting holes in a regular pattern that enables pointing the camera off-nadir if, for instance, a building's facade is being surveyed.

Due to its position under the UAV (see Figure 1.1), the puck has a lateral (cross-track) FOV[32] of about 240° (nadir $\pm 120^\circ$). The upper 120° of its lateral FOV (zenith $\pm 60^\circ$) are partially blocked by the UAV and its propellers. The side scans that the puck collects can thus be enhanced with photography by rotating the camera to the side. A photograph of the camera plus interface board assembly is shown in Figure 3.6.

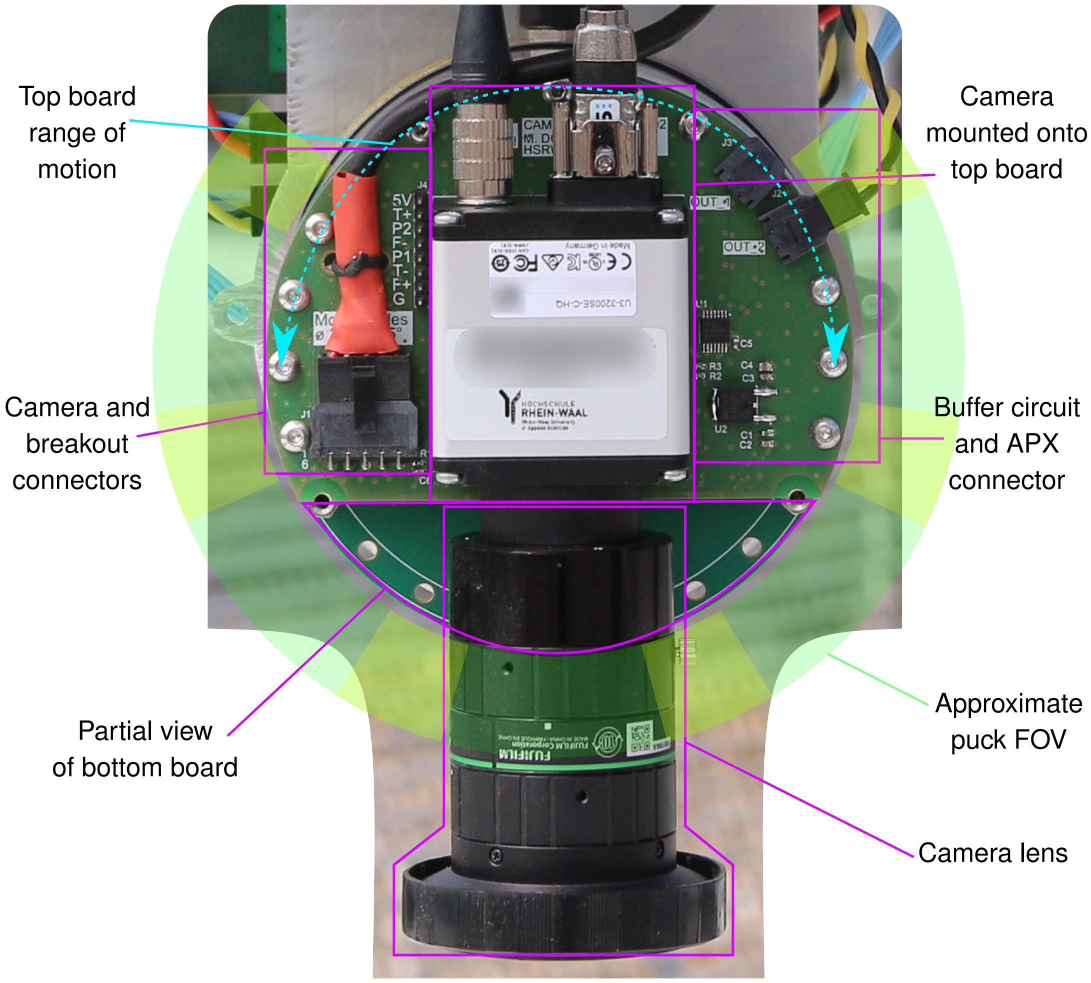

Top board range of motion
Camera mounted onto top board
Camera and breakout connectors
Buffer circuit and APX connector
Partial view of bottom board
Approximate puck FOV
Camera lens

Figure 3.6: Photograph of the camera's IFB. Note that the bottom board is only partially visible. The puck's approximate lateral FOV is also shown with a semi-transparent green-yellow pattern.

---

[32] field of view

3.2.2. POWER ELECTRONICS

The design of the power electronics consisted of the following stages:

- determining each component's requirements in terms of voltage and power;
- summarizing the needs of the whole system;
- finding appropriate sources;
- designing the necessary conversion steps and safety features;
- picking components and testing individual power stages;
- assembling and testing the system as a whole.

One key aspect of the power electronics design was minimization of voltage regulation stages required, so special care was given to determining the input voltage ranges of each component and using overlapping regions wherever possible.

The most power-hungry component was the NUC, which required at most 24 watts (W) at 15 volts (V) under full load with no peripherals attached (Prüm, 2019). Although its datasheet specified the required input voltage as 19 V $\pm 10\%$ (Intel Corporation, 2014), feeding it with 12–19 V is in fact acceptable Prüm (2017, p. 17).

The second hungriest component was the puck, due to its active laser and continuous motor operation. Its voltage requirements were stated by the manufacturer as 9–18 V (Velodyne LiDAR, Inc., 2018a). However, further inquiries revealed that it is safe to power the lidar unit with 9–32 V (Velodyne Acoustics, Inc., 2015; Velodyne LiDAR, Inc., 2019a). The puck's typical power draw is 8 W, with current surges of 3 amperes (A) at startup, independent of the supplied voltage (Velodyne LiDAR, Inc., 2018a).

Next, the APX accepts 9–30 V and consumes 3.5 W typically (Trimble Applanix, 2016) plus it needs to supply its active GNSS-antenna a further 0.1 W typically (Tallysman Inc., 2019). The Ethernet switch, the RGB-camera and the LTE dongle require 5 V and consume up to 8.8 W combined (Conrad Electronic SE, 2016; IDS GmbH, 2020d; Huawei Technologies Co., Ltd., 2015). These needed to be powered through the NUC's USB ports, so the NUC's total power demand rose to 33 W peak.

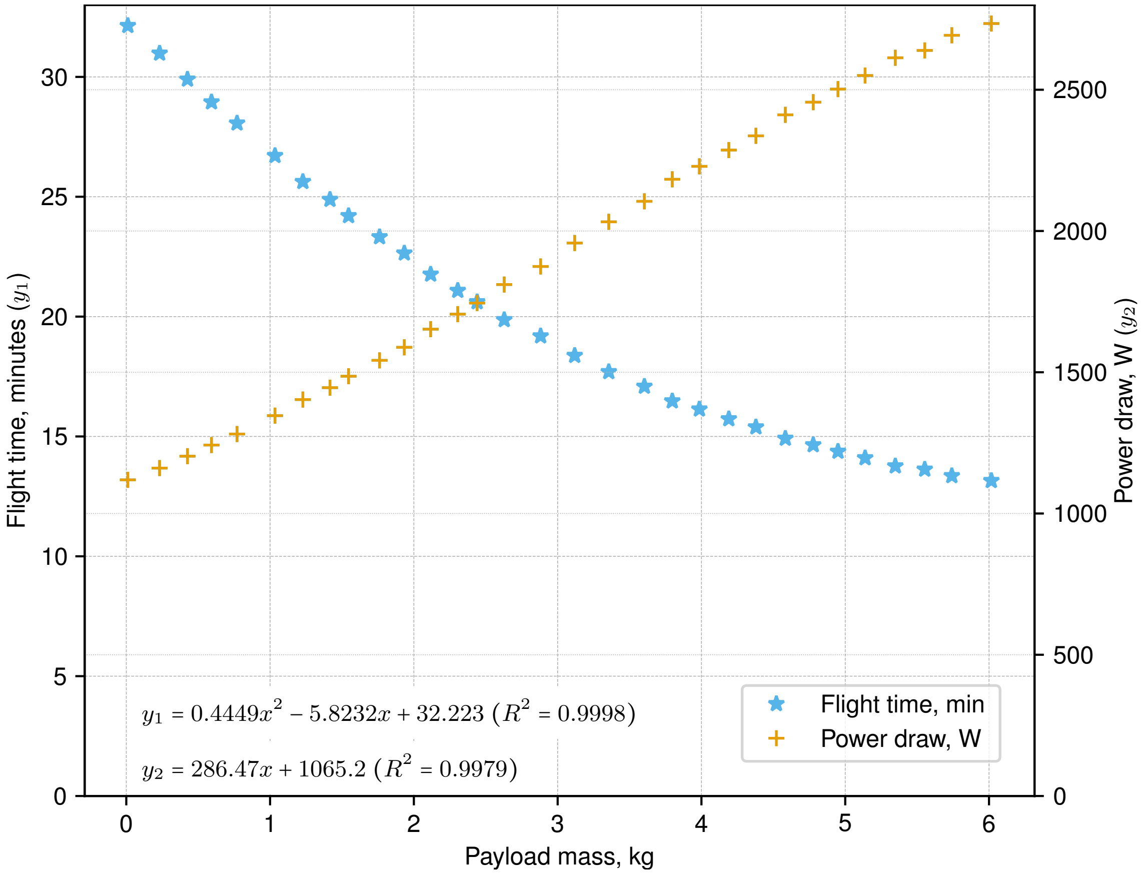
Summarizing, the system needed about 33 W at 12–19 V for the NUC with its peripherals and another 12 W at 9–30 V for the puck and the APX. Regarding current, 5–6 A had to be sustained for short times when starting the laser and booting the NUC.

Once the requirements of the system have been identified, an appropriate power source had to be found. The first instinct was to power the system from the UAV, but it is important not to deplete the UAV's accumulators too fast, as it would negatively impact the flight time. The UAV's manufacturer published a graph relating the payload mass to the platform's flight time (DJI, n.d.).[33] These data were used to derive the power consumption's dependency on the payload's mass. A linear function with a slope of ≈ 286 W/kg was obtained ($R^2 = 0.9979$, see Figure 3.7).

It takes ≈ 286 W to carry each kilogram of payload. The system weighs 3.3 kg without any power source, so it takes ≈ 945 W for the UAV to carry it, but just 45 W to power it. Given this argument, it is reasonable to power the lidar system with the UAV's accumulators

---

[33] The data most likely refers to hovering in a no-wind environment at 10 m above sea level (DJI, 2018b).



Source: Own work, based on DJI (n.d.)

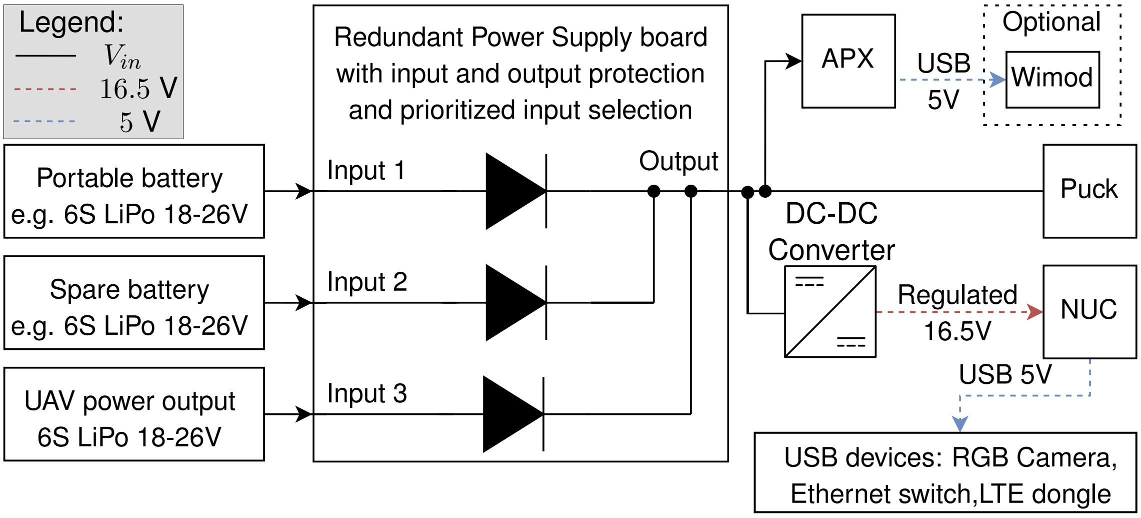
Figure 3.7: Flight time and power draw for DJI M600 Pro with TB47S accumulators. *(Conditions from Footnote 33 apply.)* The flight time versus payload mass was retrieved from a graph on the official DJI page through a graph digitization process. The power draw was obtained by dividing the stated accumulator capacity by the retrieved flight time. The high R^2 values indicate that it is appropriate to use the extracted formulas as an approximation of the "true" behaviour as long as the payload mass stays within the specified range ($[0, 6]$ kg).

instead of carrying an extra accumulator. Another way to picture it is that the 45 W required by the system are analogous to $45\cancel{W}/286\frac{\cancel{W}}{\text{kg}} \approx 0.16\text{kg}$ of extra payload and as long as the system's mass stays above 0.16 kg, powering it from the UAV's accumulators is most effective. The only sensible way to increase the flight time would be to considerably reduce the system's weight.

The UAV features a power port that directly corresponds to its accumulators' voltage of $18 - 26$ V and provides a maximum current of 10 A (DJI, 2017, p. 10). Since the voltage range fits within the requirements for the APX and the puck, it was decided to power these directly from the UAV. But because it significantly exceeds the guaranteed safe voltage for the NUC, a step-down regulator was added to the circuit.

That said, on a typical flight campaign the system would spend considerably more time on ground for various checks, settings and troubleshooting than in the air for data acquisition. Therefore it is desirable to power the system from another source while on ground and seamlessly switch between it and the UAV's accumulators while airborne. Two real power sources shall not be connected directly in parallel, as they may be damaged by reverse currents. A simple circuit using (Schottky) diodes at the accumulators' output would protect both sources, but it would be terribly inefficient. Instead, Lucieer's recommendation (2018) was implemented: an appropriate IC that performs an "ideal diode" function thus enabling the use of multiple power sources was identified and integrated in the power electronics design. The chosen IC also has built-in protection from reverse polarity, as well as over- and undervoltage conditions (Linear Technology Corporation, 2012). Further mechanisms to protect both the UAV and the system from a range of power failure modes such as transient voltages, short circuits and inrush currents were implemented. The simplified diagram of the power electronics is presented in Figure 3.8 and the complete schematic circuit – in Appendix A.

a) Schematic diagram of power system


Legend:
V_{in}
16.5 V
5 V
Portable battery e.g. 6S LiPo 18-26V
Spare battery e.g. 6S LiPo 18-26V
UAV power output 6S LiPo 18-26V
Redundant Power Supply board with input and output protection and prioritized input selection
Input 1
Input 2
Input 3
Output
APX
USB 5V
Optional
Wimod
Puck
DC-DC Converter
Regulated 16.5V
NUC
USB 5V
USB devices: RGB Camera, Ethernet switch,LTE dongle

b) Photograph of the Redundant Power Supply Board (DC-DC converter board not visible below)

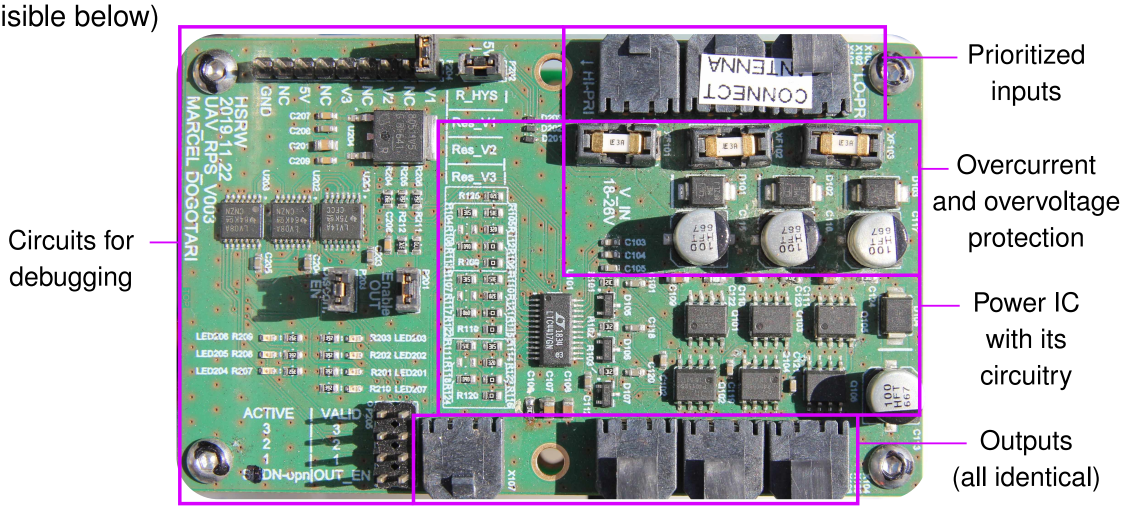


Figure 3.8: Diagram and photograph of the power electronics. Only the redundant power supply board (RPS) is shown in the photo, as the DC-DC converter is directly under it. The RPS implements input validation for detection of over- and undervoltage conditions. The validation is done by the IC and the limits are set via external resistors. Out of the 3 inputs, input 1 has the highest priority. As long as its voltage stays within the accepted limits, it is connected via a low-impedance path to the output. If its voltage is not within the limits – input 2 is used and so on. The UAV has the lowest priority, so that its battery is conserved while on the ground. Typically the inputs 1 (and 2 if used) are disconnected by the operator just before starting the UAV motors and are reconnected after landing and turning off the motors.

3.3. SOFTWARE

The software developed for this thesis divides, according to its function, into software for (in-flight) data acquisition on one hand and data (post-)processing on the other. The sole mission of the acquisition component is ensuring that all data is saved to disk while the UAV is airborne and computing resources are at a premium. The more computationally-intensive creation of point clouds takes place later in the processing step and normally on other computers. Additionally, a non-core component for in-flight data visualization has been programmed for easier in-field operation. The following sections describe some of the more important functions and processes. The entire codebase is available on GitHub (Dogotari & Rostalski, 2021).

3.3.1. DATA ACQUISITION

High-level overview A main script written in `Python 3` (Version $\geq$ 3.5.0) is started on the NUC before every flight and it deals with recording the data streams coming from the three sensors: puck, APX and camera.

The main script needs to be started by the operator, who either specifies a few command-line arguments, or the defaults are used. These tunable parameters are: custom folder name (after the mandatory mission number), flying height, maximum acceptable blur, desired RGB overlap and maximum flying time. With the exception of the folder name, the arguments refer to camera operation. The main script calculates the maximum number of images based on the flying time and passes it as an argument to the camera acquisition program, which terminates upon reaching this limit. This upper limit on the number of images to be acquired was imposed due to the space usage of the camera data, which is significantly larger than that of the other two sensors. Nonetheless, a conservative upper limit of 30 minutes, which is longer than the normal flight duration is usually used. This way, images are acquired during the entire flight duration, but the acquisition ceases shortly after landing. This is useful if the operator cannot attend to the UAV immediately and disk space is at a premium.

Figure 3.9 depicts a typical data acquisition campaign, often consisting of multiple flights.

For most of the campaign duration, a laptop is connected to the system via Ethernet to enable access to the sensors' web-interfaces as well as to the NUC's SSH[34]. Beginning and stopping data acquisition is done from a laptop computer that uses SSH to log onto the NUC and start/terminate scripts. One step that is not shown in Figure 3.9, but is implied, is disconnecting the controlling laptop before every flight and reconnecting it after each landing. Usually, the external accumulator is plugged simultaneously with the Ethernet cable. The same applies for unplugging: Ethernet and accumulator are disconnected at the same time. This way, the duration that the system feeds off the UAV's batteries is minimized. To ensure collection of consistent data, the flight missions are planned and executed with a mission planning software: the DJI GS Pro (DJI, 2018a).

---

[34] secure shell

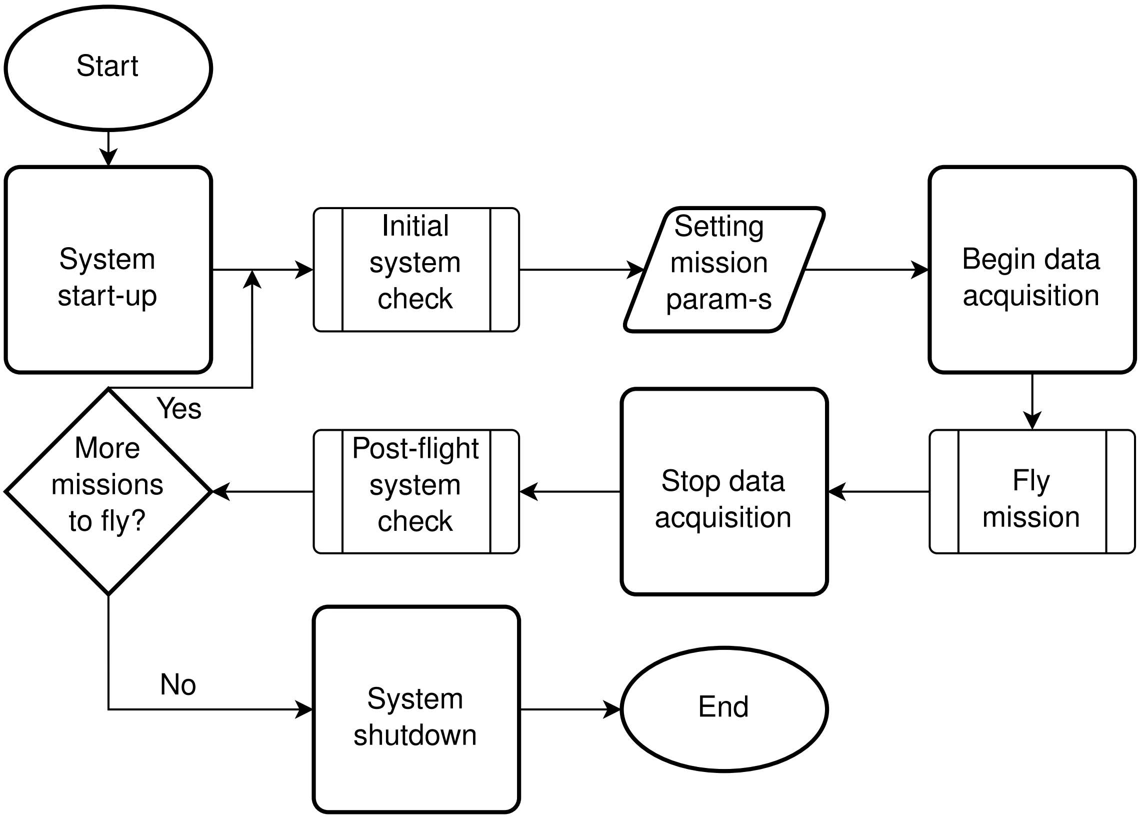


Figure 3.9: Typical flight campaign procedure. For the initial check, the operator has to ensure that: (1) the puck is receiving APX's signals, (2) the APX is getting RTK corrections and (3) the NUC has at least 50 GB of disk space available. The post-flight check ensures that: (1) data has been written to disk and (2) the main `Python` script and its children have been terminated properly.

Puck and APX data The puck's data comes over UDP[35] (Velodyne LiDAR, Inc., 2019b). The main script simply starts instances of `tcpdump` (The Tcpdump Group, 2020) that listen to the preconfigured ports and write the captured packets to `PCAP` files. The puck's data stream contains two kinds of packets: Data and Position (Velodyne LiDAR, Inc., 2019b). These are delivered through distinct UDP ports. In the current setup, the two kinds of packets are written to separate files for easier processing (see Section 3.3.2.1). Even though UDP does not guarantee data integrity, the throughput of the network is well within its capabilities and no issues, such as lost packets, were ever detected.

Due to its simplicity, the same approach of streaming data over UDP and capturing it with `tcpdump` was chosen for the APX. A set of NMEA sentences[36] containing the full pose of the puck are transmitted from the APX to the NUC via UDP at 100 Hz. The APX can provide the same information either as text (Applanix Corporation, 2019b, pp. 32-36) or binary (Applanix Corporation, 2019b, pp. 36-42) data. While the latter uses less network bandwidth and therefore space on the disk, and in the long run is in fact easier to parse in software, the former was chosen because it is easier to read for humans. Additionally, another set of timestamped messages also containing the full pose are transmitted whenever the camera takes an image and triggers the APX. Each of these streams are captured by their own `tcpdump` instances, which are also started from the main script.

---

[35] user datagram protocol

[36] For a brief explanation of NMEA sentences, see Appendix C.1

The sensors and the `tcpdump` instances are set so that each kind of data (puck ranging packets, APX position data, APX trigger data, etc.) uses their own UDP port and is captured by `tcpdump` to their own files with descriptive names. To avoid the files becoming exceedingly large, `tcpdump` creates new ones whenever the files reach a certain size. The threshold size is set for each type of data individually and aims to create one new file about every minute. This way, in the unlikely event of file corruption not all data would be lost. Simple incrementing counters are used to arrange the files in the correct order.

The `tcpdump` instances dealing with the puck and APX data are closed only after landing, when the operator manually terminates the main script. In contrast to the camera images, the data of the puck and the APX does not require much disk space so the penalty for terminating the acquisition several minutes after landing the UAV is low.

Camera data In contrast to the APX and puck, where no customization is needed once the communication ports have been set up, the RGB camera requires setting some parameters before data acquisition. A separate program written in `C++` and using the `IDS Peak SDK`[37] (IDS GmbH, 2020b) deals only with the camera and is invoked by the main (`Python 3`) script. This camera program has to be called with the following command-line arguments: target frame rate, maximum number of images to acquire, pixel format and maximum exposure time. The frame rate and maximum exposure time are calculated by the main script based on user input regarding the flying altitude, maximum acceptable blur, flying speed and desired overlap. Besides user input, a text file containing the camera and lens properties is used to derive these values. The pixel format and maximum number of images are chosen by the main script based on camera capabilities, normal flight duration and reasonable disk usage. After making the required settings, the acquisition is started and the camera program saves the images received from the camera.

The camera uses a Bayer filter. So demosaicing (aka debayering) needs to be applied in order to obtain RGB images. Even though demosaicing can be done both in camera hardware and in software on the SBC (IDS GmbH, 2021c, 2020c), it is not executed in-flight in order to preserve image quality, disk space and computing power. The image data is saved with its full twelve-bit colour depth whenever possible. However, when especially large overlap is desired, or the UAV is flying low and fast, the image quantity increases dramatically, so the main `Python` script limits the colour depth to either ten or eight bits per pixel to avoid capturing more than 50 GB of data per flight. While somewhat arbitrary, this limit proved safe in our setup and so far all the images were acquired at twelve-bit colour depth. It is also noteworthy that the space saving when using ten and twelve bits per pixel is further aided by saving the data in packed formats, as defined by EMVA (2019, pp. 12, 33–34).

---

[37] software development kit

3.3.2. DATA PROCESSING

The software developed for data processing takes the files captured in Section 3.3.1 and transforms them into usable formats.

In the case of the puck and APX, the final output files are `LAZ` point clouds. `LAZ` is a losslessly compressed variant of the `LAS` file format (ASPRS, 2019). Both formats are open and free to use and an open-source tool for converting between them is also available: LASzip (Isenburg, 2013). Another advantage when compared to ascii[38] file formats is their binary nature, which enables far smaller file sizes and faster processing. Normally tools that support `LAS` files, also work fine with `LAZ` ones, so the latter were chosen for their even smaller size. The lidar processing steps are outlined in Section 3.3.2.1.

The RGB images are processed up to `TIFF` files (Adobe, 1992). This format was chosen due to it wide-spread nature and open source support (Leffler et al., 2021). The respective workflow is described in the Section 3.3.2.2.

3.3.2.1. Lidar processing

The point cloud generation pipeline consists of two stages. First, ascii point clouds are created from the puck and APX data. This stage uses mainly in-house developed tools written in `Python 3` (Version ≥ 3.8.0). Next, the ascii clouds are converted to `LAZ` files using `LAStools` (Isenburg, 2021).

Merging `PCAP` files The first processing step consists of aggregating the files corresponding to each data stream. The `mergecap` tool from `Wireshark` (Combs et al., 2020) is used to merge the multiple separated files into one `PCAP` file per data stream and flight.

APX `PCAP` files to `CSV` In the next processing step the APX observations are parsed. Initially, these are stored in `PCAP` files, where the contents of the UDP packets are NMEA sentences. For easier access to the data, the contents of these sentences are saved to `CSV`[39] files. Because sometimes the APX splits NMEA sentences across packets, a state machine is used to ensure packets are complete before parsing them. Even though multiple fields are recorded in the output files, only the following ones are used in later processing steps: easting and northing in a projected CRS, geoid elevation, roll, pitch, heading and time of the GNSS fix. This last piece of information is crucial, as it is later used to align the APX and puck data-streams. The easting and northing are obtained by reprojecting the longitude and latitude output by APX. This is done using the `Python` bindings for `GDAL`[40] (GDAL/OGR contributors, 2021).

Segmentation of flight lines Next, the `CSV` files are used to segment each flight into individual flight lines. The aim is to extract sections of the flight where the aircraft moves in a straight line with a rather constant speed, which usually correspond to the sections flown by the autopilot. The start and end of these lines are then used to “cut” the files containing

---

[38] ASCII (American Standard Code for Information Interchange) is a standard for encoding text. In the context of point clouds, it means the data (point coordinates and other attributes) is human-readable text.

[39] comma-separated values

[40] geospatial data abstraction library

the puck ranging packets into smaller files, one flight-line each. To determine which parts of a flight mission belong to a straight line, first each observation in the recorded data is evaluated according to a few requirements. Then the continuous segments containing only observations satisfying these condition are identified and those with a duration longer than a certain threshold are considered to be the flight lines. “Cutting” the `PCAP` files is done with the `editcap` tool from `Wireshark`.

In the current work, an observation is “good”, if: the UAV (1) flies at an altitude higher than a certain threshold, (2) does not turn too quickly and (3) its velocity is within a particular range. These are conditions that one would expect to be met during a planned-mission flight. A visual aid of how the parameters are used to determine which observations might belong to flight lines is shown in Figure 3.10.

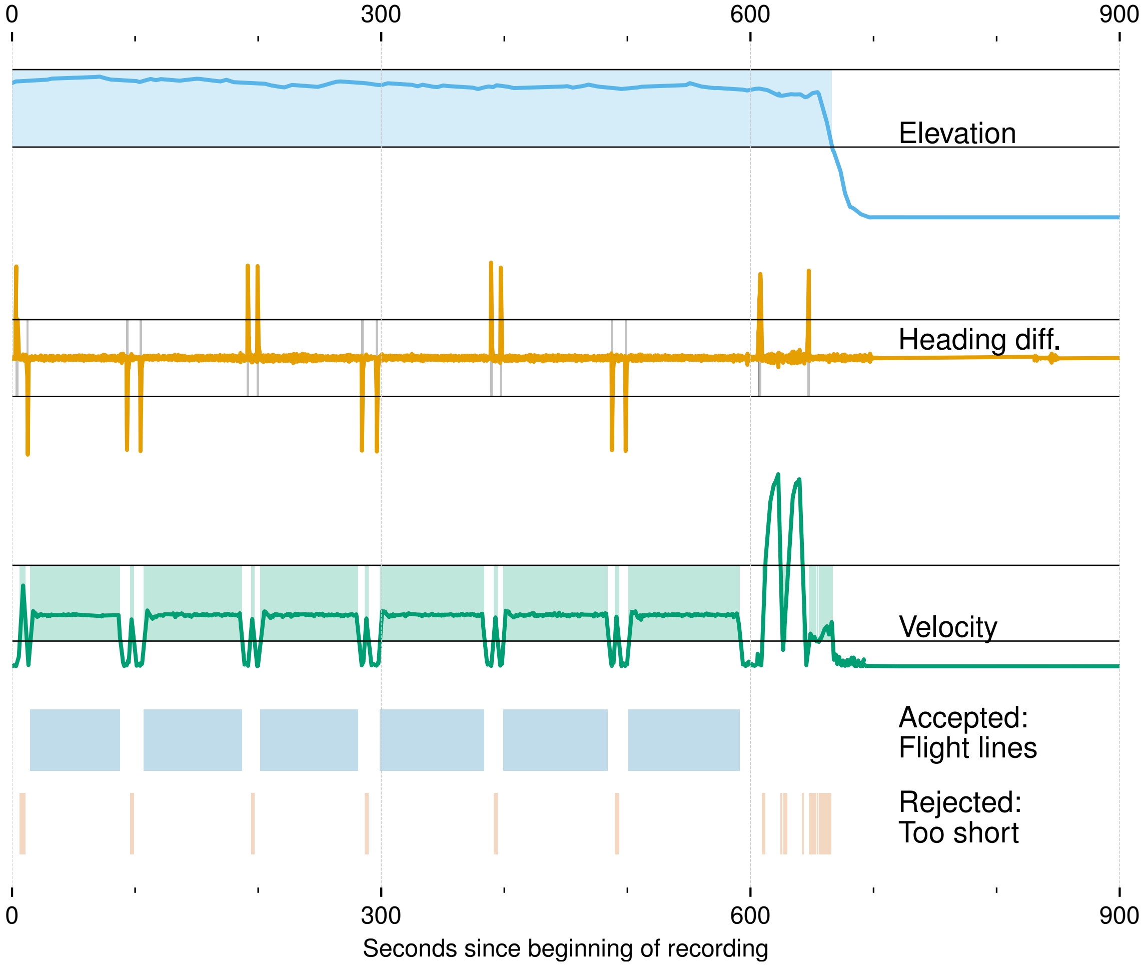


Figure 3.10: Use of flying height, heading difference and velocity to segment flight lines. Data acquired during 15 minutes of flight time of the mission 2 in the Vechta flight campaign. Values for Elevation, Heading difference and Velocity , as well as the valid ranges of the parameters are shown, followed by valid and invalid flight lines. All y-scales are linear.

In Figure 3.10, the three curves represent the raw Elevation, Heading difference and Velocity respectively. The three pairs of horizontal lines accompanying them show the range where these variables are “good”. The light-shaded areas supplementing the Elevation and Velocity curves show regions (in time), where the two variables are within the acceptable range. The grey shading on the Heading curve is inverted: it shows discontinuities in the Heading’s “goodness”. The six light-blue patches under the velocity curve show flight segments where all three parameters are good. Because these segments last longer than a specific threshold, they are interpreted as a continuous flight line each. The last row of light-orange patches shows segments of the flight that were flight-lines candidates but were discarded because their duration was too short. The following paragraphs show how the “good” ranges of each parameter were quantified and identified.

The elevation threshold depends on the parameter h (h for *height*) and is computed as a value that varies between the minimum elevation in the dataset when $h = 0$ and the maximum one when $h = 1$. These extremes normally correspond to the take-off point and to the target altitude, so this parameter allows to easily disregard data captured too low.

To assess the second condition, the forward finite difference (Δ) of the aircraft’s heading ($y(t)$) (y for *yaw*) is computed as defined in Equation 3.9.

$$\Delta_{\tau}\left[y\right](t) = y\left(t + \tau\right) - y\left(t\right) \tag{3.9}$$

Where τ represents the sample spacing and in the current context it is ten milliseconds—the time resolution of the APX data. Since the heading relative to north varies between $0°$ and $360°$, naturally there are discontinuities when the UAV’s nose passes an imaginary line pointing north. These would be picked up by the algorithm and interpreted as a break in the flight line. Data collected on a north-bound course would be susceptible to this kind of error. To prevent detection of false flight-line breaks, the heading is first made continuous by adding $\pm 360°$ when the northward line is crossed. To get back to the original heading, one would compute the continuous heading modulo $360°$. From the perspective of the heading, an observation is said to be good if its heading’s forward finite difference is within y standard deviations from the median of all headings’ finite differences.

The next condition to be checked is whether the velocity does not deviate too much from the median. This is implemented with a factor $v > 1$ (v for *velocity*), such that all velocities contained between median velocity divided by v and median velocity multiplied by v are considered “good”. This simple thresholding keeps the most frequent velocity—which is expected to be the one set in the mission planner—and (usually small) deviations from it.

The results of the segmentation based on elevation, heading continuity and velocity are aggregated with a logical-and function. Then continuous segments are extracted from the observations satisfying all three criteria. Next, the segments with a duration longer than s seconds are kept.

In contrast with the rest of the point cloud generation procedure, that always runs automatically, this step can also be run interactively. The mode is chosen by a command-line argument passed to the script. In automatic mode, the default values for h, y, v and s are used. When using the interactive mode, the user visually assesses the result obtained with the default parameters and can adjust one or more of them via the command line until an acceptable result is obtained.

Whichever mode is used, a plot of the lines is saved to disk after each iteration. The parameters used are saved in the title of the figure, so the process is reproducible. An example of such a plot is shown in Figure 3.11.

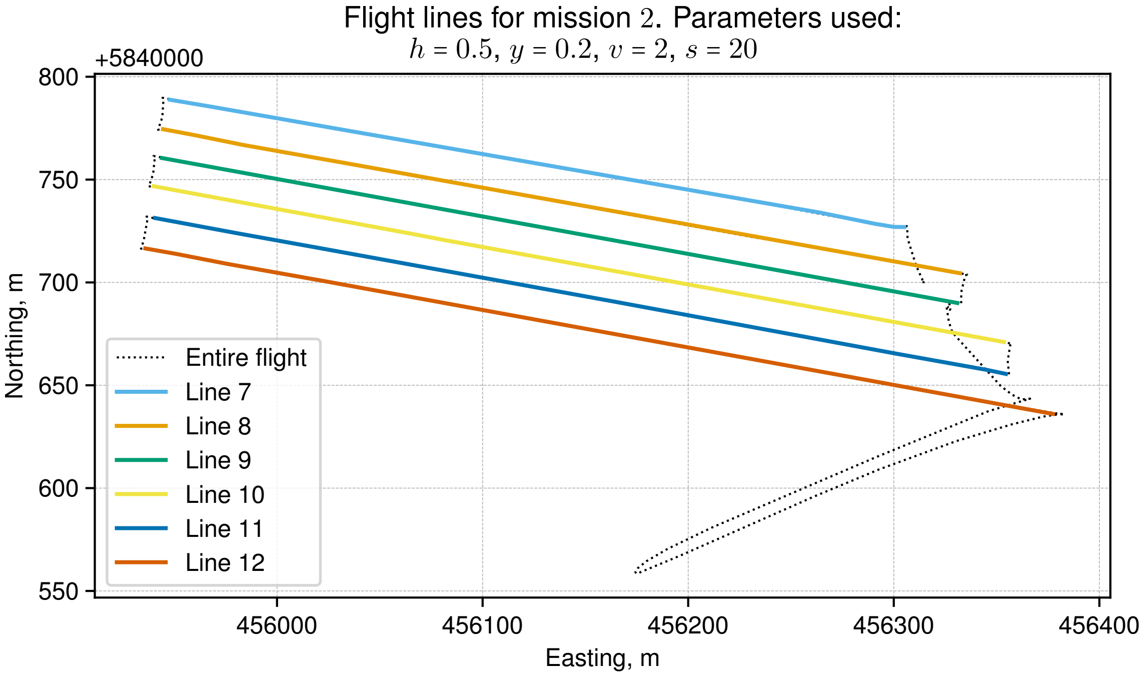


Figure 3.11: Result of flight-lines segmentation: Mission 2 from the Vechta campaign. The Easting and Northing are given in the WGS 84 / UTM zone 32N CRS (EPSG:32632). h, y, v and s refer to the “elevation (*height*) threshold”, “heading (*yaw*) deviation tolerance”, “*velocity* factor” and “time parameter (*seconds*)” that were used for the current segmentation.

The flight-line segmentation script is run as a batch process for all flight missions at once. However, in the interactive mode, the parameters are adjusted on a mission level. In some cases, multiple iterations per mission might be needed for getting the right values, so multiple plots are saved. Both in automatic and interactive mode a consistent numbering scheme $(1, 2, \ldots, N)$ for all the flight-lines of a flight campaign is used. Finally, after the user finds setting fitting each mission, the `editcap` utility from `Wireshark` is used to split the puck files into flight lines. The tool is called automatically with the start and end points of the flight-lines from the *last* plot of each mission. This way, no extra files are created for the unsuccessful iterations preceding the last one. The resulting files are named `line_n.pcap`, where `n` $\in [1, 2, \ldots, N]$. This naming scheme is preserved throughout the successive processing stages.

Puck `PCAP` files to `LAZ` point clouds Finally the puck data files, which were divided by flight line in the previous step, are combined with the APX observations to derive the point clouds. This is accomplished by a `Python 3` script that fuses the data from both sensors to create georeferenced ascii point clouds, then automatically calls `LAStools` (Isenburg, 2021) to convert them to the `LAZ` format and remove obvious noise.

The puck’s readings are georeferenced with the APX positioning data by performing the calculations outlined in Section 3.1. The APX data is read from the prepared `CSV` files with the help of the `pandas` package (Reback et al., 2021). The data within the `PCAP` files is accessed with the `Scapy` package (Biondi et al., 2020). To interpret the lidar readings, a parser based on the puck data specification (Velodyne LiDAR, Inc., 2019b, pp. 53-68) was implemented.

Once the data is read into the current script, it is converted to `NumPy` arrays. The `NumPy` package is used extensively throughout the codebase, as it offers powerful tools for efficient and vectorized calculations. Moreover it features an intuitive and comprehensive syntax for seamless manipulation of N-dimensional arrays (Harris et al., 2020). Among the first things, a few simple checks are done and data is decimated considerably: all returns that are missing or too close (likely because of hitting the UAV) are discarded.

The calculations needed for georeferencing the lidar returns have to be performed for each lidar measurement (see Section 3.1). However, there are ≈ 290 thousand laser returns per second, and only 100 puck pose estimates. Therefore, the `spatial.interp1d` method from the `SciPy` package (Virtanen et al., 2020) is used to train interpolation functions on the APX data. They take a UTC-timestamp as input and provide the full pose of the puck (roll, pitch, true heading, easting, northing and elevation) as output. By calculating the precise timestamp of each lidar return and feeding it into the six interpolation functions, full pose estimates are obtained for each point.

The structure of the data arrays is such that the operations defined in Section 3.1 can be performed on all laser returns simultaneously. For instance, to rotate a set of coordinates $\mathbf{p_0}, \mathbf{p_1}, \ldots, \mathbf{p_n}$ around the axis y by their corresponding angles $\theta_0, \theta_1, \ldots, \theta_n$, the `numpy.matmul` routine is used. This function performs matrix multiplication, even for stacked matrices and vectors, as is the case in this paper. Figure 3.12 shows how the data is arranged conceptually in stacked matrices so that `NumPy` performs vectorized operations on them.

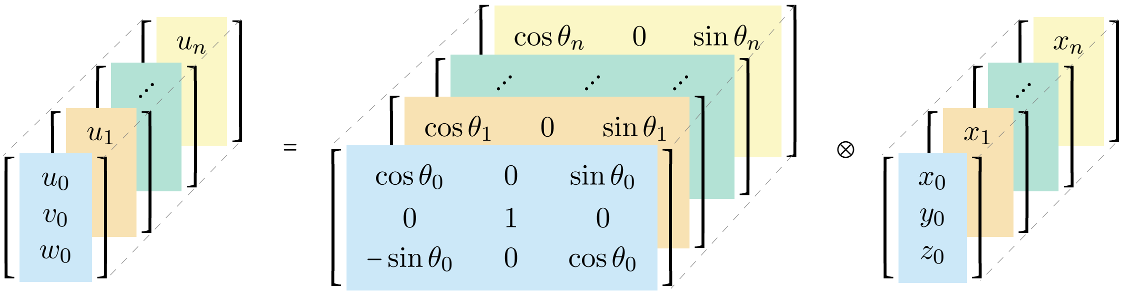

u_n
$\vdots$
u_1
u_0
v_0
w_0
=
$\cos\theta_n$ 0 $\sin\theta_n$
$\cos\theta_1$ 0 $\sin\theta_1$
$\cos\theta_0$ 0 $\sin\theta_0$
0 1 0
$-\sin\theta_0$ 0 $\cos\theta_0$
$\otimes$
x_n
x_1
x_0
y_0
z_0

Figure 3.12: Visualisation of stacked-matrices multiplication. In this example, the points $\mathbf{p_0}, \mathbf{p_1}, \ldots, \mathbf{p_n}$ with coordinates $(x_0, y_0, z_0), (x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)$ shown on the right are multiplied by their corresponding rotation matrices about the y axis (shown in the middle) to obtain the new points $\mathbf{q_0}, \mathbf{q_1}, \ldots, \mathbf{q_n}$ with coordinates $(u_0, v_0, w_0), (u_1, v_1, w_1), \ldots, (u_n, v_n, w_n)$ shown on the left. The colours of each slice in the data arrays show the scope of individual multiplications. The symbol '$\otimes$' is loosely used to represent parallel multiplication of multiple matrix-vector pairs.

In practice, the calculation is not performed on *all* points at once. Instead a loop is used to iterate over the data and process about ten seconds of observations at once. This way all the intermediate array are created when needed and have a much smaller memory footprint. The iterative processing still takes advantage of the vectorized calculations, georeferencing over seven million points per pass, but ensures the algorithm can still run on computers with modest RAM[41].

---

[41] random-access memory

After every loop, the results are saved to intermediate text files. When all returns from a flight line are processed, the intermediate files are merged into one text file per flight-line. Next the `txt2las`, `lasinfo`, `las2las` and `lasnoise` command-line utilities from `LAStools` (Isenburg, 2021) are called to convert the text point clouds to the `LAZ` format, ensure that the files are up to standard (ASPRS, 2019), the correct CRS is saved in the metadata and obvious noise (such as lone returns from birds) is removed.

Besides crucial information, i.e. coordinates for every point, `LAZ` files may contain a myriad of other fields, such as GPS timestamp, angle of laser beam from azimuth (across the flight path) and return intensity (ASPRS, 2019). In order to maximize the information content of the files and potentially enable sophisticated workflows, these fields are also populated in the output files. Additionally, a few non-standard properties, such as angle along track, are calculated and stored in the "extra bytes" fields.

Such information may prove useful in certain applications. Since it does not cost much computing power or time, it is included in the result. However the `LAS` specification also contains fields that do not really apply to UAV-borne sensors with $360°$ FOV. One such property is the "Scan Direction Flag" which is only meaningful for lidar units with oscillating mirrors. The fields that do not apply to the puck are not populated. Moreover, even though the `LAS` format has a field named "Scanner channel", it cannot be used to distinguish between the puck's 16 channels, as the field uses two bits, and as a result can only hold four distinct values. Instead a new field "LaserID", which is consistent with the manufacturer numbering scheme, is created.

Once the `LAZ` point clouds are produced and the most obvious noise is filtered out, the processing of the lidar data is concluded. The result is a collection of `LAZ` point clouds, divided by flight line, which can be fed into normal lidar processing workflows.

3.3.2.2. Processing of RGB data

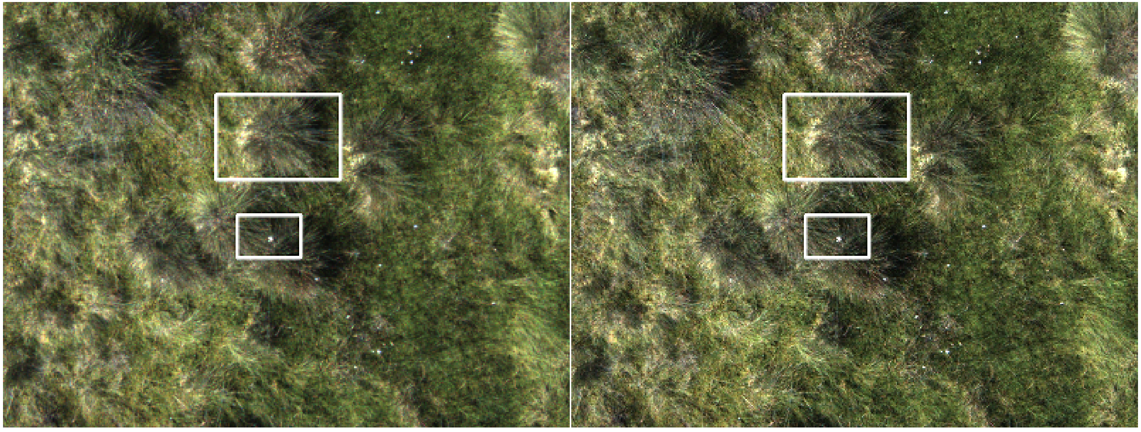
As explained in Section 3.3.1, the camera acquires mosaicked images, which are stored as raw bytes. This section discusses how these raw files are transformed into a usable format, such as `TIFF`. Most processing is done with `Python 3` (Version ≥ 3.8.0). Two workflows have been developed for demosaicing the raw images. One relies solely on the `OpenCV` library (Bradski, 2000). The other chains it with own software based on the `IDS Peak SDK` (IDS GmbH, 2020b). The output of both workflows is `TIFF` files, whose metadata is then added with `ExifTool` (Harvey, 2019).

In the first workflow, the script simply reads the raw files into `NumPy` arrays and uses `OpenCV`'s built-in `demosaicing` function. Then it usually truncates the colour depth to eight bits. Finally the same library's image writer is used to save the files to disk.

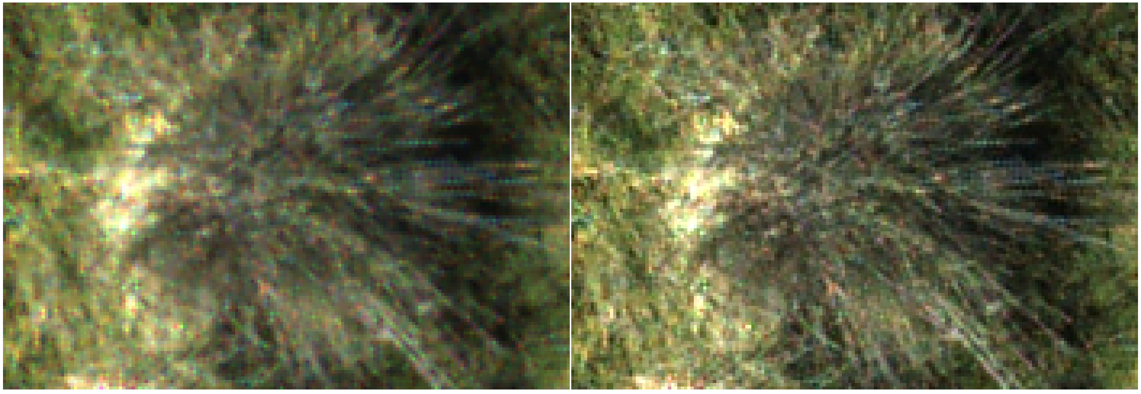
This approach proved to lead to poor image quality. While not apparent at first sight, zooming in until individual pixels became visible revealed images that were blurry and riddled with noise. To establish whether these issues came from the images themselves or were introduced by the demosaicing algorithm, the camera manufacturer's `SDK` (IDS GmbH, 2020b) was used to process a few images. An example where an image was processed with both libraries is provided in the Figure 3.13.

(a) Demosaicked with **OpenCV** (b) Demosaicked with **IDS SDK**

(1) Original crop. 800x600 pixels.



(2) Crop of upper rectangle. 176x120 pixels.



(3) Crop of lower rectangle. 90x60 pixels.

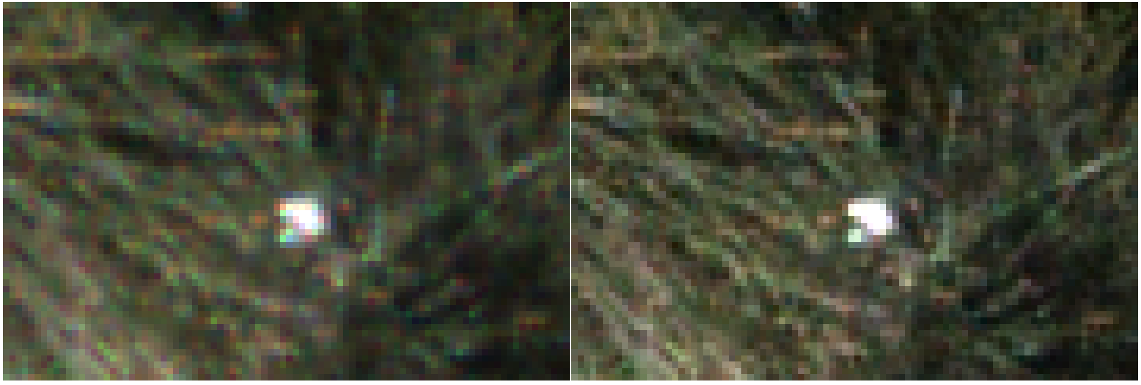


Figure 3.13: Demosaicing result: `OpenCV` vs. `IDS SDK`. Left side (a.1, a.2 & a.3): `OpenCV`. Right side (b.1, b.2 & b.3): `IDS SDK`. Top: crop out of an original image ($\approx 20\%$ of the initial resolution). Middle: crop out of a tussock (delineated by a white rectangle in the top image). Bottom: crop out of a flower (delineated by the lower white rectangle in the top image).

Looking at Figure 3.13,[42] practically no difference can be detected between the outputs of the two libraries, as long as not zooming in too much. However, when zooming in excessively, more defects are visible in the version produced with `OpenCV`, compared to that output by the `IDS SDK`. Both libraries have a fair amount of noise, most of which is likely due to the high gain settings (ISO ≈ 600). Nonetheless, it seems to be more pronounced in the version produced with `OpenCV`. Some pixels look extremely red or blue, both of which are very far from the true colour of the scene. The edges look noisier

---

[42] It is recommended to view the electronic version of this Figure, as no faithful reproduction of the artefacts can be guaranteed in print.

overall, but the versions on the left seem even more extreme. Furthermore, the border of the bright patch in the bottom row looks especially full of imperfections. To the naked eye, the images on the left, particularly the one in the middle row, also seem blurrier.

Based purely on this qualitative assessment, it was concluded that the demosaicing algorithm used *does* make a difference and the one employed by the camera manufacturer's `SDK` seemed to perform better for this set of images. A similar comparison was made between the `IDS SDK` and `OpenCV`'s edge-aware demosaicing. The latter did not yield better results than the default variant. Consequently, in order to preserve higher image quality, a second demosaicing workflow using the `IDS SDK` was implemented.

The proposed workflow was very simple: using the `IPL`[43] component of the `IDS SDK`, read the raw image data into a buffer, perform the demosaicing and save the result to a normal image file. However, this library only supports reading data from `PNG`, `BMP` or `JPEG` files. Therefore, an initial step of saving the image as a greyscale `PNG` with `OpenCV` was needed. Moreover, it was determined that the library only supports `PNG` files containing a specific `PNG` tag that so far could not be written with `OpenCV` or `ExifTool` – a library specialized in editing image metadata (Harvey, 2019).

The problematic tag (`sBIT`) specifies the number of significant bits used to represent the colour data. For example, images with 12 bits colour depth have to be stored as the upper 12 bits in a normal 16 bit image. This is because colour depths between 9 and 15 bits may only be represented as 16 bits, i.e. two bytes (W3C, 2003, Table 11.1) The `sBIT` tag can then be used to specify whether the first nine, ten, or the whole 16 bits of a two-byte integer are significant. For `PNG` viewers, the absence of this tag does not seem to be a problem. However, in order to conform to the requirements of the `IDS IPL` (IDS GmbH, 2020a), the tag had to be used. Because the tag could not be set with traditional metadata editing techniques, `TweakPNG` (Summers, 2014) was used to generate a correct tag and find out where in the file it belongs. This information was then used to insert the corresponding bytes into the files produced by `OpenCV`.

Next, another quirk of the `IDS IPL` was found: while only big-endian byte order may be used to encode image data in `PNG` files (W3C, 2003, Section 7.1), `IDS IPL` was doing the opposite. Meaning if a mosaicked `PNG` file were to be saved directly using the camera's `SDK`, its data would be all scrambled up and normal image viewers could not display it properly. However, this is the format that the `IPL` expected, so the processing script had to save the images with the wrong byte order to be able to use the `IDS IPL`'s debayering algorithm. An extra step was added for putting the data in the correct order after the demosaicing. This bug was later confirmed by the camera manufacturer (IDS GmbH, 2021a) and since it was fixed in a later version of the `IPL` (IDS GmbH, 2021b), the software was changed so that the byte swapping can be bypassed.

Then, the resulting image data is saved to the `TIFF` file format with only the upper eight bits. Regarding the lower bits, it was decided to carry them throughout the previous calculations in order to preserve accuracy in the least significant bits of the most significant byte, but they are discarded in the last step to save disk space.

---

[43] image processing library

Figure 3.14 demonstrates both described workflows. Two alternative paths are shown for the `IDS IPL` workflow, to account for the software version used.

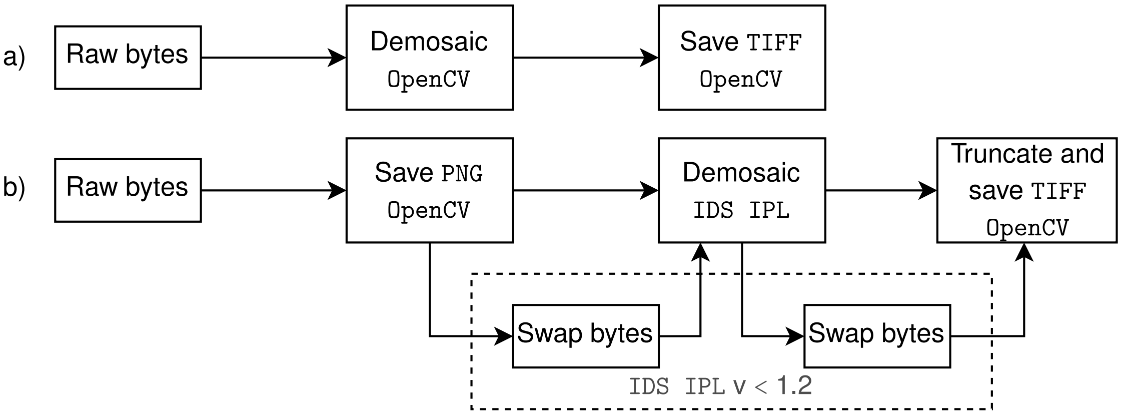


Figure 3.14: Workflows for demosaicing of RGB images. (a) Straightforward method using only `OpenCV`. (b) More convoluted process relying also on the `IDS IPL`. The second approach usually led to (subjectively) fewer artefacts and sharper images.

Finally, the timestamps of each image as saved by the APX are used to calculate the position and orientation of the camera. The georeferencing of the camera centre is done similarly to that of the lidar returns, as outlined in Section 3.3.2.1. The position (latitude, longitude and elevation), along with other metadata, such as ISO setting, lens aperture, integration time, etc. are written to the resulting files using `ExifTool` (Harvey, 2019). The orientation however is not part of the standard tags and storing it in the metadata is inconsistent among manufacturers. Therefore, it is saved as a `CSV` file that can be imported in e.g. `Agisoft Metashape` (Agisoft LLC, 2021, pp. 78-79)

3.3.2.3. General considerations for data processing

A number of optimizations were implemented to speed-up the calculations outlined in Sections 3.3.2.1 and 3.3.2.2. For instance, parallelization was used as much as possible. Significant gains were obtained due to `Python`'s `multiprocessing` module. Besides taking advantage of multiple CPU cores, this module enabled the use of `SharedMemory` objects for keeping the RAM footprint of the processing low. The `SharedMemory` objects were introduced in `Python` version 3.8.0 (Python Software Foundation, 2019), hence the minimum required version for the processing software outlined in this section.

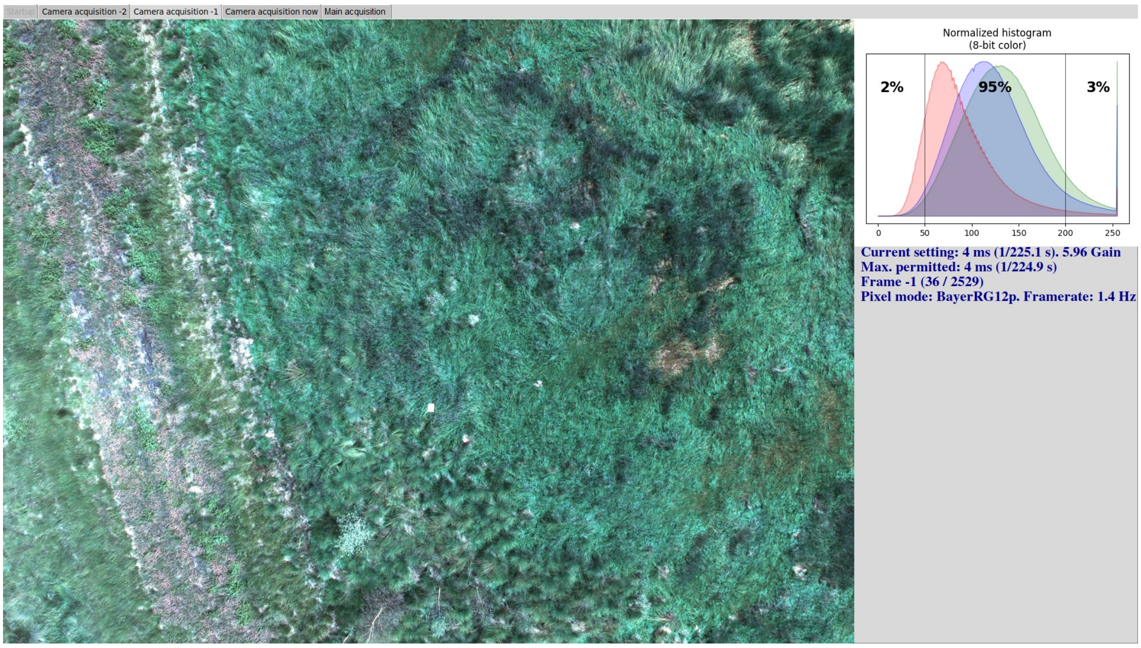
Because the processing of both the lidar and RGB data creates intermediary files, disk-IO becomes a bottleneck, especially thanks to the parallel processing introduced above. To combat this issue, a small portion of the RAM is normally used to produce a memory-mapped disk, so the intermediate disk-IO is minimized.

3.3.3. DATA VISUALIZATION

The acquisition software described in the Section 3.3.1 was designed to be started from the NUC's command line just before the UAV takes off for the flight. An SSH connection from a laptop is normally used to start the script via a temporary network connection between the two computers. This connection is closed during the flight, as cables cannot be used and a Wi-Fi connection might interfere with the UAV's remote controller, which operates in the same frequency bands. As a result, shall the acquisition script or one of its children fail during the flight, the operator would have no way of knowing and the flight would be al least partially wasted.

This issue was solved with one the features of the DJI M600 Pro. It has an HDMI input on the UAV, from where data is streamed via a 2.4 GHz radio to the remote controller, where it is displayed on a tablet. To the computer connecting to the input on the UAV side, the HDMI connection is indistinguishable from that of a normal monitor. For all practical purposes, the tablet can be regarded as just a display connected directly to the NUC.

To make use of this feature, a GUI[44] was programmed with `Python 3` (Version $\geq$ 3.5.0) for displaying the status of the scripts running on the NUC. A screenshot from this GUI is shown in Figure 3.15.



Current setting: 4 ms (1/225.1 s). 5.96 Gain
Max. permitted: 4 ms (1/224.9 s)
Frame -1 (36 / 2529)
Pixel mode: BayerRG12p. Framerate: 1.4 Hz

Figure 3.15: Screenshot from the GUI, showing one of the three most recent images. The photo's histogram and acquisition settings are shown on the right.

The GUI starts automatically when the NUC boots up. Initially it displays some general information about the system, as well as the time, so it does not appear static or frozen. As soon as it detects the acquisition script being started, it switches to in-flight mode, where it alternates between four screens every two seconds or so. On three of the in-flight screens, it displays the last three acquired images, each accompanied by a histogram and its acquisition settings, as shown in Figure 3.15. This way, the operator not only sees that the camera is working, but can also assess the image quality and the front overlap between images. The last screen displays general diagnostic information of the NUC: disk usage and free space available. It also shows the quality of the IMU and GNSS data, as reported by the APX. In between flights, the GUI switches back to the initial view.

Data acquisition and processing are not affected by the GUI. So if a UAV without an HDMI input were to be used, the GUI would simply not start and nothing would change regarding data acquisition and processing.

---

[44] graphical user interface

4. BORESIGHT CALIBRATION

So far, the alignment angles between APX, puck and UAV (aka boresight angles) were presented as multiples of $90°$. In practice, small mechanical imperfections lead to deviations of a few degrees from the designed angles. It is mandatory to measure and account for these deviations (aka boresight errors), as they lead to inaccuracies in georeferencing of the point clouds. This process is called boresight calibration.

One could argue that the precise angles between all three components (APX, puck and UAV) have to be measured perfectly for a successful boresight calibration. But in reality only the relative alignment of the puck to the APX is relevant. The APX is rigidly mounted to the puck and does not need to be unmounted between flight campaigns. However they, together with the rest of the system, are attached to the UAV using bolts and a sliding rail mechanism and usually get unmounted between flight campaigns. This way the UAV can be used with other sensor payloads. But it also means that the boresight errors between system and UAV can change from flight to flight. Therefore the UAV reference frame considered in Section 3.1 is just a construct that roughly aligns with the real UAV frame. This imaginary frame is defined by the transformations from the APX's frame as presented in Figure 3.2. So it is also part of the rigidly–mounted system, as it always moves with the APX. Therefore, for successful boresight calibration, it suffices that the puck's alignment to this imaginary frame is measured correctly.

In the present work, the boresight rotations are applied to the puck data after it is roughly brought into the imaginary UAV frame (Equation 3.5) and before applying the real-world roll, pitch and heading angles measured by the APX (Equation 3.6). This particular step in the data processing was chosen because the boresight angles applied at this stage are directly interpretable as roll, pitch and yaw misalignment between the puck and the APX. Also, for consistency with the previous sections, the same order of rotations (roll followed by pitch then yaw) was implemented for the boresight corrections.

The following sections describe the methods used to measure the boresight errors, the current results and give pointers to further work that can be done to improve these.

4.1. METHODS AND MATERIALS

When capturing lidar data with a miscalibrated system, the data appears distorted. This is because the position of each point is miscalculated. While the errors might not be apparent within one flight line, they are usually easiest to spot when comparing two lines flown in opposite directions. This is because the boresight errors lead to rotation of measured points in a particular direction relative to the system, so when facing opposite ways, the rotations are also in opposing directions, thus the errors get compounded. This property is often used for measurement of boresight misalignment. Reliable detection of features or points from different flight lines is a first step for boresight calibration.

To simplify the task of accurately detecting the same points in adjacent flight lines, a set of GCPs[45] were designed. These were prepared by applying retro-reflective foil to rigid plastic sheets. Retro-reflective materials have the property of reflecting a large proportion of the incident light back to its source. This is extremely useful for active sensing technologies, such as lidar, where a pulse of light is sent by the sensor and reflected by the environment. Because the lidar unit used reports not only distance to measured points, but also the intensity of the laser return, it is possible to extract the GCP-returns by looking for the brightest points in a point cloud. The use of retro-reflectors as GCPs was directly inspired by the work of Wallace et al. (2012).

A variety of lidar-GCPs was prepared. The plastic base sheets measured approximately 30 by 40 cm and the retro-reflective foil had sizes varying between 10 and 25 cm. The foil was applied as squares or crosses. Initially it was planned to assess the performance of the various shapes and sizes, but this comparison has not been carried out yet.

At the centre of each GCP, a hole was drilled for easy placement with survey nails. This way the position of every GCP can also be reliably measured with RTK-GNSS devices. Examples of the ground control points used can be seen in Figure 4.1.

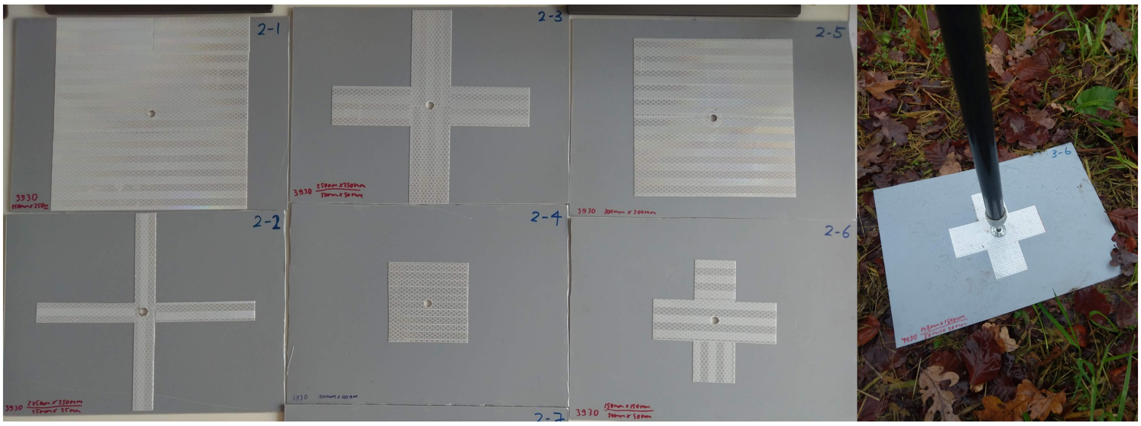


Figure 4.1: Ground control points. Left: multiple patterns and sizes. Right: GCP in the field. Measuring its position with an RTK-GNSS device.

4.1.1. BORESIGHT CALIBRATION ALGORITHM

A procedure was developed for finding the boresight angles. The following summarizes it and the successive paragraphs provide more details.

1. Identify the high-intensity returns in the raw puck data;
2. Filter the results using the known locations of the GCPs;
3. Extract the relevant pose data from the puck and APX files for each point;
4. Optimize the boresight angles:
   (a) Assign each return to its GCP;
   (b) Minimize the distance between points belonging to the same cluster.

In the first step, point clouds were generated using the methods described in Section 3.3.2.1, but only the returns with intensity > 100 were kept, as these were likely to be coming form retro-reflectors (Velodyne LiDAR, Inc., 2019b, p. 32). While all the points identified this

---

[45] ground control points

way closely resembled the actual positions of the GCPs in the Niederkamp campaigns, there were numerous false positives in the Vechta and Duisburg campaigns. This is why the second step was necessary. Here a simple spatial query was used and points situated more than x meters away from the GCP positions as recorded with a GNSS unit were discarded. The threshold was set manually for each flight campaign, by inspecting the results and choosing a value that would cut out all the false negative returns and impact as few true positives as possible.

Next, the surviving points were reidentified in the original puck files based on their GPS timestamp, which served as a unique identifier. Based on the spherical coordinates of the points in the puck frame, their Cartesian coordinates in the UAV's frame were calculated. Also using the timestamps, the northing, easting and altitude of the system, as well as the roll, pitch and heading angles were retrieved from the APX files. When available, the heading errors calculated by `Agisoft Metashape Pro` (Version 1.7.2) (Agisoft, 2021) were also collected. These parameters were aggregated and saved in a single file.

The fourth and final step commenced with reading the file generated earlier. Based on distances of the returns from the true positions of the GCPs, they were assigned to different clusters, so that all returns originating from the same point GCP belonged together. Then a routing that georeferenced the points as a function of boresight corrections was developed. The output of the routine quantified the spread of points within the clusters. The metric used was either mean, median or maximum Euclidean distance across all detected clusters. The function was used individually to identify trends and check results but most importantly, it was subjected to an optimization approach. This was implemented with the `optimize` module from the `SciPy` package (Virtanen et al., 2020) and searched for angles that minimized the intra–cluster spread. Methods looking for either global or local minima were employed.

Because the results were inconsistent across flight campaigns, the procedure was repeated for each campaign: Niederkamp 1 and 2, Vechta and Duisburg.

4.1.2. Refinement of heading with photogrammetry

The algorithm presented so far treated the errors as constant over time. This is not the case if the IMU is drifting, which is a common occurrence and has often been remediated by using cameras rigidly mounted to the payload. For instance, Wallace et al. (2012) used an RGB camera and SfM techniques to correct IMU errors in a UAV-borne lidar system. Suomalainen et al. (2014) had a similar approach, where the camera orientation obtained from SfM with a low update frequency was taken as truth and high update frequency GPS-IMU data was used to interpolate the instrument pose in between camera frames.

In order to verify whether the potential errors can at all be detected with the current camera setup, `Metashape` was used to process images taken during the Vechta campaign. The position (longitude, latitude, elevation) of each image shot was calculated using the GNSS-IMU data and the lever arm distances from the puck centre to the camera centre. Also the orientation of each capture (roll, pitch, heading) was calculated assuming perfect alignment between camera and puck coordinate systems (See Section 3.3.2.2 for details). The position was added to images as metadata, which was read be `Metashape` and the orientation was imported into the software as a `CSV` file. Thus `Metashape` had the initial pose for each capture. Then it adjusted the pose of each camera shot by matching features in adjacent images and optimizing both the internal and external camera orientation parameters. Afterwards it calculated the errors between the input angles (from the `CSV`) and the output ones (calculated by `Metashape`). These errors were analysed to determine whether they can be used to compensate for a potential IMU drift.

First, the errors computed by `Metashape` were plotted for each of the three angles (roll, pitch and heading) and 23 flight lines. Then, for each of these, a simple linear regression line was fitted. Because obvious trends could be observed but there were also outliers not fitting these general relationships, a low-pass filter was applied to each sub-dataset to accentuate these general trends. The raw errors, as well as the filtered ones and the linear regression lines are shown for each angle and flight–line in Figure 4.2.

As stated in the headers for each of the subfigures, the mean errors were: $-0.11°$ for heading, $-0.09°$ for roll and$-0.65°$ for pitch. These are mainly an indication for the general misalignment between puck and camera and their exact value would only be relevant for computing the accurate camera orientation. However, the interesting parts are the standard deviations of these errors: only $0.03°$ for roll and pitch, but an entire $0.12°$ for heading. Assuming this is not a fluke, this might mean the heading of the system changes significantly even within a flight line and this time–dependent error can indeed be measured with the RGB camera.

Although the trends seen in the heading behaviour are clearly not just linear, sometimes changing direction even within a flight line, the high and significant R^2 values accompanying them suggest the heading errors are quite time–dependent. Which cannot be said of the roll and pitch errors. The distribution of the R^2 & p values and of the absolute values of the regression coefficients for the considered errors are shown in Figure 4.3.

It follows that the heading drift seems to be real and measurable with the RGB camera. This is not the case for roll and pitch, whose variation is rather random and not time–dependent. Therefore, only the correction of heading with SfM was considered further. To assess whether the boresight calibration can be improved with this refinement, the low-pass data was used to create look-up tables which were then used by the boresight algorithm to correct the heading. The boresight calibration of the Vechta data was done twice: once with and once without the heading corrections.

4.2. RESULTS AND DISCUSSION

A few minimization algorithms were used and while all of them converged to almost identical results, the Nelder–Mead method (Nelder & Mead, 1965; Gao & Han, 2010) usually found the most plausible angles. Therefore, the results of this method are normally presented in this Section. The only exception occurred when considering a dataset combining multiple flight campaigns. Then the Powell method (Powell, 1964) yielded the more plausible results. When minimizing either the mean, median or maximum error, the results varied by $\approx 0.2°$. Still, only the results obtained when minimizing the maximum error were considered in this Section, as it was reasoned that focusing the least forgiving met-

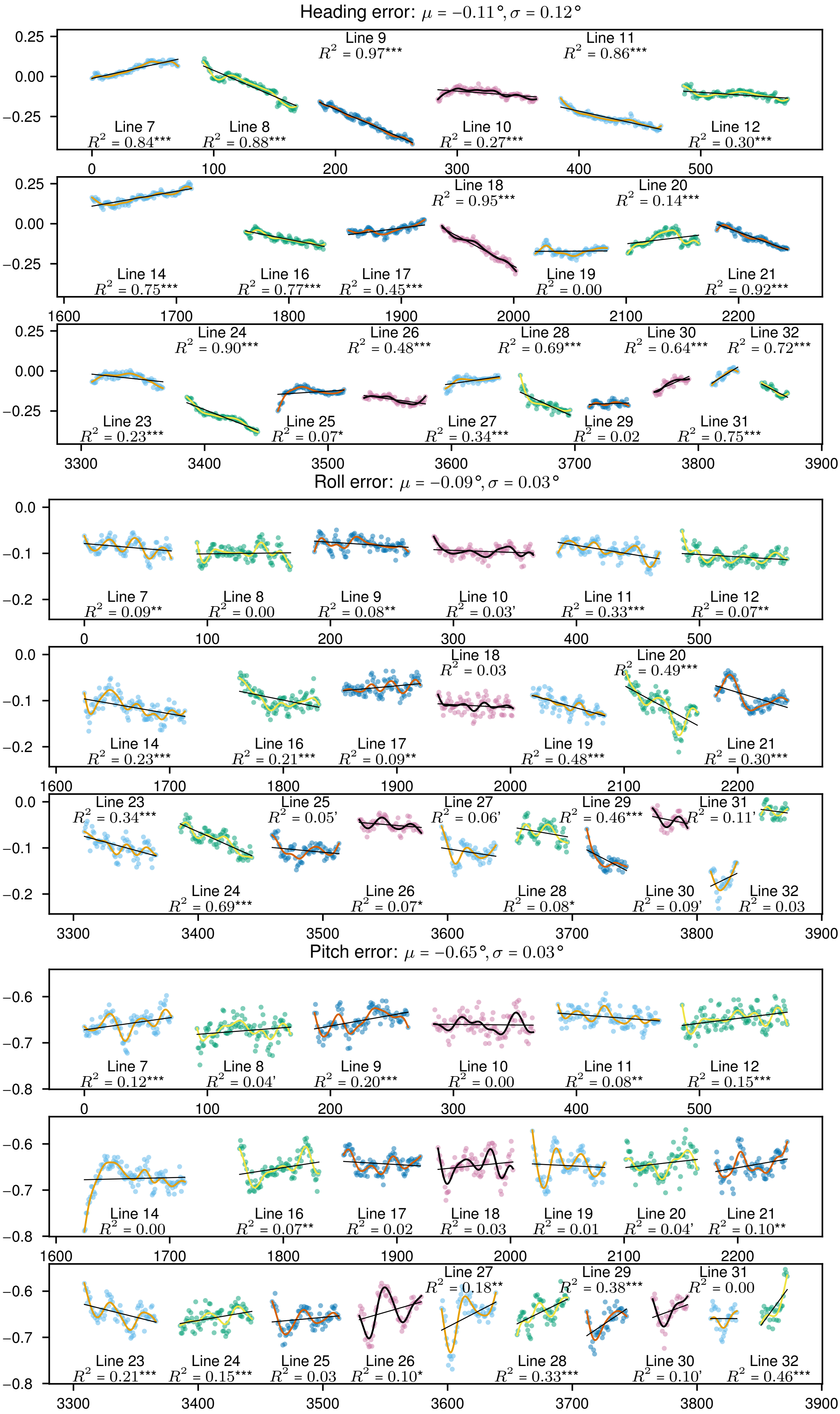


Figure 4.2: Pose error over time – Missions 2–4 in Vechta campaign. Raw values are plotted as dots. Low-pass filtered values are overlaid as curves. The straight black lines show the fitting linear models. y-axes are in degrees and x-axes — in seconds from the beginning of the flight line 7. R-style significance codes for each R^2: 0 *** 0.001 ** 0.01 * 0.05 ' 0.1. No symbol for $p > 0.1$.

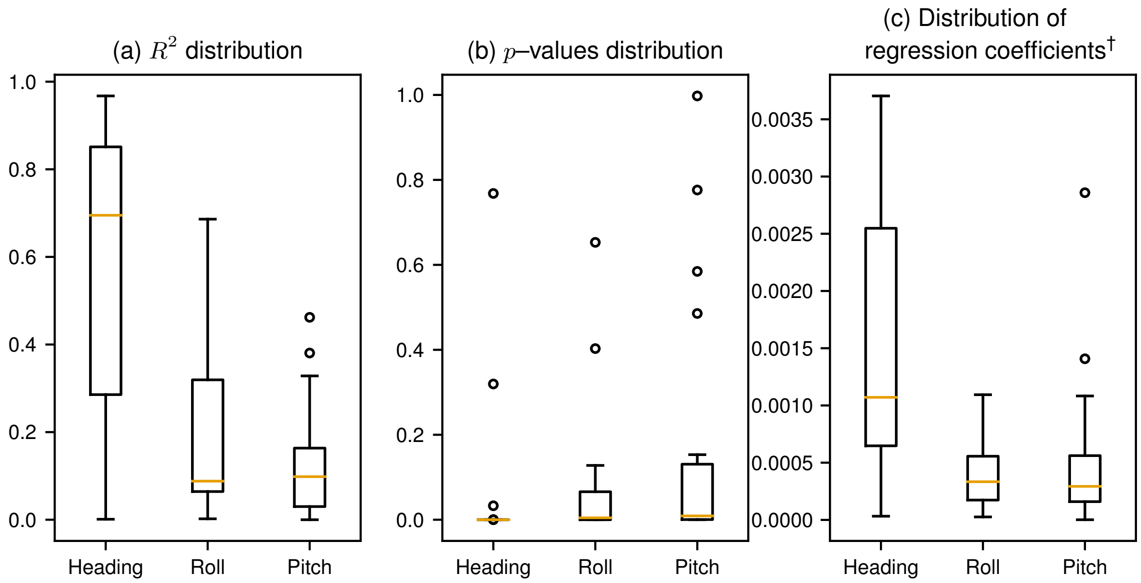


Figure 4.3: Characterization of `Metashape`–detected boresight errors: Distribution of the (a) coefficients of determination, (b) p-values and (c) regression coefficients† for the linear models.
† The absolute values are used for the latter, to indicate whether a linear relationship could be identified at all, not its sign (c). The strengths and significances of these relationships can be assessed based on the distribution of R^2 (a) and p respectively (b).

ric should suppress the errors the most. Results of the per–mission maximum boresight error minimization are presented in Table 4.1. Figures 4.4–4.8 demonstrate the maximum error as a function of the three angles for the last five columns from Table 4.1.

Table 4.1: Results of boresight calibration for each mission. Results were obtained by minimizing the maximum error. Columns one–five: Nelder–Mead method; column 6: Powell method. ε refers to distances between points belonging to the same GCP. Last row shows the number of observations and the number of GCPs identified.
* Vechta *without* yaw refinement; † Vechta *with* yaw refinement.
‡ Combined: Vechta *with* yaw refinement, Niederkamp 2 and Duisburg.

	Vechta*	Vechta†	Niederkamp 1	Niederkamp 2	Duisburg	Combined‡
Roll	0.992	0.920	0.892	1.077	0.943	1.082
Pitch	0.122	0.141	0.231	0.246	0.180	0.297
Yaw	-0.276	-0.337	-1.918	-1.775	-1.515	-1.117
Max. ε	0.400	0.373	0.311	0.644	0.413	0.922
Mean ε	0.152	0.139	0.180	0.293	0.155	0.310
Median ε	0.143	0.136	0.167	0.306	0.155	0.254
$\mathrm{N_{obs}/N_{GCP}}$	$209/19$	$209/19$	$9/2$	$81/20$	$1135/10$	$1425/49$

Figures 4.4–4.8 have three columns and three rows each. The columns correspond to one of the angles being fixed to its optimal value and the other two varying across a certain range. Each successive row zooms in on a smaller search space for the two variable angles: $3 \times 3°$ in the first row, $0.3 \times 0.3°$ in the second row and $0.08 \times 0.08°$ in the last row. The zoom-in areas are indicated by red squares in the two upper rows and the optimal solutions found with the Nelder–Mead method are marked with red crosses in the last rows. Each subplot's colour bar shows the range of the maximum intracluster error.

An exception was made for Figure 4.8: the solutions obtained with the Nelder–Mead method were suboptimal, and were still indicated by red crosses, but in the second row

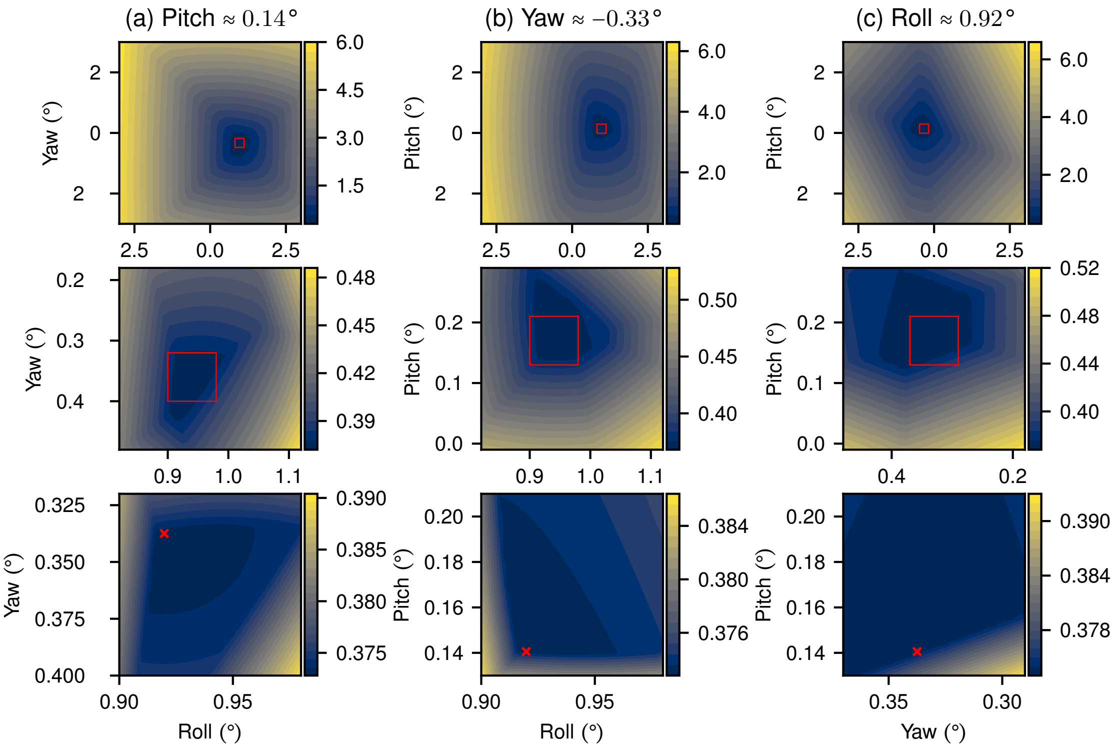


Figure 4.4: Boresight error for various angles – Vechta with yaw refinement.

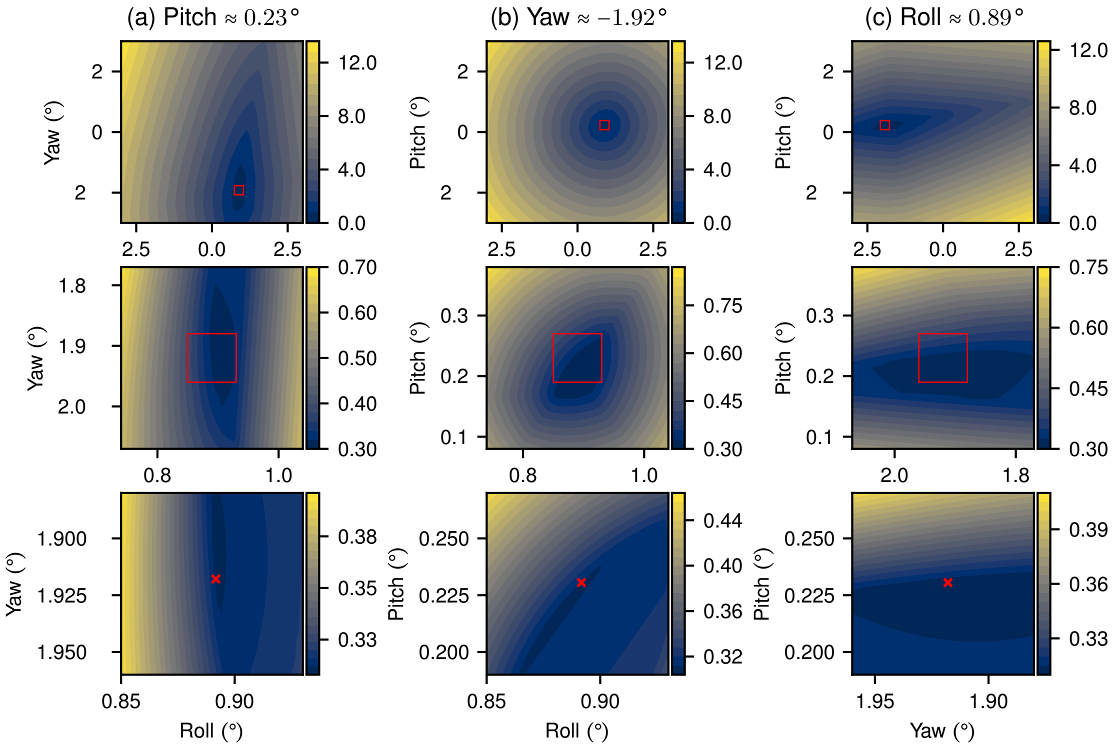


Figure 4.5: Boresight error for various angles – Niederkamp 1.

instead of the last one. A more plausible result was instead obtained with the Powell method (Powell, 1964). These minima are shown with green asterisks in the last row.

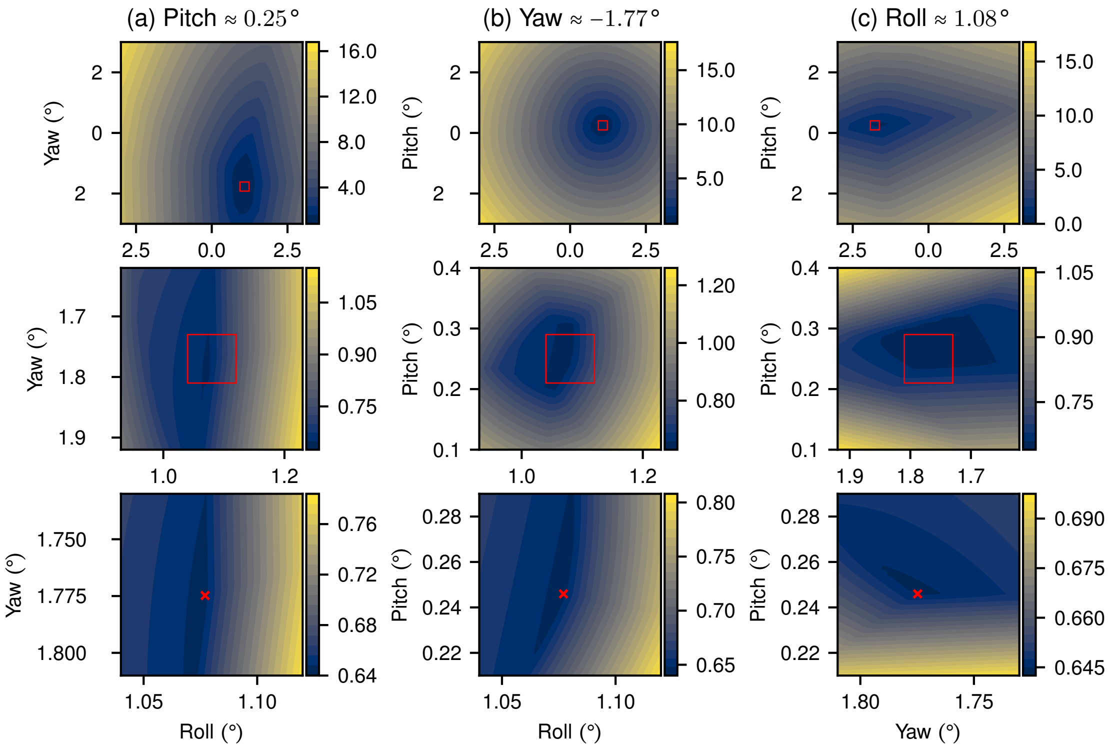


Figure 4.6: Boresight error for various angles – Niederkamp 2.

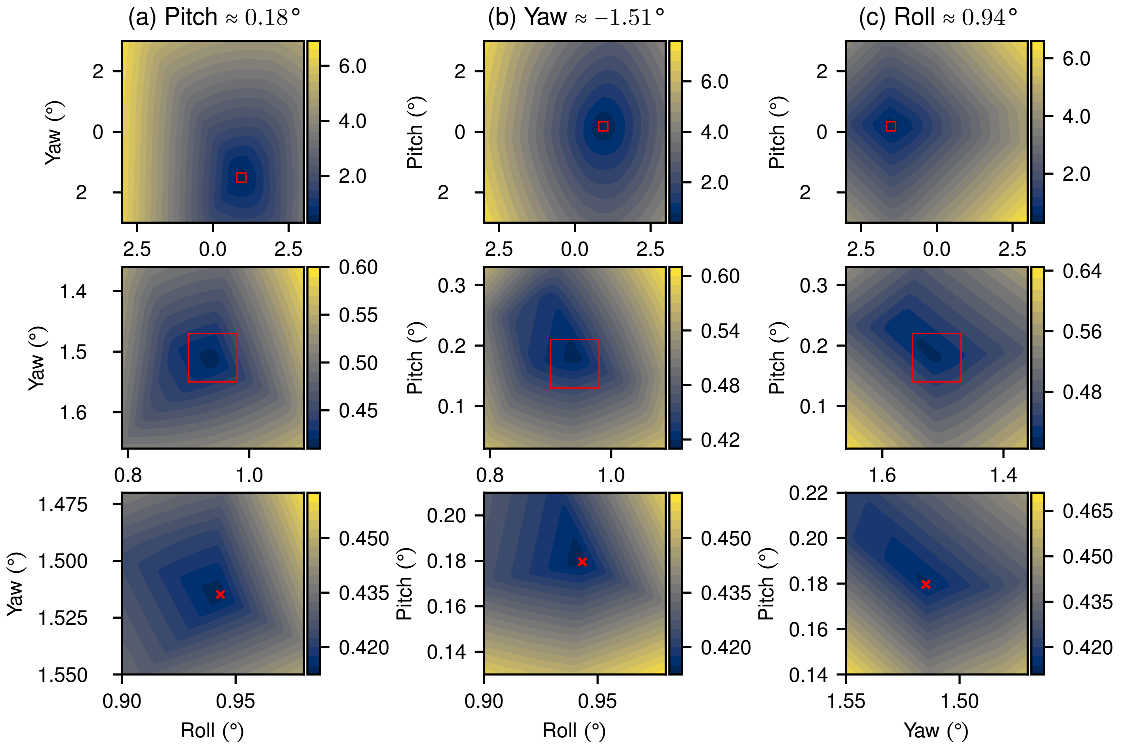


Figure 4.7: Boresight error for various angles – Duisburg.

There are a few observations to be made from Figures 4.4–4.8 and Table 4.1. In principle, there are ranges of angles that significantly reduce the maximum error for a specific flight campaign. Also, using the `optimize` functions from the `SciPy` module is an effective way

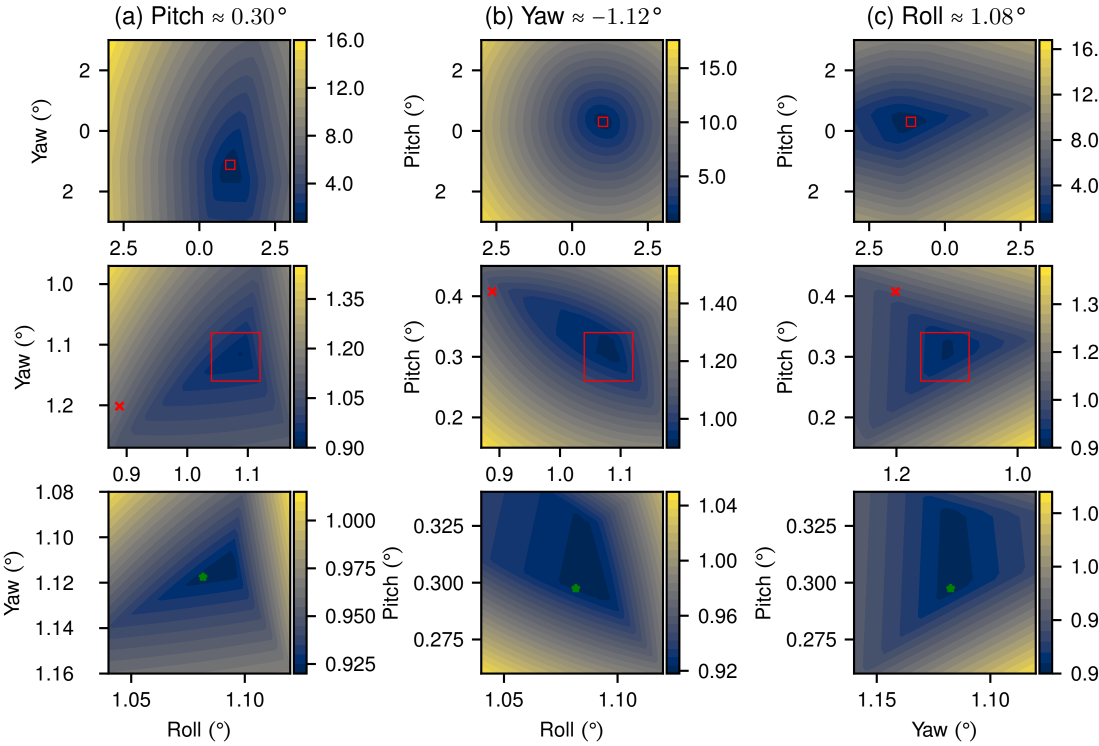


Figure 4.8: Boresight error for various angles – Combined.

to find such combinations of angles. However, combinations that minimize the metric for one campaign are far from ideal for others. The disagreement between the per campaign solutions reached $\approx 0.18°$ in roll and pitch and $\approx 1.6°$ in yaw. While the residual errors in roll and pitch are not as drastic as those in yaw, the combination of their individual inaccuracies can still lead to a horizontal positioning error of ≈ 0.5 m at a range of 100 m. There are a few reasons why the current approach is flawed.

First of all, the size of the retro-reflective panels is comparable to the error that has to be minimized. A laser return from a panel is currently trated as one point. When in fact, the laser could be striking anywhere on the 10×10 to 25×25 cm surface of the panel. This issue is further compounded by the size of the laser footprint itself, which is anywhere between 8×5 cm at a distance of 20 metres and 29×16 cm at 100 metres. And when the laser hits any surface at an angle, one dimension of the pulse grows with the inverse of the cosine of the off-nadir incidence angle. So in fact it is fairly large laser footprints landing on comparably large panels. Therefore treating their interaction as a point while trying to minimize distances between these “points” on a comparable spatial scale inherently leads to errors. The current size of the GCPs was chosen as a compromise between the accuracy of the positioning and the chance of them being hit by lasers. Evidently the first objective has not been satisfied and a different strategy should be employed. Potential improvements are discussed in Section 4.3.

The errors were so far assumed to be random, in which case increasing the sample size would cancel them out. However, even with the 1135 observations of the 10 GCPs in the Duisburg mission, the mean error could not be brought under 15 cm, which corresponds to $\approx 0.36°$ of error given the flying height of25 m. The error in the Niederkamp 2 mission (at 90 m height) corresponds to $\approx 0.19^\circ$ and in Vechta (at 20 m height) – to $\approx 0.40^\circ$. It is tempting to state that the error decreases with increasing height, but this cannot be said with confidence yet, because of the low sample sizes in all missions except for Duisburg.

To verify how the current results reflect on the quality of the point clouds, a test was conducted with the data from the two Niederkamp campaigns and their respective calibration results. Ground points from two opposing flight lines (1 and 2) were extracted and used to generate a DTM[46] for each of the two flight lines. Both operations were achieved with `LAStools` (Isenburg, 2021). Next, the difference of the two DTMs was calculated using QGIS 3 (QGIS Development Team, 2021). The above procedure was repeated three times for each of the two campaigns considered: no boresight calibration (Original), optimal solution for Niederkamp 1 as per Table 4.1 (Solution 1) and optimal solution for Niederkamp 2 (Solution 2). The distribution of errors between the two DTMs for each of the six cases is shown in Figure 4.9.

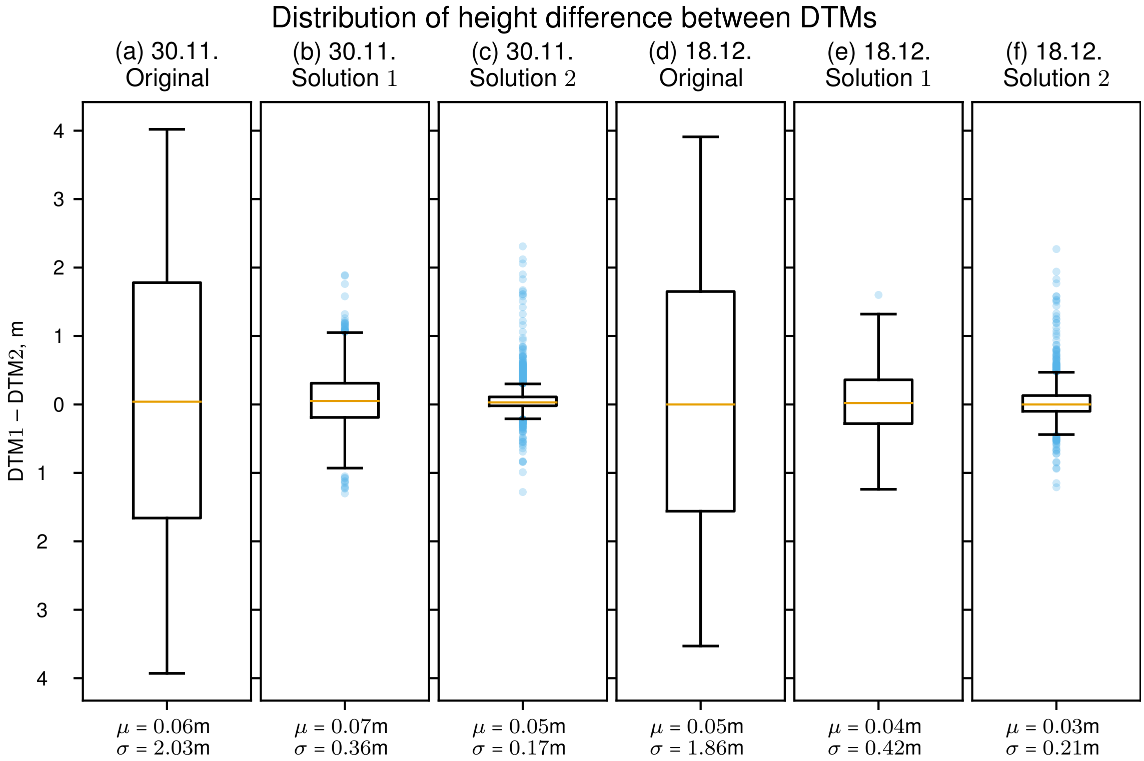


Figure 4.9: Boxplots of DTM differences. (a), (b) & (c) show data from the flight campaign Niederkamp 1. Meanwhile (d), (e) & (f) present data from the flight campaign Niederkamp 2. Plots (a) & (d) were obtained without applying any correction; plots (b) & (e) – by applying the solution computed for Niederkamp 1 (-1.918° yaw, 0.231° pitch and 0.892° roll); plots (c) & (f) – by correcting the boresight angles with the solution found for Niederkamp 2 (-1.775° yaw, 0.246° pitch and 1.077° roll).

In Figure 4.9 all means are close to zero, but the standard deviations do change considerably with different corrections. While both sets of angles lead to a substantial decrease in the range of errors, those from Solution 2 perform the best. This was to be expected, since 20 GCPs were observed 81 times in the Niederkamp 2 campaign, in contrast to just two GCPs observed nine times in the Niederkamp 1 campaign. Still the errors were not decreased sufficiently in any of the iterations. This shows on one hand that the general

---

[46] digital terrain model

trend has been correctly captured in the Solution 2, since it could be applied to datasets which were not “seen” at the optimization stage. But on the other hand, the residual error (17–21 cm even with the superior method) shows that the currently used boresight angles do not perform well enough yet and should be further refined.

Another view of the data from Figure 4.9 is offered in Figure 4.10, where the difference of the two DTMs is shown for the uncorrected data vs data corrected with Solution 2.

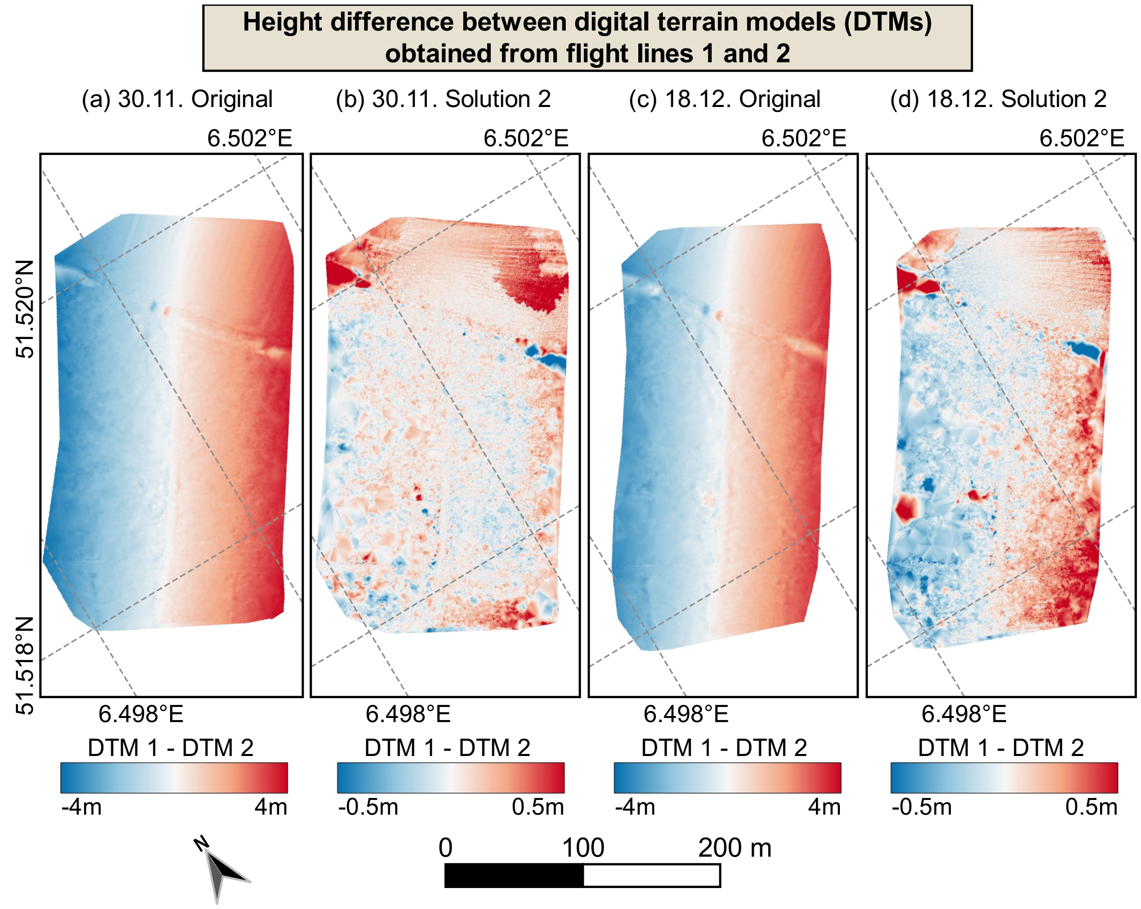


Figure 4.10: Difference between DTMs generated from flight lines 1 and 2. (a) & (b): Campaign Niederkamp 1. (c) & (d): Campaign Niederkamp 2. (a) & (c): Data without any correction. (b) & (d): Data corrected with these angles: $-1.775°$ yaw, $0.246°$ pitch and $1.077°$ roll.

It can be seen in Figure 4.10 that while considerable misalignments between opposing flight lines remained even after boresight correction, their magnitude became much smaller and the distribution – more random, the initial cross-track gradient being considerably suppressed.

4.3. CONCLUSION AND OUTLOOK

The present chapter demonstrated a method for lidar boresight calibration using retroreflective panels and a distance minimization approach. While a general range of appropriate boresight correction angles had been identified, the precise value of the boresight misalignment of the system have not been measured to a satisfactory degree yet.

A number of improvements could be made to the current approach. For once, a dedicated boresighting campaign should be undertaken. The terrain to be surveyed should be open and easily accessible. Numerous GCPs must be used, all spaced throughout the area. Their size has to be decreased, maybe to just 2–5 cm, because as shown in Appendix B, with higher flying altitudes the chance to hit (even small) targets increases considerably. Furthermore, special attention should be given to accurately surveying the targets and potentially minimizing the distances not between pulses but from pulses to targets.

Considering the current results for Vechta, Duisburg and Niederkamp 2 campaigns (Table 4.1), the optimal roll and pitch angles do seem to change with height. Since lever-arm measurement errors could manifest themselves as height-dependent systematic boresight errors, it is imperative that the current lever-arm measurements between all system components be thoroughly checked and if necessary, corrected.

Shall the approach still not yield satisfactory results, other algorithms must be used. For instance, identifying coplanar points (e.g. roofs) and adjusting the boresight angles such that the planes match proved to be an effective strategy for boresight calibration of UAV-borne lidar systems (Wallace et al., 2012; Guo et al., 2017).

5. USE CASE 1. *Vechtaer Moor*: TOWARDS MICROFORMS DETECTION IN A REWETTED CUT-OVER BOG

Peatlands are of uttermost ecological importance, featuring unique habitats and biodiversity. They also provide a myriad of ecosystem services, such as water purification, flood mitigation and carbon storage (R. Andersen et al., 2017). While intact peatlands function as net carbon sinks, they turn into major sources of greenhouse gases when dried up (Joosten & Couwenberg, 2008). Up until the 1970s, peatlands (especially bogs) in Central Europe had been intensively dewatered and cut for extending agricultural and forestry areas, as well as for peat production, which was used in potting mixes and as solid fuel (Joosten, 2012). Restoration of peatlands through rewetting is an important and urgent measure of climate change mitigation (Günther et al., 2020).

Since restoration of peatlands is a lengthy process (Nick, 2001, as reported by Raabe et al., 2018), it is imperative to monitor the state of the peatlands during this time. Detection and classification of microforms, such as hummocks and hollows, with UAV imagery has been investigated in several studies (Lehmann et al., 2016; Lovitt et al., 2017). Characterization of wide-scale bog surface has been carried out with high-altitude, low point density (10^{-1}–10^{1} points/m^2) discrete lidar data (Töyrä & Pietroniro, 2005; Korpela et al., 2009; Rapinel et al., 2011; Luscombe et al., 2015; Carless et al., 2019; LaRocque et al., 2020; Räsänen et al., 2020). By design, these approaches could not effectively describe submeter microforms. On the other hand, Korpela et al. (2020) used a waveform-recording lidar mounted on a low-flying helicopter (≈ 250 m above ground level) to acquire data with higher point density (≈ 50 points/m^2), which they used to successfully map microforms in a boreal bog.

No study using discrete-point lidar data acquired from a lower altitude and attempting to identify microforms could be found. This was confirmed by a recent review of UAV applications in wetlands, which concluded that no research using UAV-borne lidar sensors to study microforms in bogs has been conducted to date (Dronova et al., 2021). However the need for high-resolution UAV-borne lidar surveys has already been established (Lehmann et al., 2016). The end goal would be to combine geometrically-accurate lidar data with spectrally-rich RGB/multispectral data in order to e.g. improve the classification accuracy of approaches based only on imagery. This direct application was not possible in the current work, because of the persisting issues with the boresight calibration of the system. However, a method for aligning data from flight lines a small ROI[47] has been developed. Moreover, the high- and low-frequency components of a high-resolution model of the bog's surface are generated along the way.

This Chapter presents the above algorithm and its application over a small 20×20 m ROI within the *Vechtaer Moor* (the Bog Vechta), where ongoing restoration efforts are made

---

[47] region of interest

in a joint project of several parties (Raabe et al., 2018). Section 5.1 describes the data acquisition campaign and the developed method in detail. Next, Section 5.2 showcases the algorithms application to the given ROI. Finally, Section 5.3 summarizes the current results and offers pointers towards future research.

5.1. METHODS AND MATERIALS

5.1.1. STUDY AREA AND DATA COLLECTION

The general location of the ≈ 10 ha area of interest (AOI) in the *Vechtaer Moor* was shown in Figure 2.1 in Chapter 1. Figure 5.1 presents the position of the ROI within the AOI.

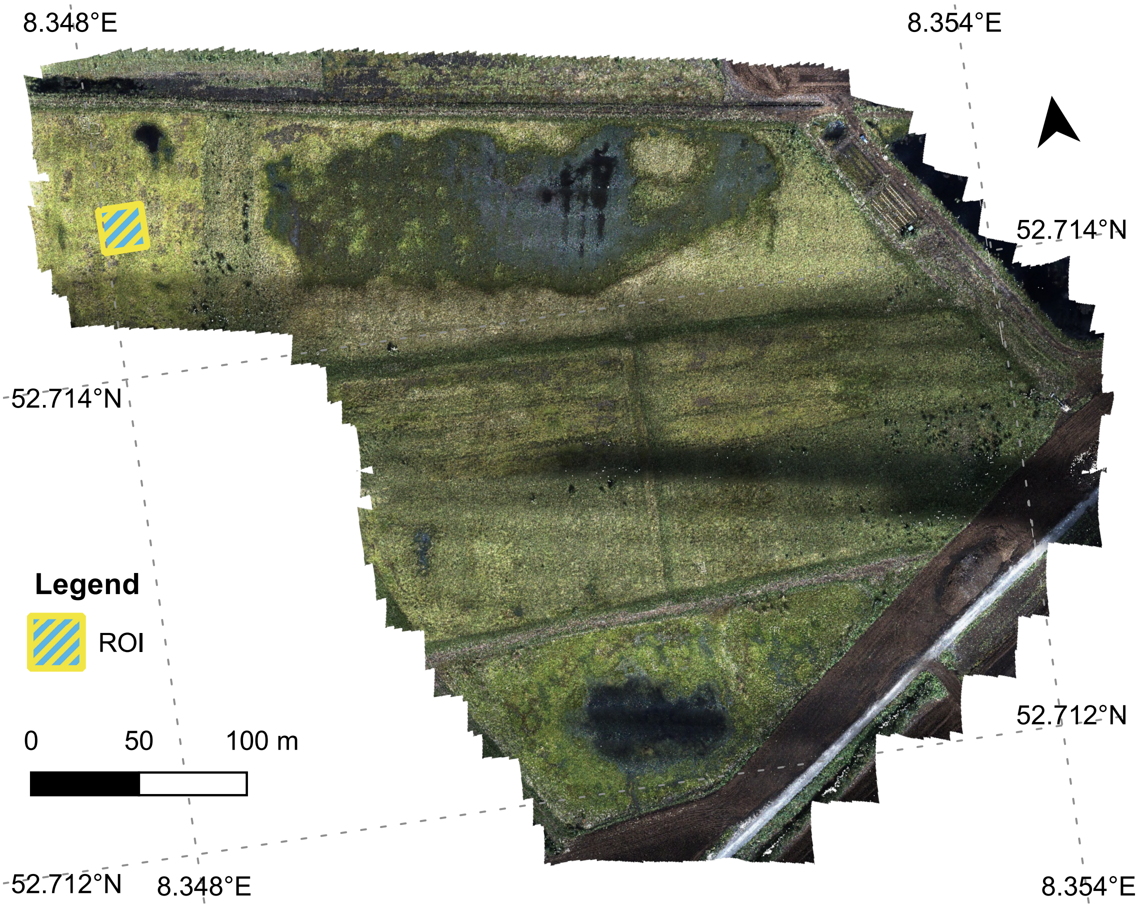


Figure 5.1: Orthomosaic of AOI and location of the 20×20 m ROI.

The data collection had been performed on $19.08.2020$. Three flights with the lidar system covering the entire AOI have been carried out between $12{:}00$ and $13{:}15$. The flying height was 20 m above ground and the UAV flew at 5 m/s. The flight lines were spaced at ≈ 14.5 m and the RGB camera was triggered at ≈ 1.3 Hz so that 85% front and 60% side overlap were achieved for the RGB imagery. The images were in turn processed in `Agisoft Metashape` (Agisoft, 2021) to derive an orthomosaic (Figure 5.1) and a DSM[48] of the area of interest. No further analysis of the RGB data has been conducted so far.

The puck's rotating frequency was set to 11 Hz and the acquisition mode to "dual", meaning a maximum of two returns would be recorded for each lidar outgoing pulse. Lidar returns from the entire $360°$ FOV of the puck were saved.

---

[48] digital surface model

On the same day, multispectral and RGB imagery had been collected by J. Lehmann and H. Schneidereit from the University of Münster. Data from the entire AOI was collected with a vertical take-off and landing UAV flying at 100 m above ground level. The data of the two systems had not been fused together yet, but it is planned to combine the spatial accuracy of the lidar with the spectral information of the latter system to support ongoing vegetation mapping efforts at the Bog Vechta.

5.1.2. PROCESSING ALGORITHM

A procedure for merging misaligned point clouds and separating the surface of the resulting point cloud into its low- and high-frequency components was developed. In this Section, the low-frequency component is loosely referred to as a DTM because it characterizes slow changes in *relief / terrain*. The high-frequency component is loosely called a DSM because it describes local variations of the terrain's *surface*. This usage is not in accordance with the normal definitions of the two terms, but was employed to aid differentiating between the two output models.

Here is an overview of the method: In the first step, the points belonging to the ROI were extracted from all the flight lines. An initial classification of ground points was performed in each of the clouds. The resulting ground points were resampled on a coarse 20×20 cm grid, keeping only the lowest point per cell. From each of the subsampled files, a 1×1 cm DTM was generated. These DTMs with artificially high resolution (simply called artificial DTMs from now on) served as the basis for aligning the original point clouds. Furthermore they were used to generate a low-frequency "background" DTM, relative to which the aligned point clouds were normalized. The results were merged into a single point cloud. This final normalized cloud had sufficient point density for the generation of a high-resolution DSM, which showed the high-frequency variation from the low-frequency background DTM.

Up to the generation of the artificial DTMs, standard point cloud processing techniques were applied using `LAStools` (Isenburg, 2021). However some details are offered for the subsequent steps. The artificial DTMs were smoothed with Gaussian blurring using large kernels ($\sigma = 200$) to obtain their low frequency versions. These are referred to as smooth DTMs. By subtracting the smooth artificial DTMs from the original ones, rough versions of the artificial DTMs were obtained.

The rough DTMs were used to detect the horizontal misalignment between the original point clouds: One of the input layers were designated as "true" and was successively convolved with each of the remaining layers. The locations of the maxima in the result were interpreted as the translation necessary for bringing the other layers into alignment with the "true" one.

A simpler approach was used to detect the vertical misalignment: The "true" terrain elevation was declared to be the mean of the median elevations of the input rasters. A correction was calculated for each of the inputs simply as the difference between its median height and the "true" terrain elevation. After applying these corrections to the artificial DTMs, they were translated such that they almost matched. But some rotation because of boresight misalignment remained and the ICP[49] (Besl & McKay, 1992) implementation in `CloudCompare` (CloudCompare, 2020) was used to refine the alignment.

The previously calculated transformations (from median, convolution and ICP approaches) were applied to the initial point clouds. They were also applied to the smooth DTMs. Next the mean smooth DTM was used for normalizing all the original point clouds, i.e. the height of each point was recalculated as the height above or below the surface of the smooth DTM. The final low-frequency DTM was obtained by averaging the now-aligned smooth DTMs. The high-frequency DSM was generated from a Delaunay triangulation of the surface of the ground points in the aligned normalized clouds.

5.2. RESULTS AND DISCUSSION

The proposed workflow was implemented for a 20×20 m ROI in the north-west part of the AOI. The location of the ROI is shown in Figure 5.1.

The ROI had been observed from seven different flight lines. The individual point clouds had a mean density of 206 points/m^2 (min = 37, max = 432 points/m^2). The distribution of the point heights in the initial clouds is shown in Figure 5.2. The misalignment of the point clouds becomes evident in this Figure: In each flight line the bulk of the returns lies at a slightly different height.

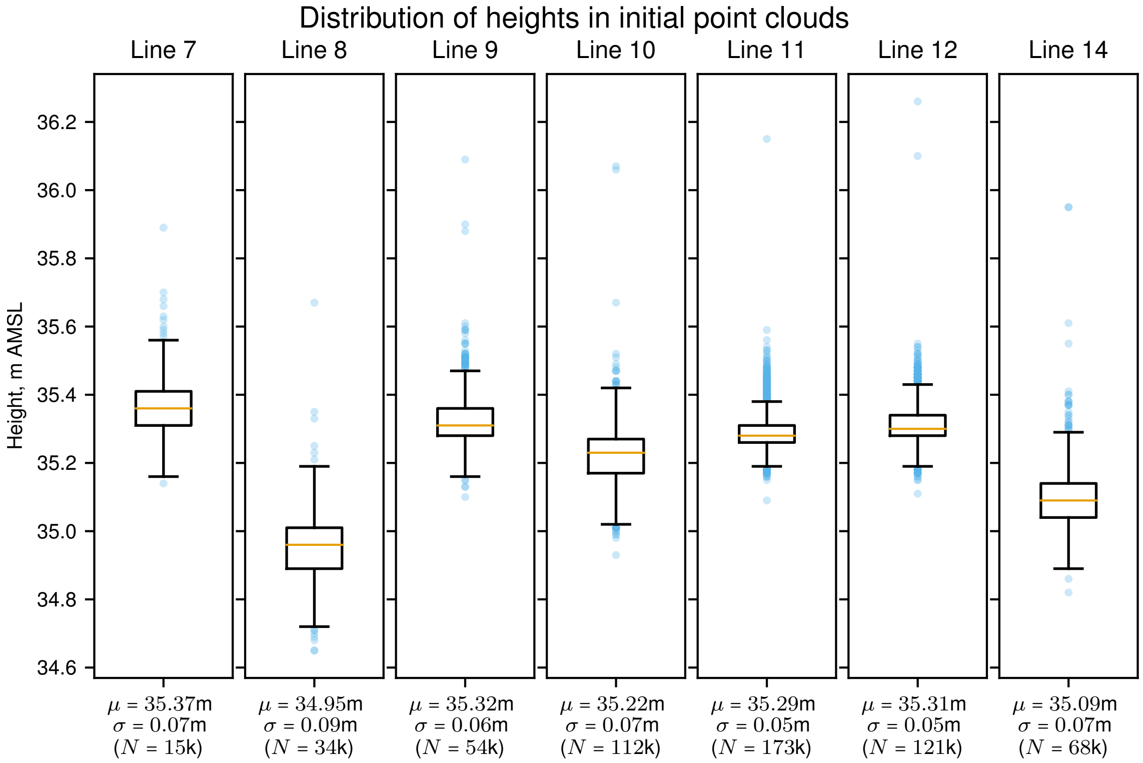


Figure 5.2: Point heights in input files.

The effects of the misalignment of the flight lines are presented in Figure 5.3: a DSM that was produced directly from the initial point clouds. Some effects of using misaligned clouds are evident: The DSM appears noisy, with numerous points higher and others lower than the general surface. This can be explained by the height mismatch of the

---

[49] iterative closest point

individual lines (see Figure 5.2): The interpolated surface contains points from all layers and not just from the e.g. the highest. This problem was surely facilitated by the relatively low point density of the individual clouds. If the highest layer had a high point density, the layers underneath would not have contributed to the DSM. It would still be an erroneous elevation map of the ROI, just not so obvious.

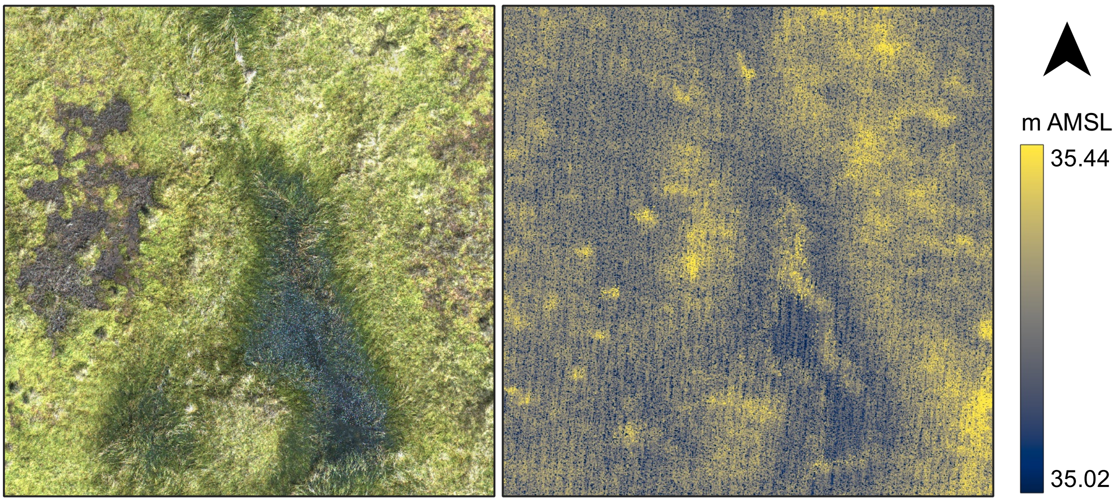


Figure 5.3: DSM produced from the original, misaligned, point clouds. Left: orthomosaic of the ROI. Right: DSM of the ROI. Produced by merging the seven original point clouds. Colour-bar: height above mean sea level.

The height distributions of all the original clouds were similar in shape, only shifted up or down. Therefore the height mismatch between the flight lines was easily solved using the median approach. The vertical corrections necessary for bringing median heights of the smooth DTMs to the same value ranged from -13 cm for line seven to $+28$ cm for line eight. For calculating the horizontal translation between the rough DTMs, it was necessary to choose a pivot DTM. Line eleven was chosen as the pivot because its vertical correction was the median among the other vertical corrections. The convolution of the other rough DTMs with that of line eleven yielded the shifts needed in easting and northing. These ranged from -28 to $+41$ cm in easting and -18 to $+13$ cm in northing.

Figure 5.4 shows the seven smooth DTMs, as well as mean smooth DTM, after translating them with the above corrections. While the rasters' mean values do agree with each other, it can be seen that they are clearly rotated about a east-west line. This rotation can be attributed to a roll error, since the flight of the UAV were roughly east-west oriented.

Figure 5.5 further shows the rotation of the smooth DTMs about the flight lines' axis.

The rotation problem seen in Figures 5.4 and 5.5 was solved with the ICP algorithm. Figure 5.6 shows the points clouds after rotating and translating them towards the mean smooth DTM.

An important observation is that while some point clouds in Figure 5.5 only remotely resembled the mean DTM, all the clouds in Figure 5.6 look similar to each other and the mean DTM. This can be further seen in Figure 5.7.

The fact that all transformed DTMs agree on the low-frequency structure of the terrain is reassuring. By applying the transformations that were calculated for the smooth DTMs to

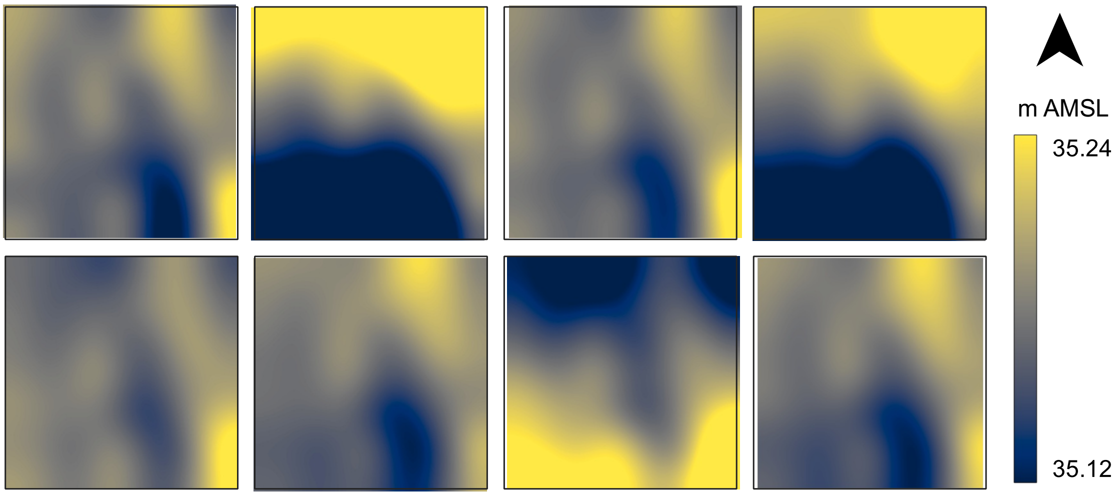


Figure 5.4: Smooth DTMs before ICP: top-view. Top row from left to right: lines 7–10. Bottom row from left to right: lines 11, 12, 14 and the mean smooth DTM.

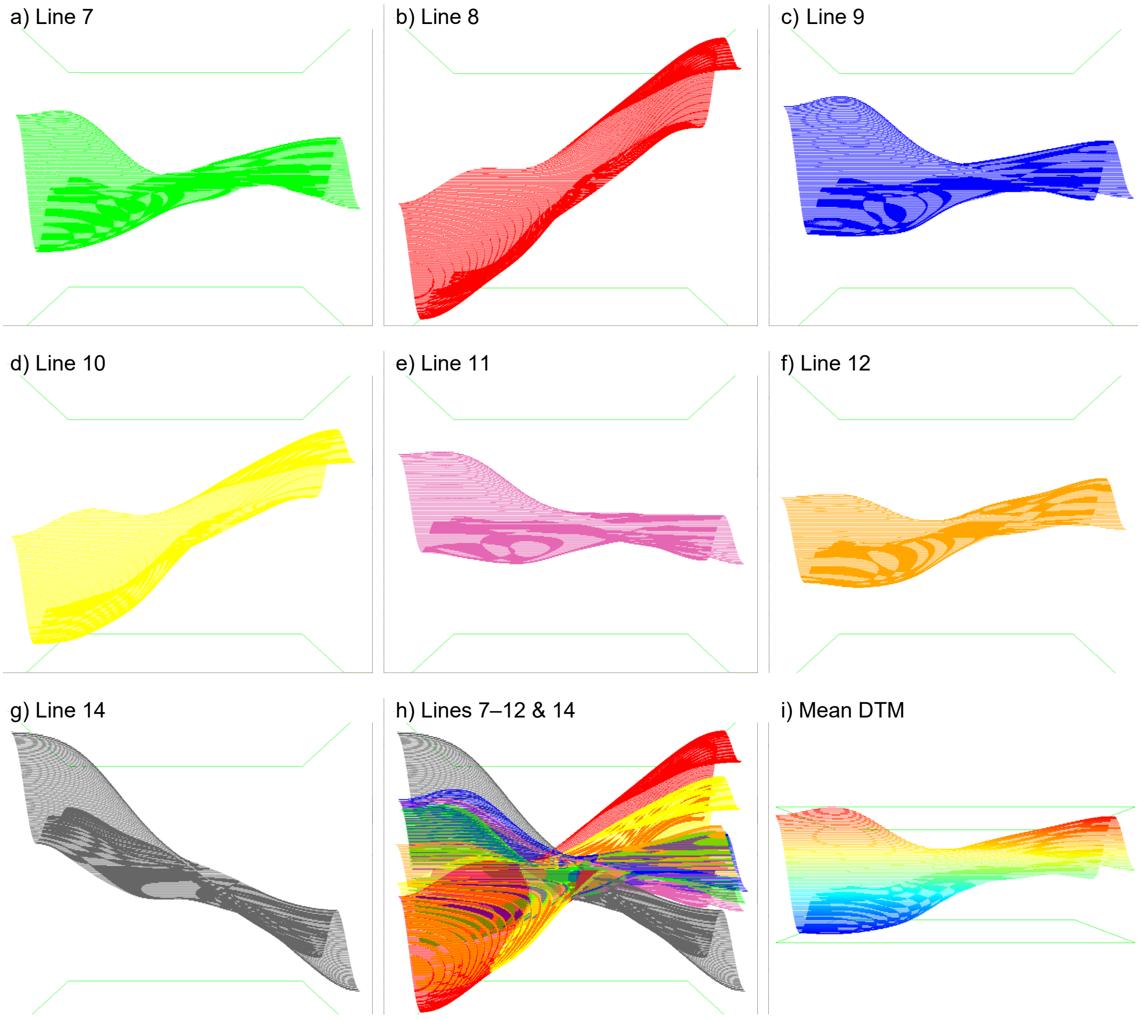
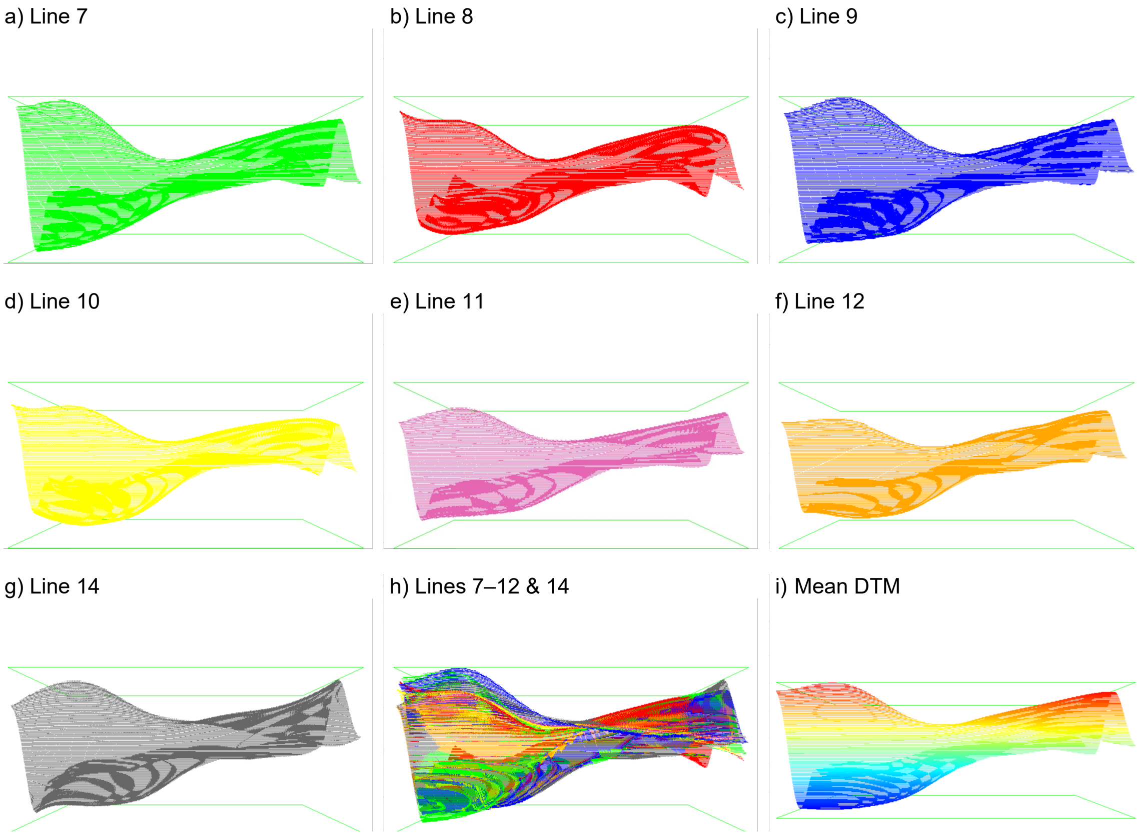


Figure 5.5: Smooth DTMs before ICP: side-view of point clouds from the east. The z-axis (elevation) is magnified $\times 64$ to make the differences noticeable. Point clouds in panels (a)–(h) are coloured by flight line; in panel (i)—by elevation.

the original point clouds, the heights of the point in these clouds are also made to follow very similar distributions. These can be seen in Figure 5.8.



a) Line 7
b) Line 8
c) Line 9
d) Line 10
e) Line 11
f) Line 12
g) Line 14
h) Lines 7–12 & 14
i) Mean DTM

Figure 5.6: Smooth DTMs after ICP: side view of point clouds from the east. The z-axis (elevation) is magnified $\times 64$. Point clouds in panels (a)–(h) are coloured by flight line; in panel (i)—by elevation.

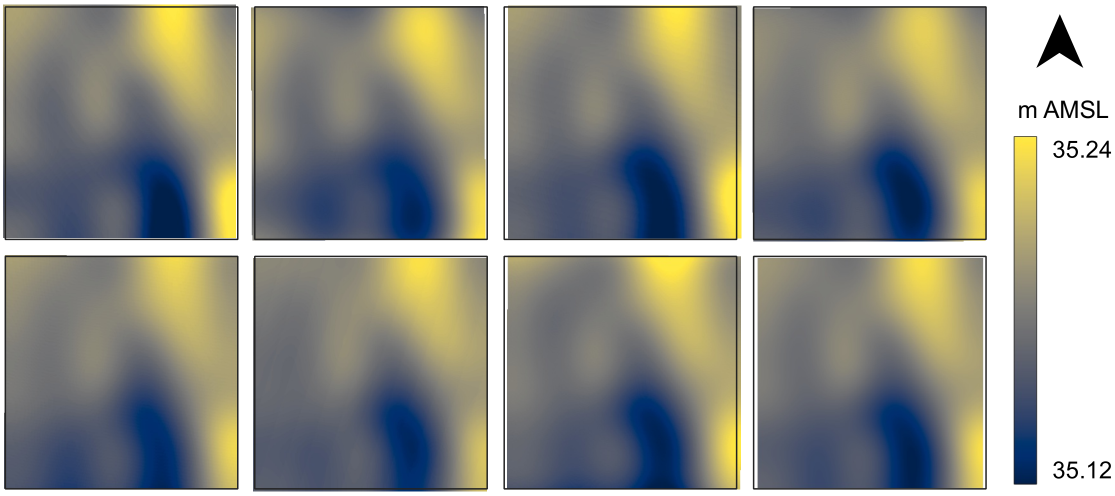


Figure 5.7: Smooth DTMs after ICP: top-view. Top row from left to right: lines 7–10. Bottom row from left to right: lines 11, 12, 14 and the mean smooth DTM.

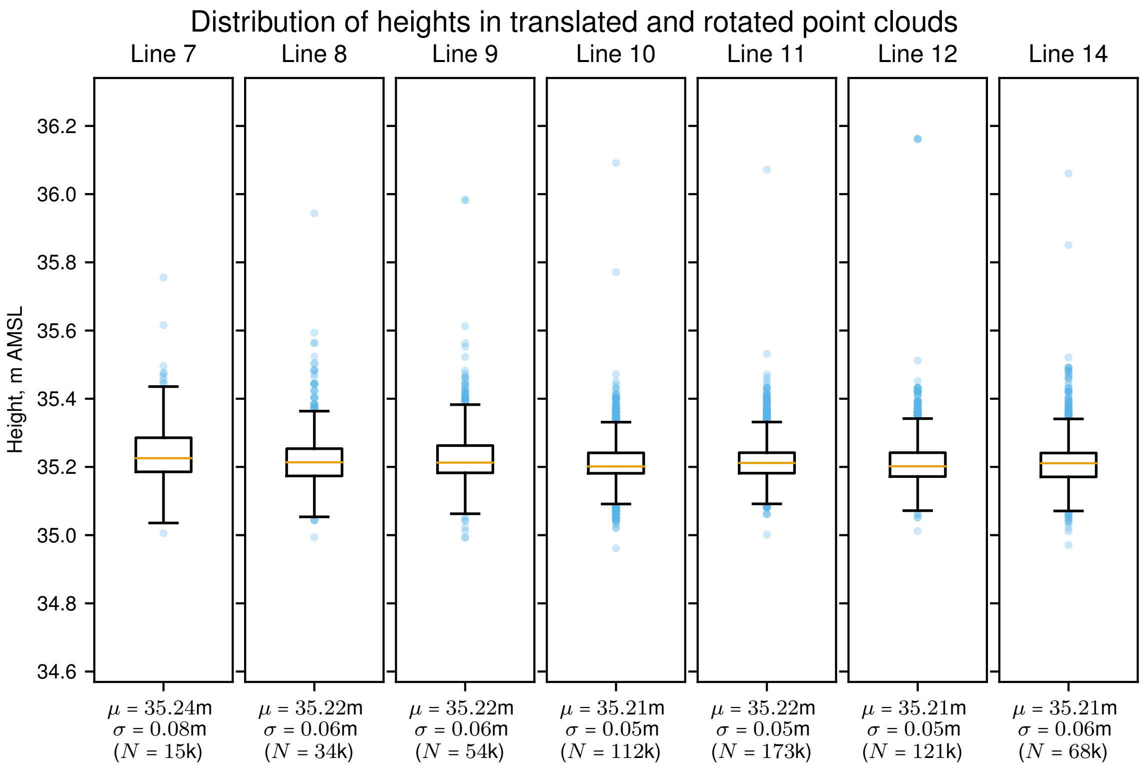


Figure 5.8: Point heights after translation and rotation of input files.

Figure 5.9 shows the low- and high-frequency components side by side. The low-frequency DTM on the left has been obtained by averaging the translated smooth DTMs. The high-frequency DSM on the right is the result of triangulating the surface of pint clouds that were transformed and normalized about the low-frequency surface.

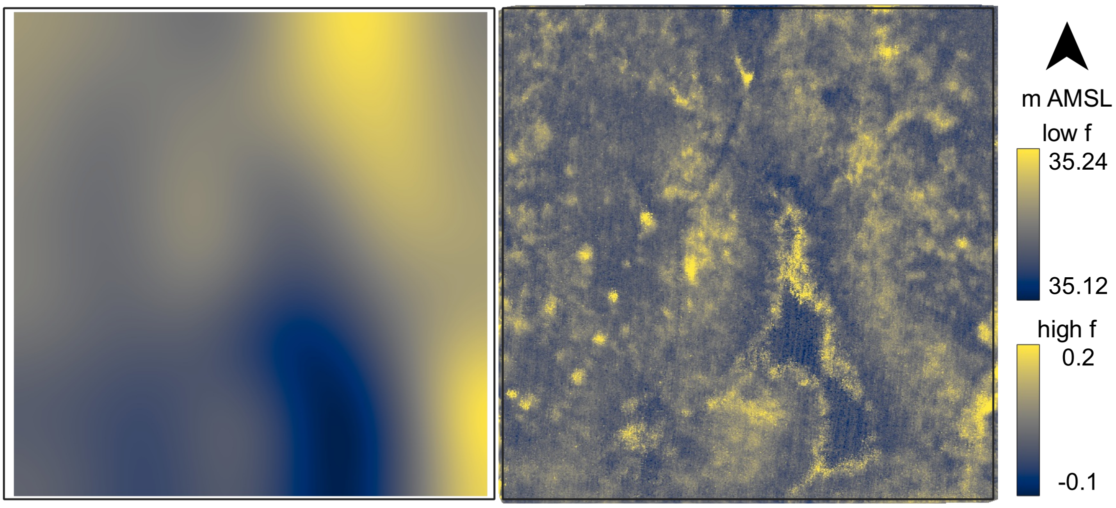


Figure 5.9: Low-frequency DTM (left) and high-frequency DSM (right). Both models have bounding boxes denoting the original 20×20 m ROI.

Finally, adding the two models together, a DSM containing both low- and high-frequency components of the scanned terrain. This is of course analogous to deriving a DSM directly from the now aligned point clouds. Compared to the initial attempt that used the misaligned point clouds and can be seen in Figure 5.3, this model represents the actual bog surface more faithfully. Figure 5.10 presents this final DSM.

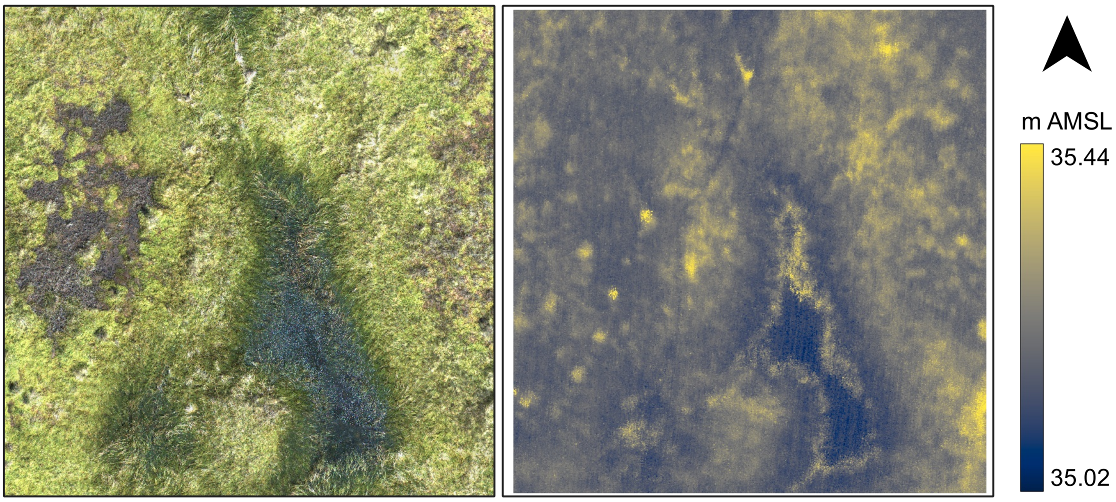


Figure 5.10: DSM produced from the final, aligned, point clouds. Left: orthomosaic of the ROI. Right: DSM of the ROI.

5.3. CONCLUSION AND OUTLOOK

This Chapter presented an algorithm for combining lidar scans from misaligned flight lines to produce high-resolution DSMs of a bog. The herein proposed method used the cross-correlation of the high-frequency structures from each of the scans to derive the initial xy translations needed to align the clouds. Furthermore the low-frequency components were used to calculate the rough height misalignments between the clouds. Finally, the ICP algorithm provided a fine co-registration of the flight lines. It operated on smooth surfaces and its solutions were successfully applied to align the clouds corresponding to each of the considered surfaces.

The viability of the method has been demonstrated for a small 20×20 m ROI. Next a tiled approach could be applied to process the entire scanned surface. Edge artefacts might be unavoidable in this case, but buffering the tiles and making the further processing use the same grid could be a simple yet effective strategy to minimize potential negative consequences.

The derived outputs and the aligned clouds can serve as inputs to further processing techniques. The low-frequency model can be used to detect slow changes in topography, which could help inform hydrological modelling of the area. The high-frequency components can be used to extract topographic and vegetation features on a fine, local scale. Future work should focus on combining the present results with ground truth data and assessing how well they could assist e.g. vegetation mapping tasks.

6. USE CASE 2. *Naturwaldzelle Niederkamp*: SEGMENTATION OF TREES IN A NEAR-NATURAL BEECH FOREST

6.1. INTRODUCTION

6.1.1. NATURAL FOREST CELLS IN NRW

Wald und Holz NRW – the forestry agency in the German state of North Rhine-Westphalia (NRW) – designated various areas throughout the state as representative of the local forest communities. These areas are called *"Naturwaldzelle(n) / NWZ"* (natural forest cell(s)) and in total, there are 75 of them, covering a total area of 1680 ha. Thus all major forest communities in NRW are represented. While their areas range from 1.4 to 110 hectares, almost half of the cells are between 11 and 20 hectares large. The natural forest cells are not used commercially: As of 2021, over half of them have not been harvested for 40–50 years. Any further interference is reduced as much as possible, to allow the areas to develop in a near-natural fashion. Each cell normally features two equal-sized core areas, each being one hectare large. Also generally one of the areas is fenced and the other is open. Here the trees and shrubs with a stem diameter of at least four centimetres (measured at 1.3 metres above ground) have a unique ID and every ten years, abundant data is collected manually for each such tree and shrub (*Wald und Holz NRW*, n.d.-a). Among these data are:

- For living trees:
  - Tree species,
  - Stem diameter at 1.30 m above ground (aka diameter at breast height / DBH),
  - Tree height (sampled for some trees, then calculated with allometric curves),
  - Layer classification (from over- to under-story), etc.
- For dead trees:
  - Classification (lying and standing deadwood, with or without crown),
  - Decomposition degree (from freshly dead to strongly decayed),
  - Diameter and height if standing, etc.

The data collection is labour- and time-intensive, so *Wald und Holz NRW* sought the assistance of the Rhine-Waal University of Applied Sciences to jointly investigate the potential to use UAVs for supporting the data collection efforts. This chapter documents the joint lidar campaigns conducted in one of the natural forest cells and a tree segmentation approach that was developed to aid future data collection campaigns.

The currently fully-manual data collection in the natural forest cells in NRW focuses on individual trees. In order to speed-up integration of UAVs for future monitoring activities, it was decided to also work on the level of individual trees in the context of this thesis. This was evidently facilitated by the possession of the present UAV-borne lidar system, whose low flying altitude and high pulse repetition rate enables the acquisition of high-density data required for differentiating individual trees.

A crucial step in data preparation for extraction of tree-based metrics is the segmentation of trees from the point cloud data. This chapter documents the acquisition of lidar data in one of the natural forest cells and the attempts made so far at segmenting above-ground points into individual trees. It is structured as follows: First, Section 6.2 presents findings from literature regarding lidar use in forestry and segmentation of trees from lidar data. Next, Section 6.3 describes the data acquisition campaign and an algorithm I developed for delineating individual trees. Current results are presented and discussed in Section 6.4. Finally, Section 6.5 summarizes the current chapter and provides an outlook at potential future developments.

6.2. LITERATURE REVIEW

6.2.1. LIDAR USE IN FORESTRY

Area-based vs. tree-based studies Airborne lidar has found extensive applications in forests over the past decades. A summary detailing the recognition of lidar's canopy-penetration properties in the eighties and its subsequent adoption to forestry was given by Nelson, Krabill, and Tonelli (1988). Næsset et al. (2004) outlined a range of studies from Nordic countries that derived tree height, stem volume and other forest metrics at the plot- and stand-levels.

Initial studies were restricted by the low point density of the available instruments. Still, due to the usefulness of area-wide metrics, further research on plot-, stand- and pixel-levels continued. Some examples include: Classification of tree species by combining lidar and hyperspectral data (Dalponte, Bruzzone, & Gianelle, 2012); Monitoring of logging in Amazonia with bi-temporal lidar datasets (H.-E. Andersen et al., 2014); Estimation of fire severity with pre- and post-event lidar and RGB data (Hillman et al., 2021).

Meanwhile, numerous studies have leveraged the evolution of the lidar sensors and the ever-higher available point density to extract information about individual trees. Among the variables estimated were: tree height (Popescu & Wynne, 2004), crown radius (Heurich, 2008) and diameter at breast height (Yao, Krzystek, & Heurich, 2012). Further examples of studies working on a tree level as well as details regarding the algorithms used to segment the trees are given in Section 6.2.

UAVs as a platform for lidar Airborne lidar was traditionally mounted on airplanes or helicopters flying at hundreds of meters above ground. With the miniaturization of the lidar sensors and the widespread use of UAVs, multiple systems and applications have been developed to enable collection of frequent datasets with high point densities. Sensors designed for the automotive industry found their way in pioneering UAV-borne forest inventory systems (Jaakkola et al., 2010; Wallace et al., 2012). Guo et al. (2017) developed an integrated solution consisting of hardware and software for generation of a myriad of tree- and canopy-level metrics. Survey-grade lidar units with dedicated UAVs also gained popularity in forest applications (Brede et al., 2017).

6.2.2. TREE SEGMENTATION ALGORITHMS

There are numerous algorithms for extraction of trees from lidar data. An overview of current and established methods is given by Camarretta et al. (2020). The present section outlines some of the main categories and developments from the last 20 years.

The earlier tree segmentation algorithms usually operated on lidar-derived CHMs[50]. Hyyppä et al. (2001) developed a fixed window-size local maxima extraction technique to find treetop candidates in smoothed CHMs. Combined with a decision tree to identify the neighbouring pixels belonging to each candidate, this technique successfully delineated tree crowns and extracted individual-tree measurements that were then extrapolated to stand-wise statistics for a forest consisting mainly of Norway spruce (*Picea abies*) and Scots pine (*Pinus sylvestris*). Dalponte and Coomes (2016) applied this method to successfully delineate individual crowns. By identifying the tree species with hyperspectral data, they used allometric relationships to estimate carbon stocks in a managed forest situated in the Italian Alps and dominated by *P. abies*, also containing further coniferous and deciduous tree species.

Chen et al. (2006) implemented a seeded watershed segmentation algorithm to delineate crowns of individual trees in cannopy height models which were subjected to maximum filtering,[51] followed by Gaussian blurring and inversion.[52] Among this study's innovations was using a variable-size window to search for local maxima in the modified CHM. These maxima were identified as treetops and further used to seed the above algorithm. The authors applied the method to segment tree crowns in a savanna woodland consisting mainly of blue oaks (*Quercus douglasii* H.&A.).

Silva et al. (2016) segmented crowns of individual longleaf pines (*Pinus palustris* Mill.) in a managed forest. They first detected treetops as local maxima in both raw and smoothed cannopy height models, then combined centroidal Voronoi tessellation with thresholding as well as heuristics relating crown width and tree height to isolate the pines from the CHMs. Overall, trees were detected with high accuracy in plots with $< 70\%$ canopy cover.

Advances have also been made in segmentation of trees directly from the point clouds. Reitberger et al. (2009) presented a novel algorithm that used a CHM-based watershed segmentation as an initial rough step, but refined its results by automatically finding tree stems and then crowns in the point clouds. By considering one candidate cluster at a time, the authors used the vertical point distribution to differentiate between stems and crowns. Then a RANSAC[53] approach was applied to the potential stem points to detect one or multiple stems. Finally, a normalized cut was used to assign the crown points to the corresponding stems. The algorithm excelled at identifying smaller trees that were shadowed by larger ones and therefore challenging for raster-based methods.

---

[50] cannopy height models

[51] Maximum filtering is achieved by replacing pixel values with the maximum value in a small specified neighbourhood.

[52] After inverting a raster, the tree crowns would resemble valleys. It can be achieved by multiplying the input with a negative scalar.

[53] random sample consensus

The normalized cut algorithm can combine multiple features to find an optimal partition in a dataset. It was first proposed for 2D image segmentation by Shi and Malik (2000). Then, Reitberger et al. (2009) adopted the algorithm for finding individual trees in point clouds. This in turn led to further studies that successfully applied the method to segment trees from full-waveform-lidar point clouds (Yao et al., 2013; Amiri et al., 2016, 2019).

Li et al. (2012) developed an algorithm that started by assuming the single highest point in a normalized point cloud was a treetop. It then progressively grew the tree by adding points which lay lower than the current solution, but within a certain distance to the points already in the tree. Next, it removed the points belonging to this tree from the point cloud and the whole procedure was repeated with the next highest point. This simple approach segmented trees with 86% recall and 94% precision in a mixed conifer forest.

Vaughn, Moskal, and Turnblom (2012) demonstrated an algorithm that operated on voxels instead of raw point clouds. By transforming the data into a voxel space, the authors could significantly reduce the amount of computation required. The developed algorithm processed entire voxel slices at once, also going from top to bottom. New voxels were added to existing clusters based on a number of conditions, some of which dictated what happened to the voxel under consideration when multiple suitable candidate clusters were detected. While satisfactory results were obtained for coniferous species, some hardwood trees required manual editing of the identified clusters. Nonetheless, after applying the Fourier transform to the original waveforms associated with each detected crown, the authors used a support vector machine to correctly classify 111 out of the 130 trees belonging to five different species.

Similarly to Reitberger et al. (2009), Amiri et al. (2016) also relied on an initial watershed segmentation of the CHM. However, their method differed in that mean shift clustering was used to detect “super voxels” in the point cloud – meaning closely situated points which were likely to belong to the same tree. This way, significant savings in the computation cost of the following steps could be made. Next, a normalized cut algorithm was applied to the detected clusters, using lidar pulse width and previously detected maxima in the canopy to merge the “super voxels” into complete trees.

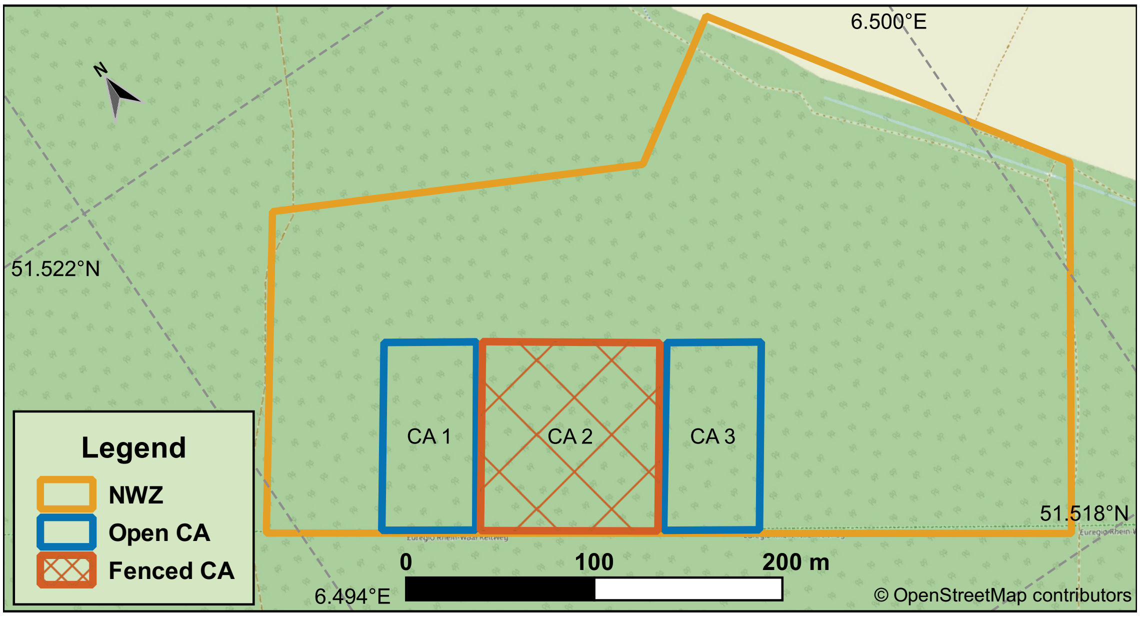
Wallace, Lucieer, and Watson (2014) conducted a comparative study and assessed five different tree segmentation algorithms, considering CHM-based approaches, as well as voxel and point-cloud based ones. Among the candidates, they found a hybrid approach to work best. Local maxima from a blurred CHM were used to seed a k-means clustering algorithm, which successfully identified 98% of the trees in a four-year-old *Eucalyptus globulus* plantation.

A more recent development was the use of deep-learning approaches. Windrim and Bryson (2020) extracted features from voxelized point clouds, such as vertical point density, maximum height and average return on an xy-grid and mapped them to a three-band raster. Then they could apply a classical deep learning approach wherein a Faster-RCNN classifier was trained to produce 2D bounding boxes around trees in this three-band representation of point clouds. Furthermore, a 3D fully convolutional neural network was used to segment individual trees into crowns and stems.

6.3. METHODS AND MATERIALS

6.3.1. STUDY AREA AND DATA COLLECTION

An 8.2-ha natural forest cell (*NWZ* 43 *Niederkamp*, *Wald und Holz NRW*, n.d.-b) situated in Kamp-Lintfort was selected as the pilot location for collecting data with the university's UAV-borne sensors. The forest cell is dominated by common beech (*Fagus sylvatica*), with a few sessile and common oaks (*Quercus petraea* and *Q. robur*). Small areas in the understory are covered by common holly (*Ilex aquifolium*). In *NWZ Niederkamp*, the general rule of having two core areas has been somewhat bent. Instead of one continuous non-fenced area, two 0.5-ha core areas had been established: one and three. They are bordering with the fenced core area two, which is as usual, one hectare large. The location of the three core areas within the *NWZ* is shown in Figure 6.1.



NWZ 43 Niederkamp: Location of core areas

Figure 6.1: Location of the three core areas within the natural forest cell (*NWZ*) Niederkamp.

The cell is surrounded by similar forest from all sides, except the north-east side, where it borders with an agricultural field. With the farmer's permission, two lidar flight campaigns were conducted from the field. Thus take-off and landing were facilitated and good visual contact was maintained with the UAV. The flight campaigns took place in late 2020.

Using the observations from the first campaign on $30.11.2020$ (Niederkamp 1), which was aborted due to weather conditions, the flight parameters for the next campaign on $18.12.2020$ (Niederkamp 2) were optimized. For instance, because laser returns were obtained from distances longer than initially anticipated (Velodyne LiDAR, Inc., 2018a), the flying height was increased from 80 m above ground [54] in the first campaign to 90 m in the second one. Moreover, data from the Niederkamp 1 campaign was used to perform the system's boresight calibration. The campaign Niederkamp 2 was successfully completed and lidar as well as RGB data were collected for the entire cell. The flight paths of the two campaigns and the general location of the natural forest cell were shown in Figure 2.2.

---

[54] Without ground-following mechanisms, the flying height is always stated above the take-off point.

6.3.2. SEGMENTATION ALGORITHM

An initial attempt to segment the acquired point clouds with the `lidR` package (Version 3.1.1; Roussel et al., 2020) for `R` (Version 4.0.5; R Core Team, 2021) was made. Among the available algorithms, those developed by Li et al. (2012), Dalponte and Coomes (2016) and Silva et al. (2016) were used. Unfortunately, no combination of parameters that would yield satisfactory results with any of the methods was found. Instead, an alternative algorithm was developed.

In contrast to the more widespread paradigm where trees are found based on identifying the high points, the possibility to detect tree stems instead and let them grow upwards was investigated. After all, every tree needs to start growing from the ground, but not every one necessarily produces a prominent treetop. This was especially facilitated by the data collection in the leaf-off condition, so that a considerable number of returns was obtained from under the canopy.

Description of the segmentation algorithm The developed algorithm operates directly on a normalized point cloud. For each point the algorithm creates and keeps track of an attribute called cluster-ID. Initially, all points above ground have the ID 0 and the ground points: ID -1. The aim is to assign the same ID > 0 to all points that make a tree together, so that at the end every tree has a unique ID. In order to speed-up the numerous spatial queries, the points were organized in a k-dimensional tree, with $k = 3$. The tree was implemented with the `spatial.cKDTree` function from the `SciPy` package for `Python` (Virtanen et al., 2020).

The procedure starts at a certain height above ground (h_{min}) and for every point in a slice of height Δh, finds neighbours that are located in an ovoid, such that the current point is slightly above the ovoid's base and the ovoid's long axis looks upwards. This shape was chosen to stimulate the growth of clusters mainly upwards, but also include the canopy. The shape and dimensions of the ovoid are controlled by two parameters: height and pointedness. The algorithm decides the label of both the current point and the unlabelled neighbours based on the membership of the found neighbours and of the point under consideration. For instance, if the current point already belongs to a cluster, the unassigned neighbours get the same cluster ID. Otherwise, the label of the most prevalent cluster among the considered neighbours wins.

When no adequate cluster is identified among neighbours, the points are assigned a new cluster-ID. Once all the points from the current slice have been exhausted, the points in the next slice are processed. This continues until all points above h_{min} are looked at.

The above procedure is repeated twice: the first pass is called initial clustering. The second – refined clustering. Only in the first pass can new clusters be created and only as long as the point being considered is lower than a maximum height h_{max}. The second pass does not create new clusters but only tries to expand those found in the first pass.

Because the first pass is usually more conservative and does not grow the clusters excessively, the second pass uses a wider ovoid usually identifies edges of canopies or long branches not detected initially.

Evaluation of the segmentation algorithm The algorithm has been tested on data from the core area one (See Figure 6.1). A point cloud containing the 50×100 m area surrounded by a 10 m buffer has been extracted from a single flight line. This cloud covered an area of 0.88 ha and had an average point density of 78 pulses/m^2. The cloud was then normalized by classifying the ground points and setting each point's height relative to the ground surface. These steps were accomplished with the `lasclip`, `lasground` and `lasheight` utilities from `LAStools` (Isenburg, 2021).

The initial clustering was performed with the following parameters: $h_{\min} = 2.0$ m, $h_{\max} = 15.0$ m and $\Delta h = 0.1$ m. The ovoid had a height of 3.0 m and a diameter of 1.9 m at its widest. The refined clustering used an ovoid with a height of 3.0 m and a diameter of 2.8 m at its widest. These values were chosen in order to balance a few different requirements:

- Enable the algorithm to correctly "jump" from a stem to its crown where gaps were encountered in a tree's height profile;
- Avoid "jumping" onto the crowns of other trees;
- Grow mostly upwards in the first pass;
- Complete the crown in the second pass, with more lateral growth.

The values were fixed after running the algorithm on data from another flight line and a different part of the natural forest cell. No attempt to optimize any of the parameters were made when working with data from core area one.

After running the algorithm, the resulting point cloud was inspected visually. First, the entire point cloud was scrutinized. Then different sections were cut and looked at and finally, individual clusters were extracted and visualized with `lasview` from `LAStools` and `CloudCompare` (CloudCompare, 2020). This visual examination attempted to assess whether the results were plausible.

Next, the distribution of tree heights as obtained by segmenting the point cloud was compared with ground truth data. *Wald und Holz NRW (WuH)* provided a dataset collected manually in 2019 and containing, among other variables, the height of the trees in core areas one and two. Included were both the live trees and the standing dead trees (snags). The height of some trees was indeed measured,[55] but that of the others was calculated using allometric relationships. Both the complete dataset (*"Reference"*) and the subset containing only measured trees (*"Measured"*) were considered in the following.

In the *WuH* dataset, the trees were numbered, so that the parameters were assigned to individual trees, which can then be found again after 10 years. However, the position of the trees was not specified. Therefore, only the distribution of heights was compared.

Out of all the clusters identified by the segmentation algorithm, those that had at least 50% of their 2D convex hull within the core area were selected, eliminating those situated mostly or completely in the buffer zone. A further condition was that the clusters either had at least 100 points, or had at least ten points but also a ratio of height to "mean width" of at least five. To find the "mean width" of a cluster, its projection onto the xy plane was found. Then the area of the projection's 2D convex hull was calculated. The "mean width"

---

[55] The measurement method was not specified.

was the square root of this area. This metric was used because numerous snags could be seen in the point cloud. They had few laser returns but a very prominent tall shape. To estimate the heights of the identified clusters, the elevations above ground of their highest points were considered. This dataset is referred to as *"Segmented"*. The histograms and the summaries of the considered datasets are presented in Section 6.4.

6.4. RESULTS AND DISCUSSION

Visual assessment Out of $473\,817$ points situated at least 2 metres above ground, 97% were assigned to one of 846 clusters. However, 98% of the assigned points were contained in just 108 clusters containing at least 100 points each. Also, 92% of all the assigned points were contained in just 74 clusters which had a size of at least 2000 points.

Figure 6.2 presents the results viewed from above. In the first two panels, it shows the processed point cloud cropped to the core area (without the 10m–buffer). In the last panel, the trees whose height was compared to the ground–truth are shown. It is noteworthy that some of their crowns are not contained entirely within the core area.

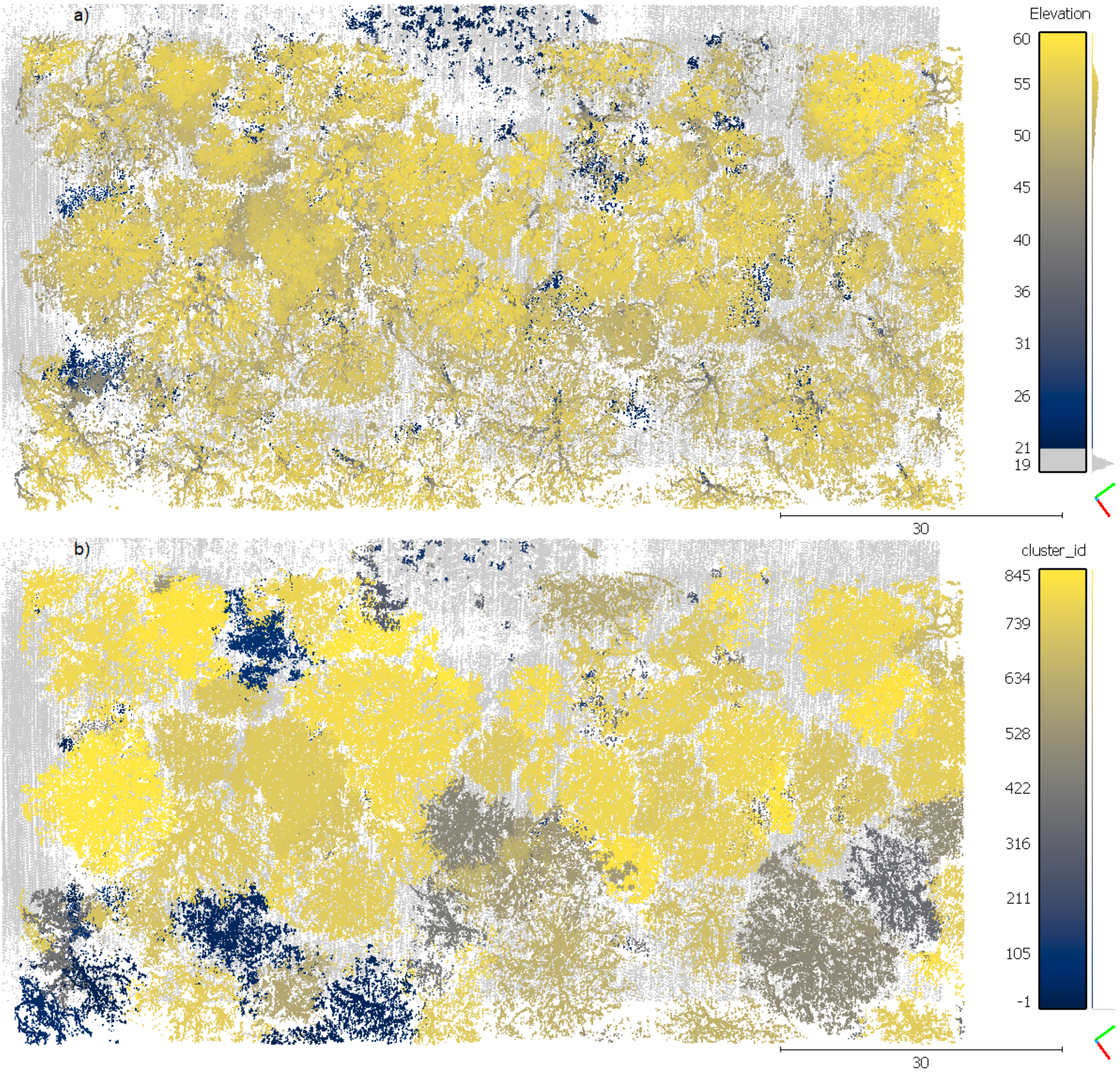


Figure 6.2: Segmentation results for core area 1 (50×100 m). Top–view. (a) Point cloud coloured by height above ground. All points below two metres are grey. (b) The same point cloud coloured by cluster–ID. IDs -1 (ground) and 0 (unclassified) are greyed out. *(cont.)*

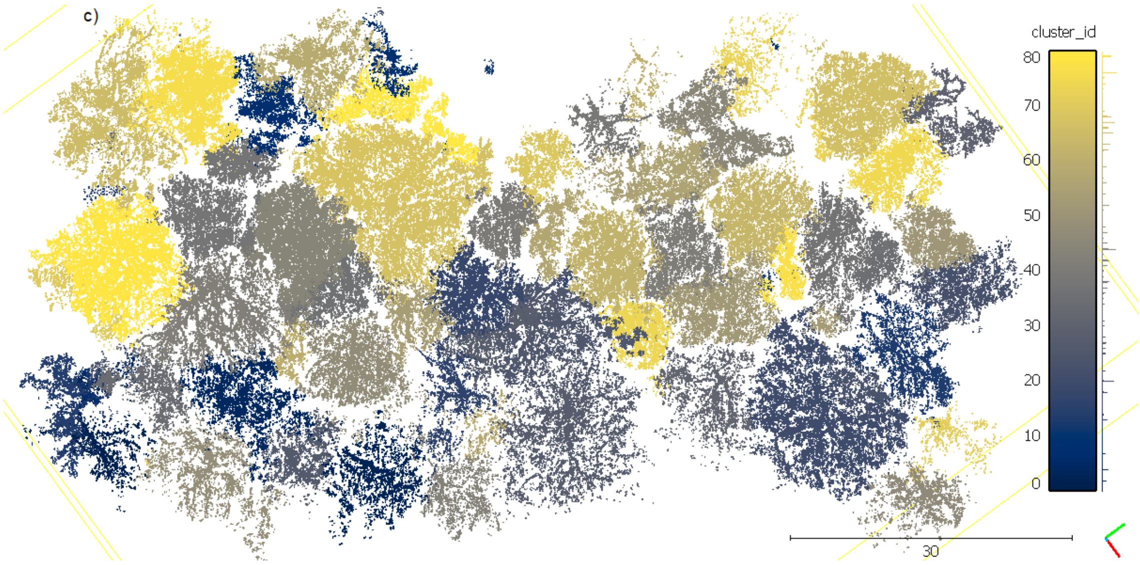


Figure 6.2: Segmentation results for core area 1 $(50 \times 100$ m). Top–view. (c) A point cloud containing only clusters with at least 2000 points.

Figure 6.3 shows the same data viewed roughly from south-west.

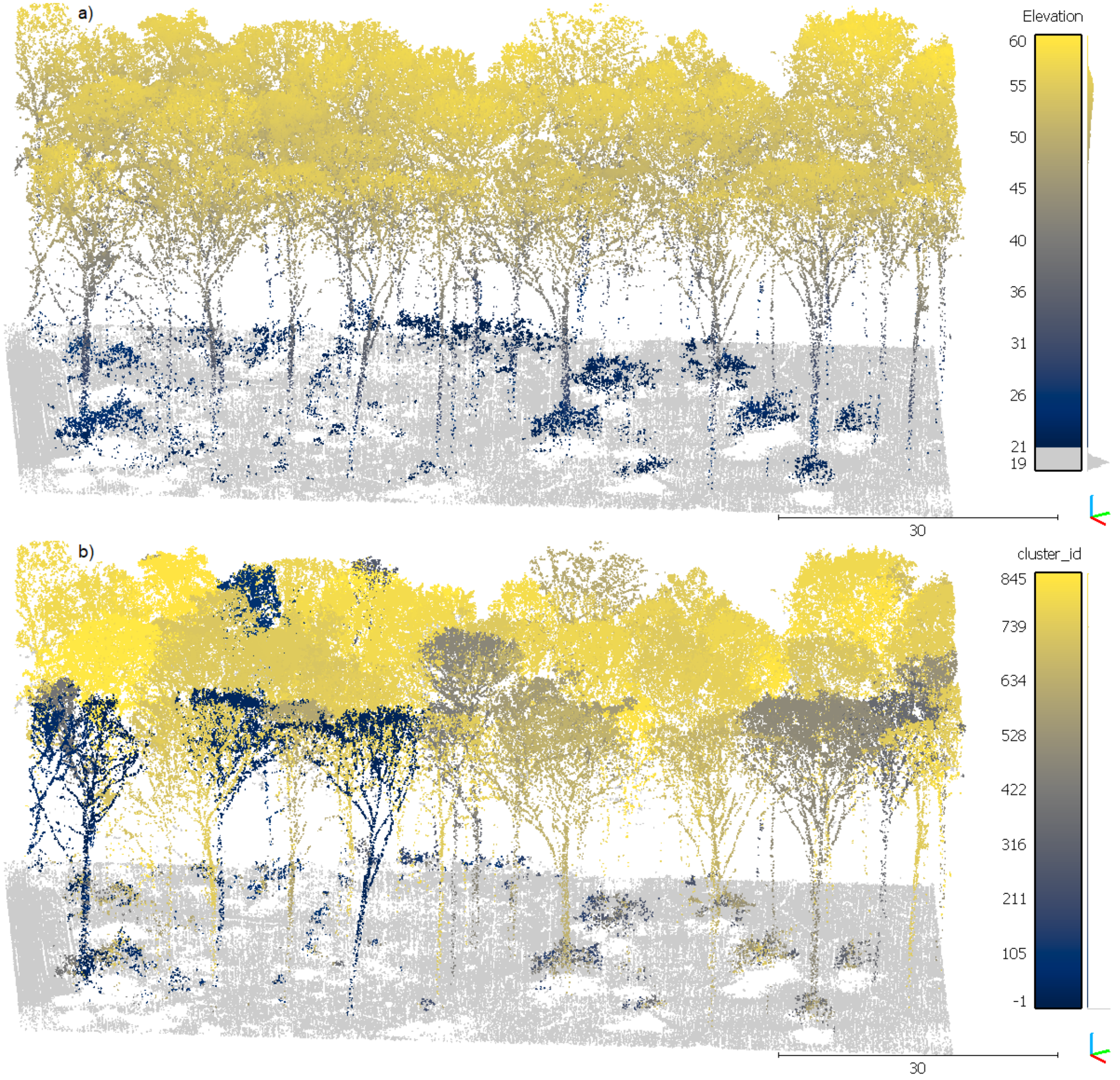


Figure 6.3: Segmentation results for core area 1 $(50 \times 100$ m). Side–view. Same point clouds and colouring schemes as in Figure 6.2. *(cont.)*

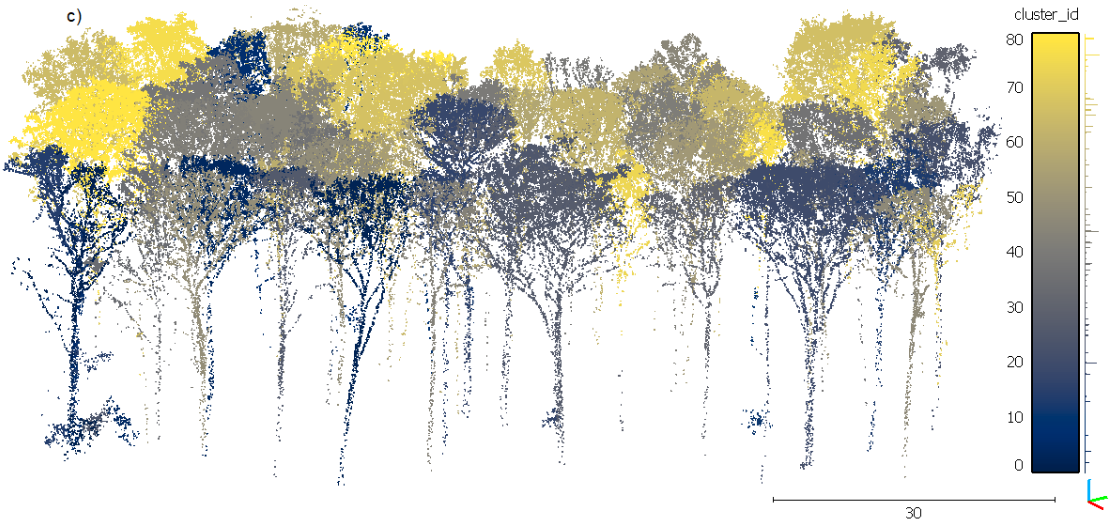


Figure 6.3: Segmentation results for core area 1 $(50 \times 100$ m). Side–view.

It can be seen from the Figures 6.2 and 6.3 that most large trees are delineated correctly. However, the algorithm still suffers from over-segmentation, i.e. more clusters than trees are produced. While it would be best to avoid over-segmenting the point cloud in the first place, filtering out the particularly small clusters can be a quick and effective fix. This becomes obvious when the distribution of cluster sizes is considered.

Figures 6.4 and 6.5 show two clusters that have been extracted from the segmented point cloud. First, an example of a well–segmented tree can be seen in Figure 6.4: The structure of a tree can be traced and it seems complete. This cluster plausibly represents one single tree.

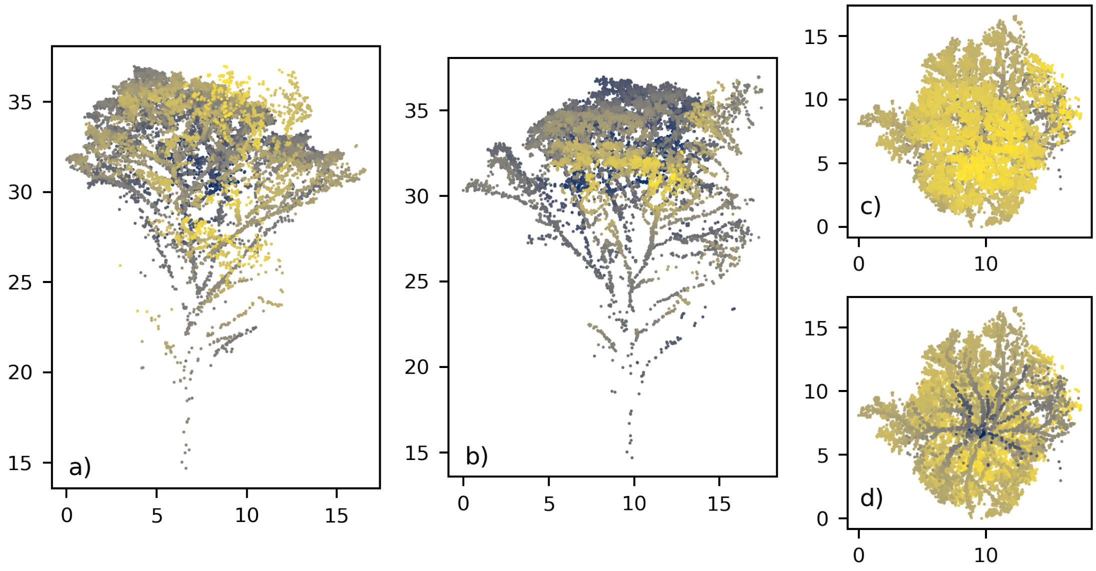


Figure 6.4: Example of a well–segmented tree. Colour encodes depth in the first two panels (dark = far) and height – in the last two (dark = low). a) View from south-east. b) View from south–west. c) View from above. d) View from below.

In contrast, Figure 6.5 shows an example of grave under-segmentation. While a rough tree structure can be inferred from the image, it is also clear that multiple stems were included in this cluster. In addition to two complete trees, the crown of a third tree was also incorporated into the result.

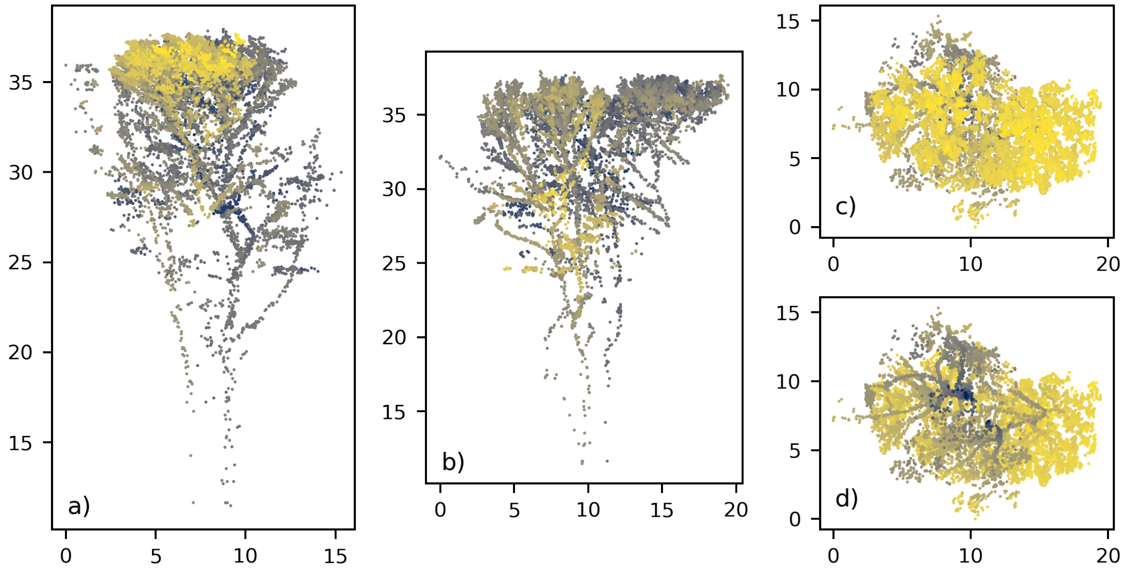


Figure 6.5: Example of grave under-segmentation. Colour and perspective are to be interpreted like in Figure 6.4. Note the problems with this segmentation: two stems (panels (a) and (b), lower half); extra crown (panel (b), right–upper corner). In panel (d) it can further be seen that there are multiple origins for branches (two stems at $(10, 9)$, $(12, 6)$ and third crown at $(16, 7)$).

In the example from the Figure 6.5, where the two trees were quite close to one another, the algorithm was too eager to grow, especially sideways. I suppose this tendency could have been curbed by adjusting the shape of the ovoid, but the present analysis is rather a proof of concept, so no further optimizations have been done yet.

Comparison of height distribution with ground truth In the reference dataset, there were 127 trees that were supposed to be located within the core area 1. However, only 88 of them (*"Reference"* dataset) could be located during the data collection campaign from 2019. The explanation provided for the missing trees was that either they fell, or could not be reached because of holly. If the latter were the case – a segmentation of remotely–acquired lidar data could indeed support the field work.

Out of the identified trees, 49 had their height measured (*"Measured"* dataset), while the height of the other 39 units was calculated using allometric relationships.

Based on the conditions outlined in Section 6.3, i.e. cluster size and height–to–"mean width" ratio, 81 clusters had been identified in the *Segmented* dataset.

Table 6.1 gives a summary of the three datasets. The distribution of these heights is also shown in Figure 6.6.

Table 6.1: Summary of tree heights in Core Area 1: field–data vs. segmentation results.

	Reference	Measured	Segmented
count	88	49	81
mean	31.8	33.2	31.2
std	11.9	10.2	10.7
min	1.6	6.2	4.4
25%	28.8	30.8	32.5
50%	37.9	38.2	36.4
75%	39.6	39.4	37.4
max	44.5	43.6	40.3

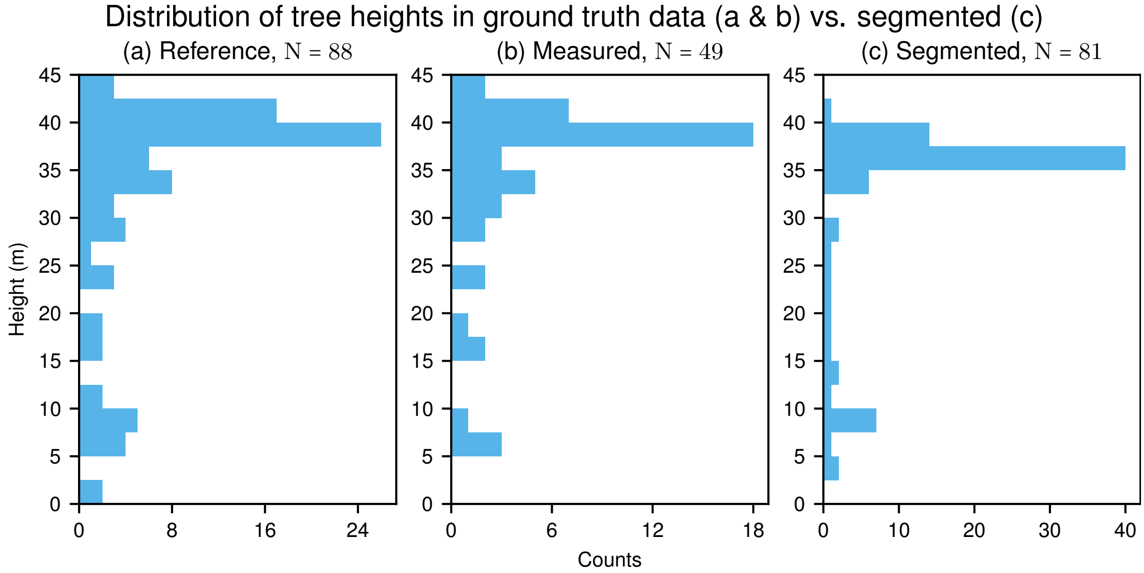


Figure 6.6: Distribution of tree heights in Core Area 1: field–data vs. segmentation.

Based on Table 6.1 and Figure 6.6, the distributions of the three datasets seem to match rather well. However a number of differences can be observed: The mean, median and maximum height estimates are lower for the segmented trees. But these discrepancies are not caused by the algorithm itself. The highest point in the analysed point cloud was situated at 40.6 m above ground, so the algorithm underestimated the result by only 0.3 m. There is a further four–meter difference between my results and the reference data. An investigation into the cause of this error is still pending.

A positive result is that the number of detected trees comes close to the manual count. Assuming that after filtering out the clusters, trees were rather missed instead of counting too many, the number of false negatives would be seven and that of false positives – zero. This would translate into 1.00 precision and 0.92 recall which would correspond to an F_1 metric of 0.96. These of course are the most optimistic estimate. But, considering that the example from Figure 6.5 consists of ≈ 2.5 trees and was included in the dataset as one tree, the actual metrics are definitely worse. I did not attempt to find out how much worse, because this would have required validating the results with a certain confidence. As stated before, providing a proof of concept was sufficient at the present stage. Section 6.5 reflects upon the merits and drawbacks of the current approach and explores possibilities to improve the results.

6.5. CONCLUSION AND OUTLOOK

The point-cloud-based tree segmentation algorithm presented in this Chapter demonstrated plausible results in a near-natural beech forest. Up to now, the assessment was qualitative, rather than quantitative. Future work should involve post-segmentation processing, such as extraction of tree metrics other than height. Also, for meaningful quantitative analysis of the results, these should be carefully checked against field data.

Once the accuracy of segmentation can be reliably quantified and tree metrics can be extracted and verified with ground-truth, a comparison of performance with other algorithms would be appropriate. I suppose that, by developing this algorithm, a better understanding of the motivation behind the methods employed in the papers mention in Section 6.2 was gained. Therefore, selecting suitable parameters in e.g. `lidR` (Roussel et al., 2020) should be a more doable task than it was before working on tree segmentation.

The idea to segment trees from the base instead of the top has some merit and is worth pursuing further. A weakness of the employed dataset was that many tree trunks had vertical gaps of a few meters. One consequence was that some crowns were not assigned to their correct stems and were instead incorporated into neighbouring clusters.

A possible solution could be using data from multiple flight lines. In fact, 17 flight lines generated returns from the core area and its buffer, leading to a total point density of 478 pulses/m^2. But data from overlapping flight lines was not combined because the boresight calibration had not been adequately solved yet. It can be seen in Figure 6.7 that data from multiple flight lines produced more complete but also blurrier trees, overestimating the stem diameters and letting the crowns grow more into each other.

Because the boresight parameters had not been estimated well enough yet, it was decided to only work with one flight line for now. This way, no additional difficulty had been presented to the algorithm. But I am optimistic about decreasing the boresight errors considerably, which will in turn allow obtaining coherent high-point-density scans. Figure 6.7 demonstrates that the tree stems would in turn be way better represented in the data, especially due to oblique scanning angles.

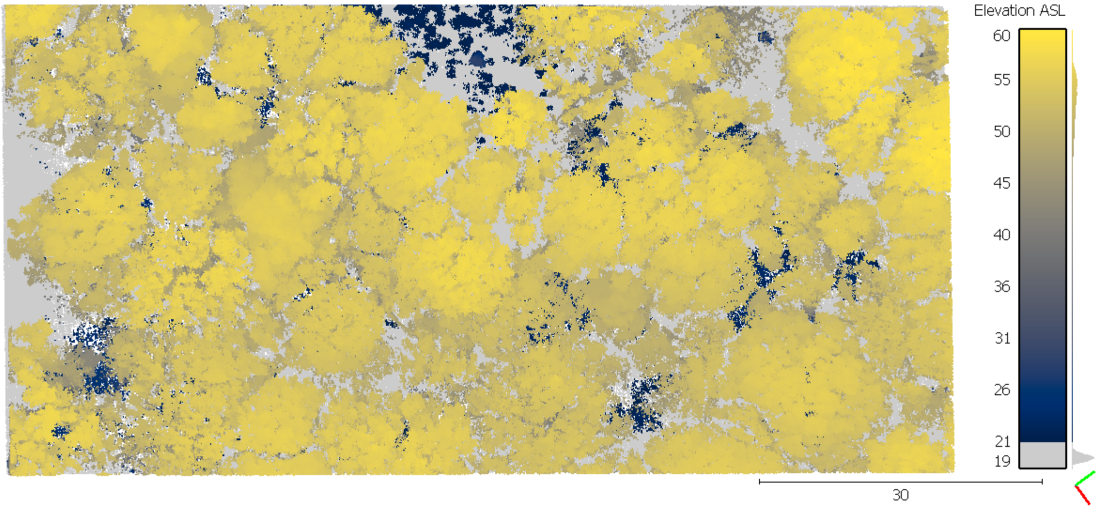


Figure 6.7: Core Area 1. Multiple flight lines combined. Top-view (Compare to Fig. 6.2.a). *(cont.)*

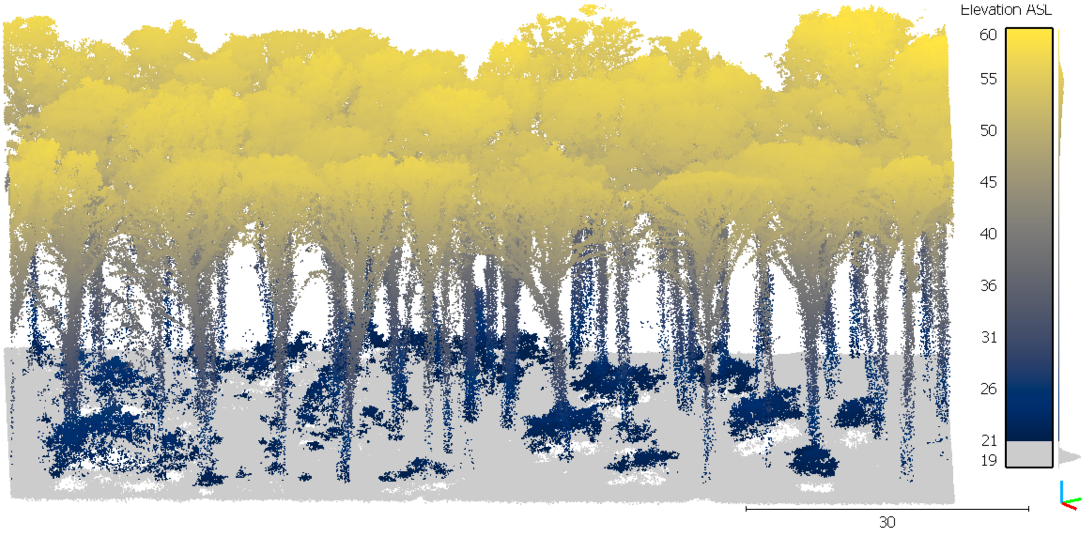


Figure 6.7: Core Area 1. Multiple flight lines combined. Side-view (Compare to Figure 6.3.a).

The fact that the point density at the stem height is disproportionately increased by combining data from multiple flight line is further confirmed by Figure 6.8. It shows the normalized distribution of the points at all heights above two metres for a cloud obtained from one line (*“single”*) vs. another cloud combining data from 16 flight lines (*“multiple”*). The data below two metres was not included in the Figure, since the relatively minute differences between the two density plots are harder to spot when the large number of ground returns and low vegetation is included.

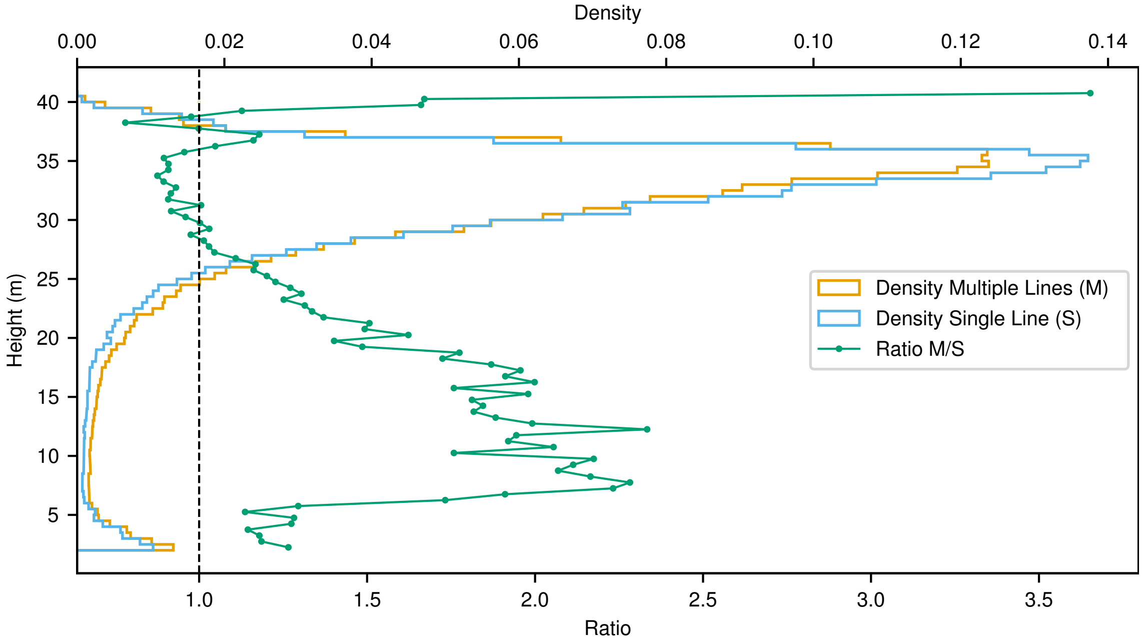


Figure 6.8: Distribution heights in point cloud of Core Area 1.

The marked line in Figure 6.8 shows the ratio of the two normalized distributions. It is larger than one where the fraction of returns in a given height bin is higher in the case of multiple lines vs. a single line. When it is lower than one, the opposite is true. It can thus be seen that combining data from multiple flight lines increases the proportion of points lower than ≈ 27 metres (stem returns) or higher than ≈ 40 metres (treetops), but decreases the proportion of points constituting the bulk of the point cloud: between $\approx 27-36$ metres (crown returns).

The better representation of stems in combined flight-lines data, as seen in Figures 6.7 and 6.8, is in part due to the oblique scanning angles. However the absence of leaves might have contributed even more, by enabling penetration of laser pulses deeper than just the crowns. The collection of another dataset in *NWZ Niederkamp* during the leaf-on season is being planned for the end of this summer. When this data becomes available, it would be interesting to assess how the height-distribution of points changes and how the performance of the herein developed algorithm is affected by the presence of leaves.

Assuming the lidar penetration in the upcoming leaf-on dataset is good enough to allow obtaining sufficient stem returns, then the algorithm would not be restricted to data acquired in a specific season. Under these conditions, there are certain ideas that could be incorporated to make it more robust.

For instance, stem detection using point height distribution (Figure 6.8) and a RANSAC-based stem line reconstruction (Reitberger et al., 2009). This method of consistently finding tree stems, combined with the high point density that the herein presented system can generate when multiple flight lines are used, could greatly aid the first stage of the algorithm. By restricting each cluster to exactly one such stem, a much more reliable segmentation than the current one could be achieved. Over-segmentation would be curbed because no clusters could be formed besides the existing stems. So far under-segmentation was only observed for the current algorithm when stems were not correctly identified and their crowns could be taken over by other clusters (Figure 6.5). Therefore if all stems were to be found, then under-segmentation would also be unlikely.

Multiple crowns were attributed to the same cluster in the example presented in Figure 6.5. This particular mis-segmentation could be identified in e.g. a CHM, so a subsequent normalized cut step could be added to the algorithm for solving these cases. To the best of my knowledge the normalized cut has been applied to segment trees only from full-waveform-lidar data so far (Reitberger et al., 2009; Yao et al., 2013; Amiri et al., 2016, 2019). These datasets were particularly suited for the algorithm due to their feature-richness (multiple returns per pulse, as well as pulse width and height for each return). Still, the applicability of the algorithm to discrete-return lidar and features stemming from e.g. simultaneously-collected RGB-data or lidar return intensity could be investigated and applied as a post-processing step for particularly difficult clusters.

Summarizing, this chapter presented a point-cloud-based tree segmentation approach. Its performance has been loosely evaluated in a 0.5 ha forest dominated by beeches. The algorithm's primary contribution is the point clustering from the forest floor towards the canopy top, which is different from currently established approaches. This method might have an advantage in high-density datasets with numerous stem returns, since it could effectively prevent both over- and under-segmentation. Also, a few strategies for increasing the robustness of the algorithm have been presented and the need to rigorously evaluate its performance and compare it to existing methods has been established.

7. REFLECTION AND OUTLOOK

The present thesis documented the development of a UAV-borne lidar system. Off-the-shelf components have been used when available. Both the hardware and software developed to tie the components into a functioning whole have been thoroughly explained herein. Furthermore, the hardware schematics are distributed via the Appendix A of the current thesis. Also the entire software suite, from data acquisition, to visualization and processing into georeferenced point clouds had been made publicly available (Dogotari & Rostalski, 2021).

On the hardware side, special attention had been given to designing the power electronics. These combine efficient circuits on one hand with several layers of safety on the other, such that both the sensors and power sources (e.g. the carrying UAV) are protected from several scenarios of user error. Moreover, the system operation has been designed with user friendliness in mind: Cumbersome power-up or sequences for the APX have been eliminated. Further considerations have been given to using standard connectors and protocols, thus facilitating the swapping of individual components.

Regarding the software developed, the acquisition and processing of data have been separated completely. While the first uses time-tested techniques for ensuring reliable field operation, the latter takes advantage of modern multi-core processors and in-memory computing to speed up the results. Most of the programming had been done in Python 3, which is a modern and extremely maintainable language. Moreover, the much faster computations in C, available via the NumPy library, were used to boost performance.

A procedure for boresight calibration using retro-reflectors has been developed. So far only rough misalignments could be corrected, despite best ongoing efforts. However the likeliest sources of error have been identified and solutions to both improve the current approach and apply other boresight calibration methods were proposed.

The current paper synthesises approximately three years that were invested into developing, integrating and using the system. As such, given the open-source nature of the deliverables, it serves to immensely boost further research groups that would integrate their own lidar system. It can be used as is, or as a basis for further developments. With trivial software changes, the developed workflows can be adapted to other sensor configurations, so for instance Velodyne's latest 128-channel lidar sensor with 300 m range (Velodyne LiDAR, Inc., 2021) could easily be integrated into a similar system.

The applicability of the system to diverse environments has also been demonstrated in the current thesis. An algorithm for segmenting trees in a bottom-up fashion has been developed in this work. The proposed method has been applied to a 0.5-ha point cloud acquired over a beech natural forest cell and trees up to 40 metres tall have been segmented. Even though the algorithm in its current form is prone to over- and under-segmentation errors, significant improvements are expected once the boresight issue is fixed and data from multiple flight lines can be combined for a better representation of trees. Also some ideas from existing algorithms that mainly operate in a top-down approach have been selected. Implementing these could significantly improve the algorithm's performance and versatility.

The system had also been applied in a completely different environment. Instead of 40 m tall trees, the second use-case looked at finding microforms in a bog. The features of interest were on a sub-meter spatial scale both horizontally and vertically. While the end-goal of reliably identifying microforms has not been achieved, the present work was the first one to use a UAV-mounted lidar to attempt this task. Moreover, the groundwork had been laid for detecting microforms: For a small 20×20 m ROI, the boresight errors of the system have been overcome to create a high-resolution DSM, which was further split into its low- and high-frequency components. The current results are promising and should be used to pursue the end-goal of identifying microforms.

Both use-cases presented in this thesis are of high relevance.

Segmentation of trees in natural forest cells and further measurements on the individual segments would amplify the effectiveness of the as-of-yet time- and labour-intensive data collection efforts of the state forestry agency. Such data would be directly responsible for gaining a better understanding of climate change's impact on regional flora. As the pace of climate change increases and the forests' ability to adapt is not a given, it is crucial to enable the collection and evaluation of high spatial and temporal resolution data.

Vechtaer Moor, the bog where the current algorithm has been implemented, is home to ongoing restoration efforts through rewetting (Raabe et al., 2018). The detection of microforms, such as hummocks and hollows, is an important aspect of bogs' monitoring. The plant communities that form these features are on one hand crucial for the recovery of the bogs to their natural state. And on the other hand they are extremely fragile, so detecting them early on can facilitate quick measures aimed to enhance these plants' circumstances. Mainly the water table can be promptly adjusted to optimize the well-being of the target species. Moreover, the low-frequency DTM that was extracted could help inform the decisions so as to maximize the measures' effectiveness. Restoration of degraded bogs is an essential aspect of climate change mitigation. That is because functioning bogs act as moderate net carbon sinks, but due to the huge accumulation of partially decayed organic matter, become important sources of greenhouse gases.

Summarizing, the present work combined know-how across a broad array of technical topics, varying from fundamental linear algebra to advanced mechatronics and software design to devise, build and test a UAV-borne lidar. The system has been used in two contrasting scenarios. The algorithms developed for the two use-cases have the potential to support important renaturation and conservation efforts directed at threatened ecosystems. Further work will surely be necessary to achieve actionable outcomes for either of the applications, but the intermediate results are encouraging.

REFERENCES

Adobe. (1992, June 3). *TIFF™ specification.* Adobe Systems Incorporated, Mountain View, US-CA. Retrieved from `https://www.adobe.io/content/dam/udp/en/open/standards/tiff/TIFF6.pdf` (Revision 6.0)

Agisoft. (2021, March 18). *Metashape Professional Edition.* Closed-source Software. Retrieved from `https://www.agisoft.com/` (v. 1.7.2)

Agisoft LLC. (2021). *Agisoft Metashape User Manual: Professional Edition, Version 1.7.* St. Petersburg, Russia. Retrieved from `https://www.agisoft.com/pdf/metashape-pro_1_7_en.pdf`

Amiri, N., Krzystek, P., Heurich, M., & Skidmore, A. (2019, November). Classification of Tree Species as Well as Standing Dead Trees Using Triple Wavelength ALS in a Temperate Forest. *Remote Sensing*, *11*(22), 2614. Retrieved from `https://doi.org/10.3390/rs11222614`

Amiri, N., Yao, W., Heurich, M., Krzystek, P., & Skidmore, A. K. (2016, October). Estimation of regeneration coverage in a temperate forest by 3D segmentation using airborne laser scanning data. *International Journal of Applied Earth Observation and Geoinformation*, *52*, 252–262. Retrieved from `https://doi.org/10.1016/j.jag.2016.06.022`

Analog Devices, Inc. (2019, September). *LTC4365 Overvoltage, Undervoltage and Reverse Supply Protection Controller Datasheet.* Wilmington, US-MA. Retrieved from `https://www.analog.com/media/en/technical-documentation/data-sheets/LTC4365.pdf` (Rev.B)

Andersen, H.-E., Reutebuch, S. E., McGaughey, R. J., d'Oliveira, M. V., & Keller, M. (2014, August). Monitoring selective logging in western Amazonia with repeat lidar flights. *Remote Sensing of Environment*, *151*, 157–165. Retrieved from `https://doi.org/10.1016/j.rse.2013.08.049`

Andersen, R., Farrell, C., Graf, M., Muller, F., Calvar, E., Frankard, P., ... Anderson, P. (2017, August). An overview of the progress and challenges of peatland restoration in Western Europe. *Restoration Ecology*, *25*(2), 271–282. Retrieved from `https://doi.org/10.1111/rec.12415`

Applanix Corporation. (n.d.). *Integration Guidelines: APX15UAV and Velodyne LiDAR.* Richmond Hill, CA-ON.

Applanix Corporation. (2016). *APX-15 User Guide.* Richmond Hill, CA-ON.

Applanix Corporation. (2019a, March). *APX-15 Hardware Integration Manual.* Richmond Hill, CA-ON. (PUBS-MAN-005351 Rev.5)

Applanix Corporation. (2019b, December). *APX-15 UAV Configuration Guide.* Richmond Hill, CA-ON. (PUBS-MAN-005355 Rev.5)

ASPRS. (2019, July 9). *LAS Specification.* The American Society for Photogrammetry & Remote Sensing, Bethesda, US-MD. Retrieved from `https://www.asprs.org/wp-content/uploads/2019/07/LAS_1_4_r15.pdf` (V. 1.4 - R15)

Becker, R., & Mosler, B. (2019, April). SPECTORS Structure and Goals. *Proceedings*, *30*(1), 55. Retrieved from `https://doi.org/10.3390/proceedings2019030055`

Besl, P., & McKay, N. D. (1992, February). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(2), 239–256. Retrieved from `https://doi.org/10.1109/34.121791`

Biondi, P., Valadon, G., Lalet, P., & Potter, G. (2020, September). *SCAPY- A powerful interactive packet manipulation program.* Retrieved from `https://scapy.net/` (v. 2.4.4)

Bizouard, C. (2021, January 7). *INFORMATION ON UTC - TAI.* Bulletin C 61. International Earth Rotation and Reference Systems Service, Paris, France. Retrieved from `https://datacenter.iers.org/data/latestVersion/16_BULLETIN_C16.txt`

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Brede, B., Lau, A., Bartholomeus, H., & Kooistra, L. (2017, October). Comparing RIEGL RiCOPTER UAV LiDAR Derived Canopy Height and DBH with Terrestrial LiDAR. *Sensors*, *17*(10), 2371. Retrieved from `https://doi.org/10.3390/s17102371`

Camarretta, N., Harrison, P. A., Bailey, T., Potts, B., Lucieer, A., Davidson, N., & Hunt, M. (2020, October). Monitoring forest structure to guide adaptive management of forest restoration: a review of remote sensing approaches. *New Forests*, *51*(4), 573–596. Retrieved from `https://doi.org/10.1007/s11056-019-09754-5`

Carless, D., Luscombe, D. J., Gatis, N., Anderson, K., & Brazier, R. E. (2019, June). Mapping landscape-scale peatland degradation using airborne lidar and multispectral data. *Landscape Ecology*, *34*(6), 1329–1345. Retrieved from `https://doi.org/10.1007/s10980-019-00844-5`

Chen, Q., Baldocchi, D., Gong, P., & Kelly, M. (2006, August). Isolating Individual Trees in a Savanna Woodland Using Small Footprint Lidar Data. *Photogrammetric Engineering & Remote Sensing*, *72*(8), 923–932. Retrieved from `https://doi.org/10.14358/pers.72.8.923`

CloudCompare. (2020, September 08). *3D point cloud and mesh processing software.* GPL Software. Retrieved from `https://www.cloudcompare.org/` (v. 2.11.3)

Combs, G., Ramirez, G., Bottom, T., Pane, C., Boehm, H., Hall, M., … The Wireshark team (2020, February 26). *Wireshark.* Retrieved from `https://www.wireshark.org` (V. 3.2.2)

Conrad Electronic SE. (2016). *4 Port Gigabit Metal Switch Mini Operating Instructions.* Hirschau, Germany. Retrieved from `https://asset.conrad.com/media10/add/160267/c1/-/gl/001423415ML02/manual-1423415-renkforce-rf-4270245-network-switch-4-ports-1-gbps-usb-power-supply.pdf` (1423415)

Couwenberg, J., Thiele, A., Tanneberger, F., Augustin, J., Bärisch, S., Dubovik, D., … Joosten, H. (2011, May). Assessing greenhouse gas emissions from peatlands using vegetation as a proxy. *Hydrobiologia*, *674*(1), 67–89. Retrieved from `https://doi.org/10.1007/s10750-011-0729-x`

Dalponte, M., Bruzzone, L., & Gianelle, D. (2012, August). Tree species classification in the Southern Alps based on the fusion of very high geometrical resolution multispectral/hyperspectral images and LiDAR data. *Remote Sensing of Environment*, *123*, 258–270. Retrieved from `https://doi.org/10.1016/j.rse.2012.03.013`

Dalponte, M., & Coomes, D. A. (2016, May). Tree-centric mapping of forest carbon density from airborne laser scanning and hyperspectral data. *Methods in Ecology and Evolution*, *7*(10), 1236–1245. Retrieved from `https://doi.org/10.1111/2041-210x.12575`

Dargie, T. (2003). Monitoring and assessing change on the bog restoration surface: a review. In *Introduction to the workshop* (Vol. 26, p. 48).

DJI. (n.d.). *Matrice 600 Pro.* Shenzhen, China. Retrieved 19 March 2021, from `https://www.dji.com/uk/matrice600-pro`

DJI. (2017, October). *Matrice 600 Pro Disclaier and Safety Guidelines.* Shenzhen, China. Retrieved from `https://dl.djicdn.com/downloads/m600%20pro/20171017/Matrice_600_Pro_Disclaimer_and_Safety_Guidelines_v1.6_EN.pdf` (V 1.6)

DJI. (2018a, November). *DJI GS PRO User Manual.* Shenzhen, China. Retrieved from `https://dl.djicdn.com/downloads/groundstation_pro/20181102/GS_Pro_User_Manual_v2.0_EN_201811.pdf` (V 2.0)

DJI. (2018b, April). *Matrice 600 Pro User Manual.* Shenzhen, China. Retrieved from `https://dl.djicdn.com/downloads/m600%20pro/1208EN/Matrice_600_Pro_User_Manual_v1.0_EN_1208.pdf` (V 1.0)

Dogotari, M., Prüm, M., Lam, O. H. Y., Vithlani, H. N., Vu, V. H., Melville, B., & Becker, R. (2019, May). Development of a UAV-Borne LiDAR System for Surveying Applications. *Proceedings*, *30*(1), 75. Retrieved from `https://doi.org/10.3390/proceedings2019030075`

Dogotari, M., & Rostalski, S. (2021, April). *HSRW PG02 Laser-Scanner: Initial release.* GitHub repository. Retrieved from `https://doi.org/10.5281/zenodo.4671498` (v0.1-alpha-1)

Dronova, I., Kislik, C., Dinh, Z., & Kelly, M. (2021, May). A Review of Unoccupied Aerial Vehicle Use in Wetland Applications: Emerging Opportunities in Approach, Technology, and Data. *Drones*, *5*(2), 45. Retrieved from `https://doi.org/10.3390/drones5020045`

EMVA. (2019, January 18). *GenICam™ Pixel Format Naming Convention (PFNC).* European Machine Vision Association, Barcelona, Spain. Retrieved from `https://www.emva.org/wp-content/uploads/GenICam_PFNC_2_3.pdf` (V. 2.3)

Fursa, V. (2021, April 15). *Reach Firmware 26 Update: Legacy RTCM3 Messages and NMEA Output Settings.* Forum post. Retrieved from `https://community.emlid.com/t/reach-firmware-26-update-legacy-rtcm3-messages-and-nmea-output-settings/24653`

Gao, F., & Han, L. (2010, May). Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, *51*(1), 259–277. Retrieved from `https://doi.org/10.1007/s10589-010-9329-3`

GDAL/OGR contributors. (2021). GDAL/OGR geospatial data abstraction software library [Computer software manual]. Retrieved from `https://gdal.org`

Günther, A., Barthelmes, A., Huth, V., Joosten, H., Jurasinski, G., Koebsch, F., & Couwenberg, J. (2020, April). Prompt rewetting of drained peatlands reduces climate warming despite methane emissions. *Nature Communications*, *11*(1). Retrieved from `https://doi.org/10.1038/s41467-020-15499-z`

Guo, Q., Su, Y., Hu, T., Zhao, X., Wu, F., Li, Y., ... Wang, X. (2017, January). An integrated UAV-borne lidar system for 3D habitat mapping in three forest ecosystems across China. *International Journal of Remote Sensing*, *38*(8-10), 2954–2972. Retrieved from `https://doi.org/10.1080/01431161.2017.1285083`

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020, September). Array programming with NumPy. *Nature*, *585*(7825), 357–362. Retrieved from `https://doi.org/10.1038/s41586-020-2649-2`

Harvey, P. (2019, April 24). *ExifTool.* Retrieved from `https://exiftool.org/` (V. 11.38)

Heurich, M. (2008, April). Automatic recognition and measurement of single trees based on data from airborne laser scanning over the richly structured natural forests of the Bavarian Forest National Park. *Forest Ecology and Management*, *255*(7), 2416–2433. Retrieved from `https://doi.org/10.1016/j.foreco.2008.01.022`

Hillman, S., Hally, B., Wallace, L., Turner, D., Lucieer, A., Reinke, K., & Jones, S. (2021, March). High-Resolution Estimates of Fire Severity—An Evaluation of UAS Image and LiDAR Mapping Approaches on a Sedgeland Forest Boundary in Tasmania, Australia. *Fire*, *4*(1), 14. Retrieved from `https://doi.org/10.3390/fire4010014`

Huawei Technologies Co., Ltd. (2015, January). *HUAWEI E3372 LTE USB Stick Product Description.* Shenzhen, China. Retrieved from `https://usermanual.wiki/Huawei/HUAWEIE3372LTEUSBStickProductDescriptionV100R00102Englishpdf.2079741206.pdf` (Issue 02)

Hyyppä, J., Kelle, O., Lehikoinen, M., & Inkinen, M. (2001, May). A segmentation-based method to retrieve stem volume estimates from 3-D tree height models produced by laser scanners. *IEEE Transactions on Geoscience and Remote Sensing*, *39*(5), 969–975. Retrieved from `https://doi.org/10.1109/36.921414`

IDS GmbH. (2017, March). *I/O standard cable, straight, 3 m.* Specification sheet. Obersulm, Germany. Retrieved from `https://en.ids-imaging.com/tl_files/downloads/STORE/acc_spec/233/datasheet/AD00044.03_EN.pdf` (AD00044.03)

IDS GmbH. (2020a). *IDS Peak IPL.* Software Development Kit. Obersulm, Germany. (v. 1.1.15)

IDS GmbH. (2020b). *IDS Peak SDK.* Software Development Kit. Obersulm, Germany. (v. 1.1.6)

IDS GmbH. (2020c, December). *IDS Vision Reference 2.4.* Specification sheet. Obersulm, Germany. Retrieved from `https://en.ids-imaging.com/files/downloads/ids-vision-reference/ids-vision-reference-2.4_EN.pdf`

IDS GmbH. (2020d). *U3-3200SE-C-HQ Datasheet.* Obersulm, Germany. (AB02354)

IDS GmbH. (2020e, December). *uEye SE USB 3.1 Gen 1.* Technical Manual. Obersulm, Germany. Retrieved from `https://en.ids-imaging.com/files/downloads/usb3-vision/manuals/technical-manual-u3v-usb-3.1-gen-1-se_EN.pdf`

IDS GmbH. (2021a, March 01). *Demosaicing.* Email Technical Support. Obersulm, Germany.

IDS GmbH. (2021b, March 08). *IDS Peak IPL.* Software Development Kit. Obersulm, Germany. Retrieved from `https://en.ids-imaging.com/manuals/ids-peak/ids-peak-ipl-documentation/1.2.1/en/index.html` (v. 1.2.1)

IDS GmbH. (2021c). *List of supported features – firmware version 2.7.* Specification sheet. Obersulm, Germany. Retrieved from `https://en.ids-imaging.com/files/downloads/STORE/genapi/ids-list-of-supported-features-usb3-vision.pdf`

IMST GmbH. (2018a, March 26). *WiMOD iM282A Datasheet.* Kamp-Lintfort, Germany. Retrieved from `https://wireless-solutions.de/wp-content/uploads/2019/02/iM282A_Datasheet_V1_0.pdf` (4000/40140/0124. V. 1.0)

IMST GmbH. (2018b, April 11). *WiMOD - iM282A Range Test.* Application Note. Kamp-Lintfort, Germany. Retrieved from `https://wireless-solutions.de/wp-content/uploads/2019/02/iM282A_AN023_RangeTest_V1_0-1.pdf` (AN023. V. 1.0)

IMST GmbH. (2019, May 24). *WiMODino Datasheet.* Kamp-Lintfort, Germany. Retrieved from `https://wireless-solutions.de/wp-content/uploads/2019/12/WiMODino_Datasheet_V1-0.pdf` (4100/40140/0136. V. 1.0)

Intel Corporation. (2014, March). *Intel® NUC Board D53427RKE Technical Product Specification.* Retrieved from `https://www.intel.com/content/dam/support/us/en/documents/boardsandkits/D53427RKE_TechProdSpec05.pdf` (G97389-005)

Isenburg, M. (2013). LASzip: lossless compression of LiDAR data. *Photogrammetric engineering and remote sensing*, *79*(2), 209–217. Retrieved from `https://www.cs.unc.edu/~isenburg/lastools/download/laszip.pdf`

Isenburg, M. (2021, January 28). *LAStools - efficient LiDAR processing software.* Retrieved from `https://rapidlasso.com/LAStools` (V. 210128 academic)

Jaakkola, A., Hyyppä, J., Kukko, A., Yu, X., Kaartinen, H., Lehtomäki, M., & Lin, Y. (2010, November). A low-cost multi-sensoral mobile mapping system and its feasibility for tree measurements. *ISPRS Journal of Photogrammetry and Remote Sensing*, *65*(6), 514–522. Retrieved from `https://doi.org/10.1016/j.isprsjprs.2010.08.002`

Jaeger, N. (2017). *APX UAV Heading Initialization and Alignment.* Technical Note. Applanix Corporation, Richmond Hill, CA-ON. (Rev. 0)

Joosten, H. (2012, February). Zustand und Perspektiven der Moore weltweit. *Natur und Landschaft*, *87*(2), 5055. Retrieved from `https://doi.org/10.17433/2.2012.50153141.50-55`

Joosten, H., & Couwenberg, J. (2008). Peatlands and Carbon. In F. Parish et al. (Eds.), *Assessment on Peatlands, Biodiversity and Climate Change (Main Report)* (pp. 99–117). Global Environment Centre, Kuala Lumpur & Wetlands International, Wageningen.

Korpela, I., Haapanen, R., Korrensalo, A., Tuittila, E.-S., & Vesala, T. (2020, February). Fine-resolution mapping of microforms of a boreal bog using aerial images and waveform-recording LiDAR. *Mires and Peat*, *26*(03), 124. Retrieved from `https://doi.org/10.19189/MaP.2018.OMB.388`

Korpela, I., Koskinen, M., Vasander, H., Holopainen, M., & Minkkinen, K. (2009, September). Airborne small-footprint discrete-return LiDAR data in the assessment of boreal mire surface patterns, vegetation, and habitats. *Forest Ecology and Management*, *258*(7), 1549–1566. Retrieved from `https://doi.org/10.1016/j.foreco.2009.07.007`

LaRocque, A., Phiri, C., Leblon, B., Pirotti, F., Connor, K., & Hanson, A. (2020, June). Wetland Mapping with Landsat 8 OLI, Sentinel-1, ALOS-1 PALSAR, and LiDAR Data in Southern New Brunswick, Canada. *Remote Sensing*, *12*(13), 2095. Retrieved from `https://doi.org/10.3390/rs12132095`

Leffler, S., Rouault, E., Friesenhahn, B., Warmerdam, F., Kiselev, A., Damme, J. V., ... Silicon Graphics (2021, April 16). *LibTIFF - TIFF Library and Utilities.* Retrieved from `https://libtiff.gitlab.io/libtiff/` (V. 4.3.0)

Lehmann, J., Münchberger, W., Knoth, C., Blodau, C., Nieberding, F., Prinz, T., ... Kleinebecker, T. (2016, February). High-Resolution Classification of South Patagonian Peat Bog Microforms Reveals Potential Gaps in Up-Scaled CH4 Fluxes by use of Unmanned Aerial System (UAS) and CIR Imagery. *Remote Sensing*, *8*(3), 173. Retrieved from `https://doi.org/10.3390/rs8030173`

Li, W., Guo, Q., Jakubowski, M. K., & Kelly, M. (2012, January). A New Method for Segmenting Individual Trees from the Lidar Point Cloud. *Photogrammetric Engineering & Remote Sensing*, *78*(1), 75–84. Retrieved from `https://doi.org/10.14358/pers.78.1.75`

Linear Technology Corporation. (2012). *LTC4417 Prioritized PowerPath™ Controller Datasheet.* Milpitas, US-CA. Retrieved from `https://www.analog.com/media/en/technical-documentation/data-sheets/4417f.pdf`

Lovitt, J., Rahman, M. M., & McDermid, G. J. (2017, July). Assessing the Value of UAV Photogrammetry for Characterizing Terrain in Complex Peatlands. *Remote Sensing*, *9*(7), 715. Retrieved from `https://doi.org/10.3390/rs9070715`

Lucieer, A. (2018, December). *Recommendations for buidling a UAV-borne lidar system.* Personal communication. Hobart, AU-TAS.

Luscombe, D. J., Anderson, K., Gatis, N., Wetherelt, A., Grand-Clement, E., & Brazier, R. E. (2015, August). What does airborne LiDAR really measure in upland ecosystems? *Ecohydrology*, *8*(4), 584–594. Retrieved from `https://doi.org/10.1002/eco.1527`

Marak, C., & Havens, P. (2015, May). *Ethernet Protection Design Guide* (Tech. Rep.). Chicago, US-IL. Retrieved 09 April 2021, from `https://www.littelfuse.com/~/media/electronics/design_guides/esd/littelfuse_ethernet_protection_design_guide.pdf.pdf`

Microsemi. (2018, June). *Magnetics Guide.* Application Note. Aliso Viejo, US-CA. Retrieved from `https://ww1.microchip.com/downloads/en/AppNotes/VPPD-01740.pdf` (ENT-AN0098 Rev. 2.2)

Næsset, E., Gobakken, T., Holmgren, J., Hyyppä, H., Hyyppä, J., Maltamo, M., ... Söderman, U. (2004, December). Laser scanning of forest resources: the nordic experience. *Scandinavian Journal of Forest Research*, *19*(6), 482–499. Retrieved from `https://doi.org/10.1080/02827580410019553`

Nelder, J. A., & Mead, R. (1965, January). A Simplex Method for Function Minimization. *The Computer Journal*, *7*(4), 308–313. Retrieved from `https://doi.org/10.1093/comjnl/7.4.308`

Nelson, R., Krabill, W., & Tonelli, J. (1988, March). Estimating forest biomass and volume using airborne laser data. *Remote Sensing of Environment*, *24*(2), 247–267. Retrieved from `https://doi.org/10.1016/0034-4257(88)90028-4`

Nick, K.-J. (2001). Ergebnisse und Erkenntnisse aus der Wiedervernässung des Leegmoores. In *Moorregeneration im Leegmoor/Emsland nach Schwarztorfabbau und Wiedervernässung. – Angewandte Landschaftsökologie* (pp. 163–204). Bundesamt für Naturschutz, Bonn-Bad Godesberg.

NMEA. (2021). *NMEA 0183 Standard.* National Marine Electronics Association, Severna Park, US-MD. Retrieved 21 April 2021, from `https://www.nmea.org/content/STANDARDS/NMEA_0183_Standard`

Nonami, K., Kendoul, F., Suzuki, S., Wang, W., & Nakazawa, D. (2010). Autonomous Control of a Mini Quadrotor Vehicle Using LQG Controllers. In *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles* (pp. 61–76). Tokyo: Springer Japan. Retrieved from `https://doi.org/10.1007/978-4-431-53856-1_3`

Popescu, S. C., & Wynne, R. H. (2004). Seeing the Trees in the Forest: Using Lidar and Multispectral Data Fusion with Local Filtering and Variable Window Size for Estimating Tree Height. *Photogrammetric Engineering & Remote Sensing*, *70*(5), 589–604.

Powell, M. J. D. (1964, February). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, *7*(2), 155–162. Retrieved from `https://doi.org/10.1093/comjnl/7.2.155`

Prüm, M. (2017, August). *Automated Hyperspectral Field Scanner for Vegetation Monitoring.* Bachelor Thesis. Kamp-Lintfort, Germany: Hochschule Rhein-Waal Fakultät Kommunikation und Umwelt. Retrieved from `https://nbn-resolving.org/urn:nbn:de:hbz:1383-opus4-2247`

Prüm, M. (2019, January 23). *Power consumption and operating voltage of the NUC D53427RKE board.* Personal communication.

Pulse Electronics. (2020). *Layout Considerations for Pulse Ethernet Magnetics and Ethernet Connector Modules.* San Diego, US-CA. Retrieved from `https://www.pulseelectronics.com/wp-content/uploads/2020/12/Pulse_Layout-Considerations-v7.pdf`

Python Software Foundation. (2019, October 14). *Whats New In Python 3.8.* Wilmington, US-DE. Retrieved from `https://docs.python.org/3.8/whatsnew/3.8.html`

QGIS Development Team. (2021). QGIS Geographic Information System [Computer software manual]. Retrieved from `https://www.qgis.org`

R Core Team. (2021). R: A Language and Environment for Statistical Computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

Raabe, P., Kleinebecker, T., Knorr, K.-H., Hölzel, N., & Gramann, G. (2018). Propagation and establishment of hummock peat mosses in bog restoration – first results of a pilot study in Vechta, Lower Saxony. *TELMA-Berichte der Deutschen Gesellschaft für Moor-und Torfkunde*, *48*, 71–80.

Rapinel, S., Hubert-Moy, L., & Clément, B. (2011). Using LiDAR data to evaluate wetland functions. In *34th International Symposium for Remote Sensing of the Environment (ISRSE).*

Räsänen, A., Juutinen, S., Kalacska, M., Aurela, M., Heikkinen, P., Mäenpää, K., ... Virtanen, T. (2020, October). Peatland leaf-area index and biomass estimation with ultra-high resolution remote sensing. *GIScience & Remote Sensing*, *57*(7), 943–964. Retrieved from `https://doi.org/10.1080/15481603.2020.1829377`

Raymond, E. S. (2021, January). *NMEA Revealed.* Retrieved 21 April 2021, from `https://gpsd.gitlab.io/gpsd/NMEA.html` (V. 2.24)

Reback, J., McKinney, W., jbrockmendel, den Bossche, J. V., Augspurger, T., Cloud, P., … The pandas development team (2021, March). *pandas-dev/pandas: Pandas.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.3509134` (v. 1.2.3)

Reitberger, J., Schnörr, C., Krzystek, P., & Stilla, U. (2009, November). 3D segmentation of single trees exploiting full waveform LIDAR data. *ISPRS Journal of Photogrammetry and Remote Sensing*, *64*(6), 561–574. Retrieved from `https://doi.org/10.1016/j.isprsjprs.2009.04.002`

Roussel, J.-R., Auty, D., Coops, N. C., Tompalski, P., Goodbody, T. R., Meador, A. S., … Achim, A. (2020). lidR: An R package for analysis of Airborne Laser Scanning (ALS) data. *Remote Sensing of Environment*, *251*, 112061. Retrieved from `https://doi.org/10.1016/j.rse.2020.112061`

Seller, O., & Sornin, N. (2013, February 5). *Low power long range transmitter.* European Patent Application. Semtech Corporation. Retrieved from `https://patentimages.storage.googleapis.com/01/d2/4f/3abd85ffa8a318/EP2763321A1.pdf` (EP 2 763 321 A1)

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(8), 888–905.

Silva, C. A., Hudak, A. T., Vierling, L. A., Loudermilk, E. L., O'Brien, J. J., Hiers, J. K., … Khosravipour, A. (2016, July). Imputation of Individual Longleaf Pine (*Pinus palustris* Mill.) Tree Attributes from Field and LiDAR Data. *Canadian Journal of Remote Sensing*, *42*(5), 554–573. Retrieved from `https://doi.org/10.1080/07038992.2016.1196582`

SNIP Support. (2020, August 7). *Adding MT1008 RTCM Messages.* Technical support post. Retrieved from `https://www.use-snip.com/kb/knowledge-base/adding-mt1008-rtcm-messages/`

Summers, J. (2014, September 7). *TweakPNG - A PNG image file manipulation utility.* Software. Retrieved from `https://entropymine.com/jason/tweakpng/` (v. 1.4.6)

Suomalainen, J., Anders, N., Iqbal, S., Roerink, G., Franke, J., Wenting, P., … Kooistra, L. (2014, November). A Lightweight Hyperspectral Mapping System and Photogrammetric Processing Chain for Unmanned Aerial Vehicles. *Remote Sensing*, *6*(11), 11013–11030. Retrieved from `https://doi.org/10.3390/rs61111013`

Tallysman Inc. (2019). *HC975 Triple Band Helical Antenna + L-band Datasheet.* Ottawa, CA-ON. Retrieved from `https://www.tallysman.com/app/uploads/2018/03/Tallysman%C2%AE-HC975-Datasheet.pdf` (Rev. 1.0)

The Tcpdump Group. (2020). *tcpdump – dump traffic on a network.* Computer software. (v. 4.9.3)

torriem. (2019, April 3). *How to use ZED-F9P as a base station for Trimble.* Forum post. Toronto, CA-ON. Retrieved from `https://www.thecombineforum.com/threads/how-to-use-zed-f9p-as-a-base-station-for-trimble.331721/`

Töyrä, J., & Pietroniro, A. (2005, July). Towards operational monitoring of a northern wetland using geomatics-based techniques. *Remote Sensing of Environment*, *97*(2), 174–191. Retrieved from `https://doi.org/10.1016/j.rse.2005.03.012`

Trimble. (n.d.). *NMEA-0183 messages: Overview.* Sunnyvale, US-CA. Retrieved 21 April 2021, from `https://www.trimble.com/OEM_ReceiverHelp/V4.44/en/NMEA-0183messages_MessageOverview.html`

Trimble Applanix. (2016, March). *APX-15-V2 UAV Datasheet.* Richmond Hill, CA-ON. Retrieved from `https://www.applanix.com/downloads/products/specs/APX15_DS_NEW_0408_YW.pdf`

Vaughn, N. R., Moskal, L. M., & Turnblom, E. C. (2012, February). Tree Species Detection Accuracies Using Discrete Point Lidar and Airborne Waveform Lidar. *Remote Sensing*, *4*(2), 377–403. Retrieved from `https://doi.org/10.3390/rs4020377`

Velodyne Acoustics, Inc. (2015). *Velodyne LiDAR PUCK™ Datasheet.* San Jose, US-CA. (63-9229 Rev-B)

Velodyne LiDAR, Inc. (2018a). *Puck LITE™ Datasheet.* San Jose, US-CA. (63-9286 Rev-H)

Velodyne LiDAR, Inc. (2018b, April). *Returning Sensors with Cut Cables PI.* Confidential Document. San Jose, US-CA. (70-0544 Rev-A)

Velodyne LiDAR, Inc. (2018c, April). *Velodyne VLP product cable cut process instructions.* Confidential Document. San Jose, US-CA. (70-0543 Rev-A)

Velodyne LiDAR, Inc. (2019a, January 31). *Power consumption and operating voltage of PUCK LITE.* Technical support. San Jose, US-CA.

Velodyne LiDAR, Inc. (2019b). *VLP-16 User Manual.* San Jose, US-CA. (63-9243 Rev-E)

Velodyne LiDAR, Inc. (2021). *Alpha Prime™ Datasheet.* San Jose, US-CA. (63-9679 Rev-A)

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., . . . SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. Retrieved from `https://doi.org/10.1038/s41592-019-0686-2`

Vu, V. H., & Dogotari, M. (2020, April). *Wimod shield for APX and Emlid.* GitLab repository. Retrieved from `https://gitlab.spectors.eu/hung/WimoDinoLora`

W3C. (2003, November 10). *Portable Network Graphics (PNG) Specification (Second Edition).* W3C Recommendation. Retrieved from `https://www.w3.org/TR/2003/REC-PNG-20031110`

Wald und Holz NRW. (n.d.-a). *Das Naturwaldzellen-Programm Nordrhein Westfalen.* Münster, Germany. Retrieved 21 March 2021, from `https://www.wald-und-holz.nrw.de/wald-in-nrw/naturwaldzellen/naturwaldzellen-programm-in-nrw`

Wald und Holz NRW. (n.d.-b). *Niederkamp. Naturwaldzelle 43.* Münster, Germany. Retrieved 22 March 2021, from `https://www.wald-und-holz.nrw.de/wald-in-nrw/naturwaldzellen/niederkamp`

Wallace, L., Lucieer, A., Watson, C., & Turner, D. (2012, May). Development of a UAV-LiDAR System with Application to Forest Inventory. *Remote Sensing*, *4*(6), 1519–1543. Retrieved from `https://doi.org/10.3390/rs4061519`

Wallace, L., Lucieer, A., & Watson, C. S. (2014, December). Evaluating Tree Detection and Segmentation Routines on Very High Resolution UAV LiDAR Data. *IEEE Transactions on Geoscience and Remote Sensing*, *52*(12), 7619–7628. Retrieved from `https://doi.org/10.1109/tgrs.2014.2315649`

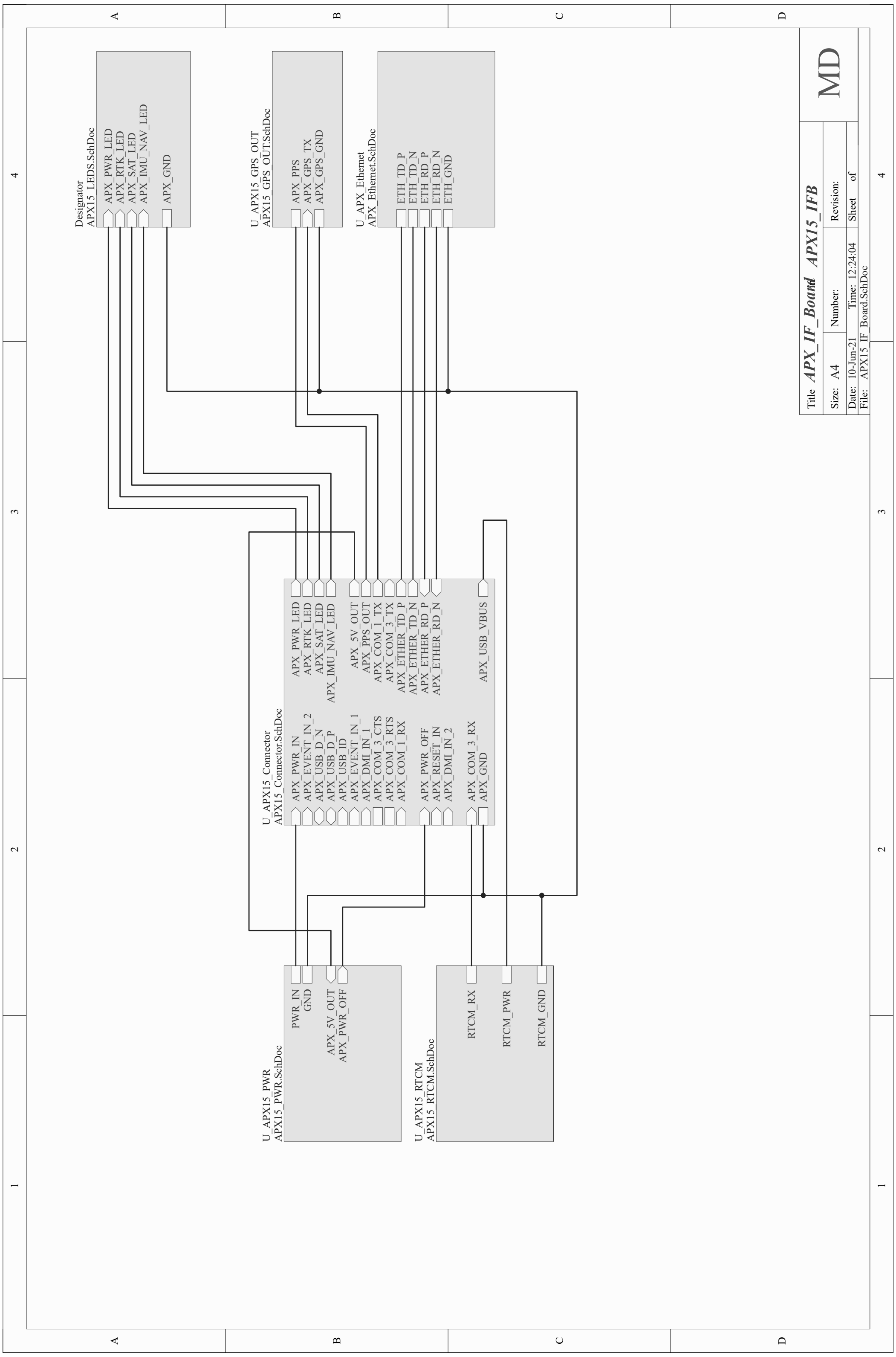
Windrim, L., & Bryson, M. (2020, May). Detection, Segmentation, and Model Fitting of Individual Tree Stems from Airborne Laser Scanning of Forests Using Deep Learning. *Remote Sensing*, *12*(9), 1469. Retrieved from `https://doi.org/10.3390/rs12091469`

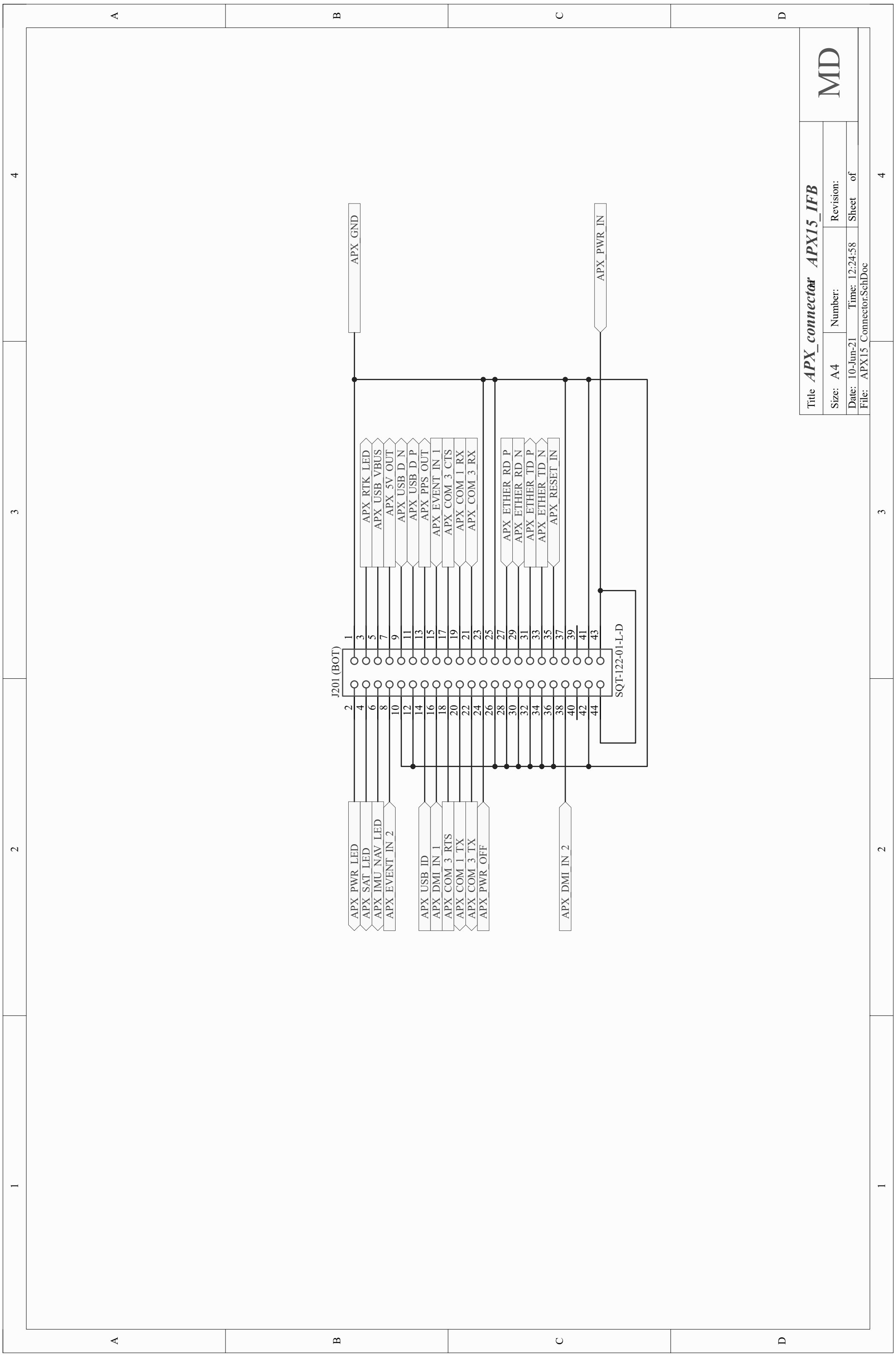
Yao, W., Krzystek, P., & Heurich, M. (2012, August). Tree species classification and estimation of stem volume and DBH based on single tree extraction by exploiting airborne full-waveform LiDAR data. *Remote Sensing of Environment*, *123*, 368–380. Retrieved from `https://doi.org/10.1016/j.rse.2012.03.027`

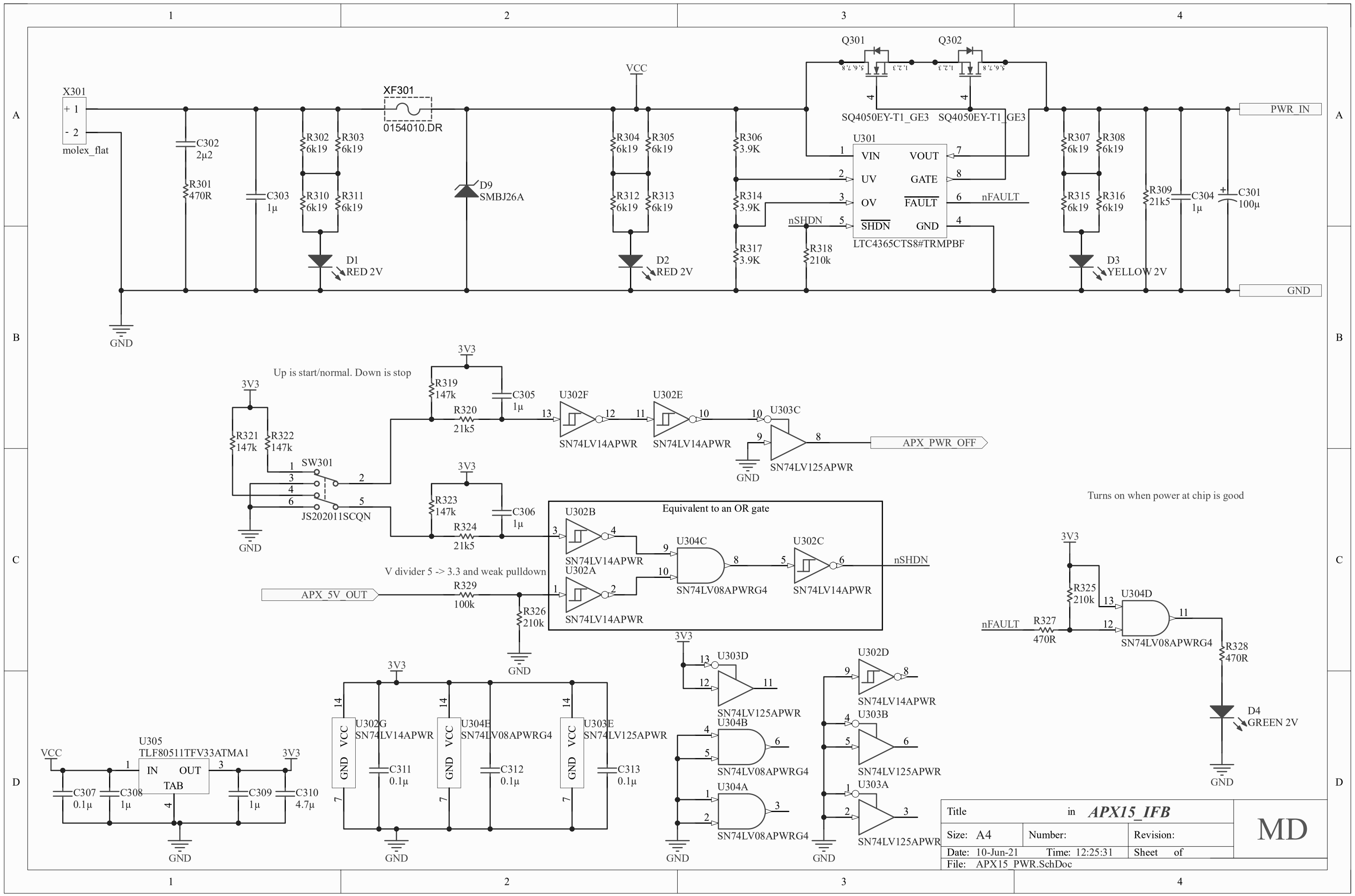
Yao, W., Krzystek, P., & Heurich, M. (2013, October). Enhanced detection of 3D individual trees in forested areas using airborne full-waveform LiDAR data by combining normalized cuts with spatial density clustering. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *II-5/W2*, 349–354. Retrieved from `https://doi.org/10.5194/isprsannals-ii-5-w2-349-2013`

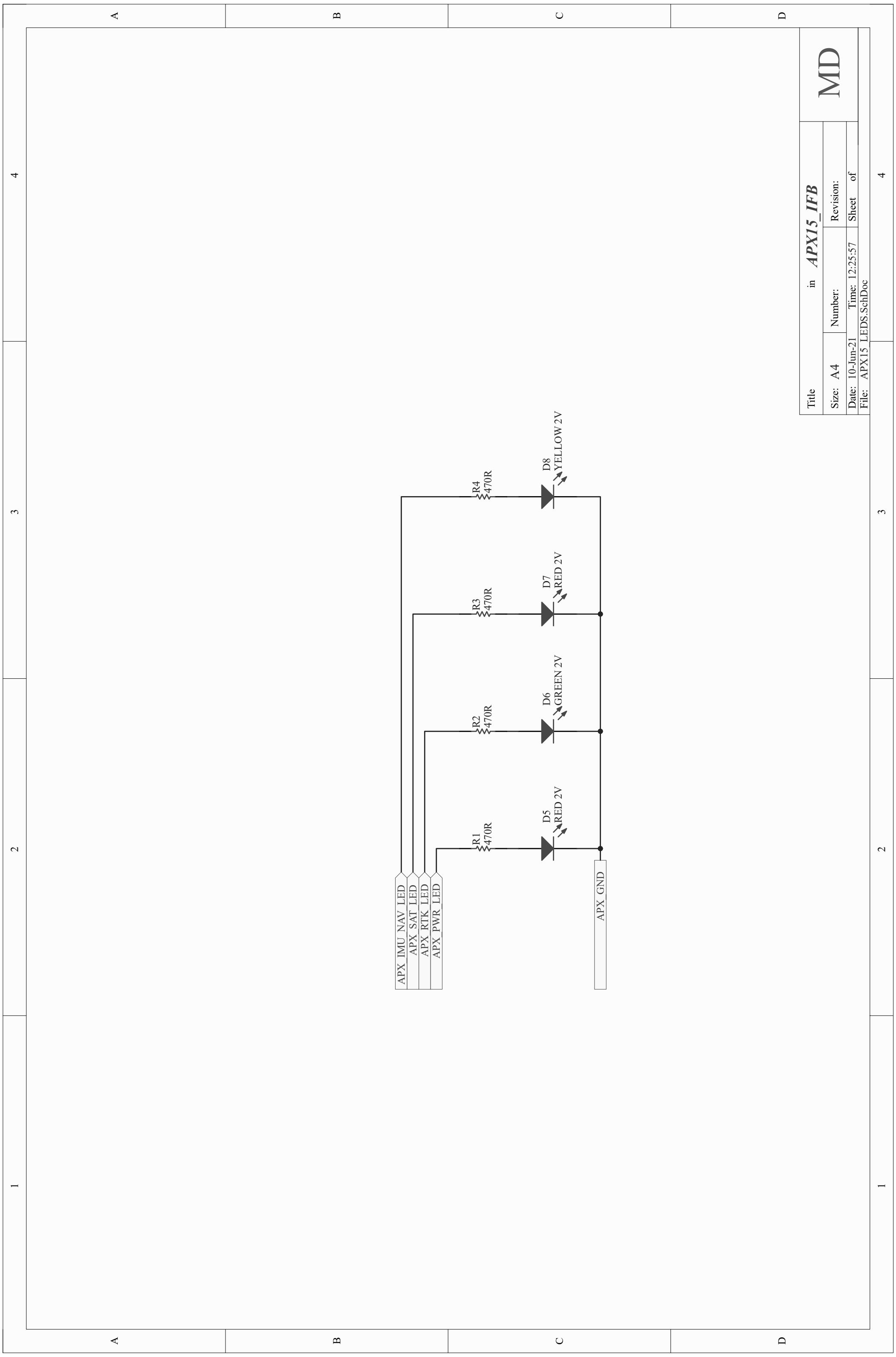
Appendices

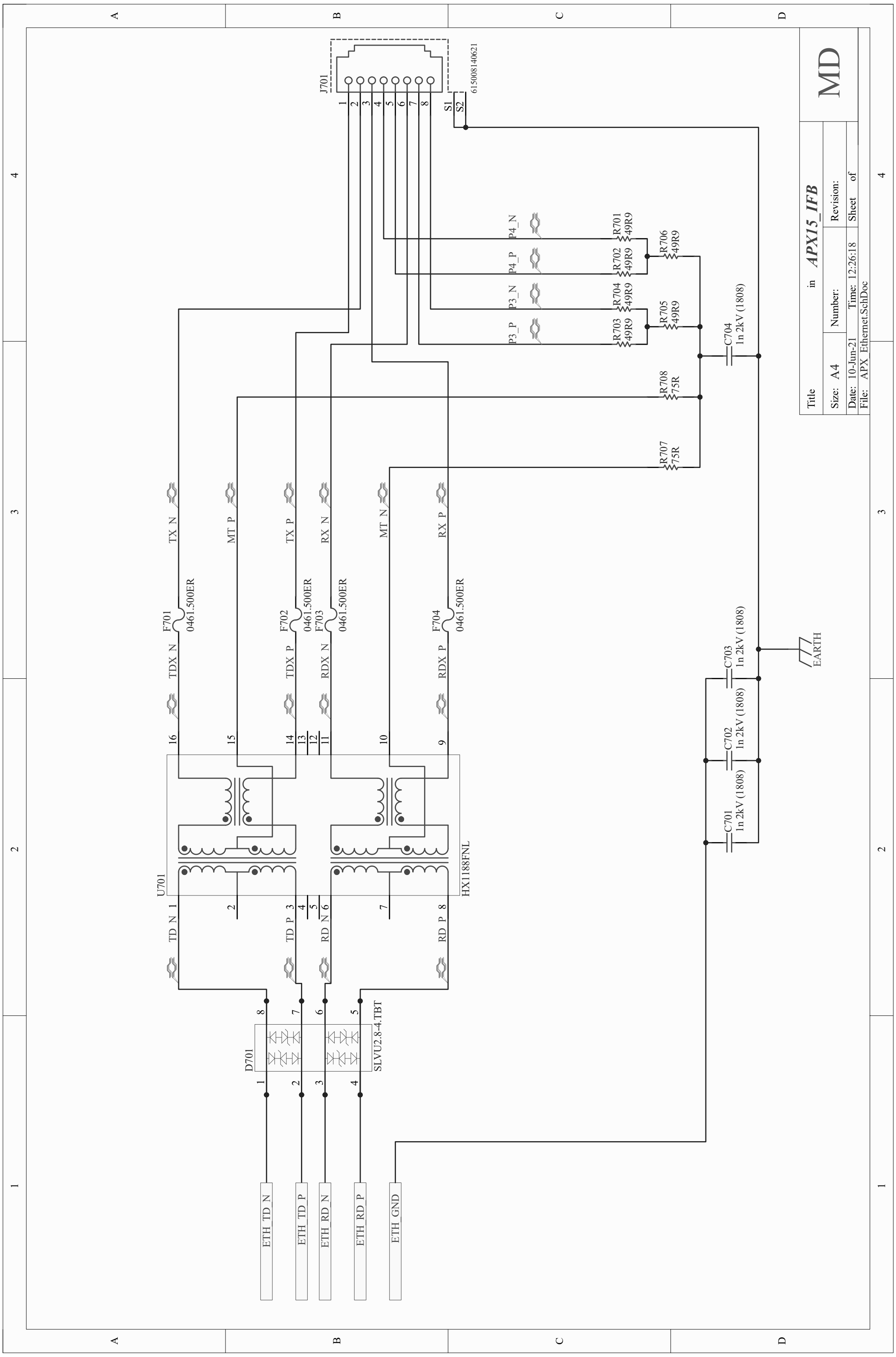
A. SCHEMATIC DIAGRAMS

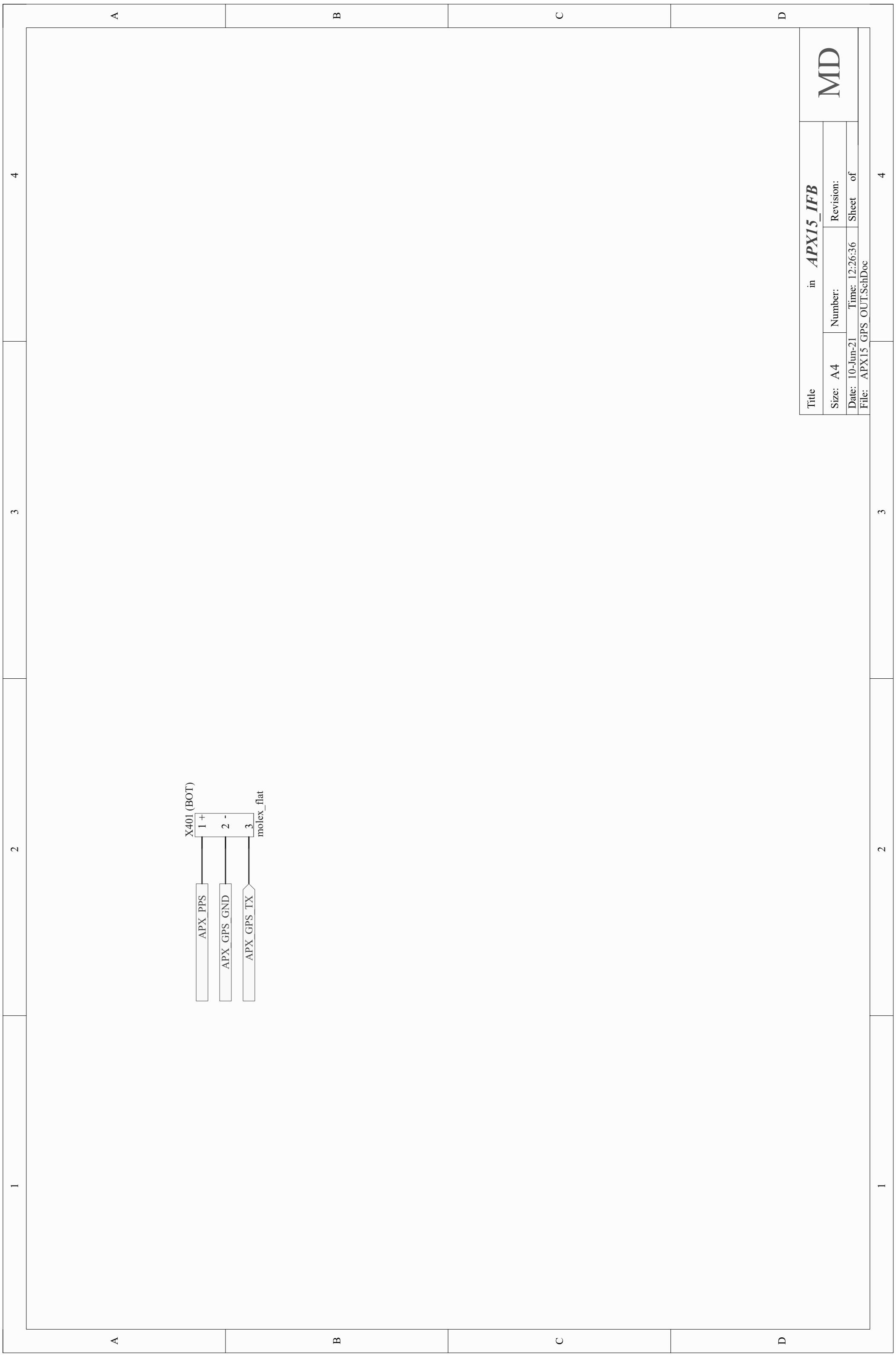


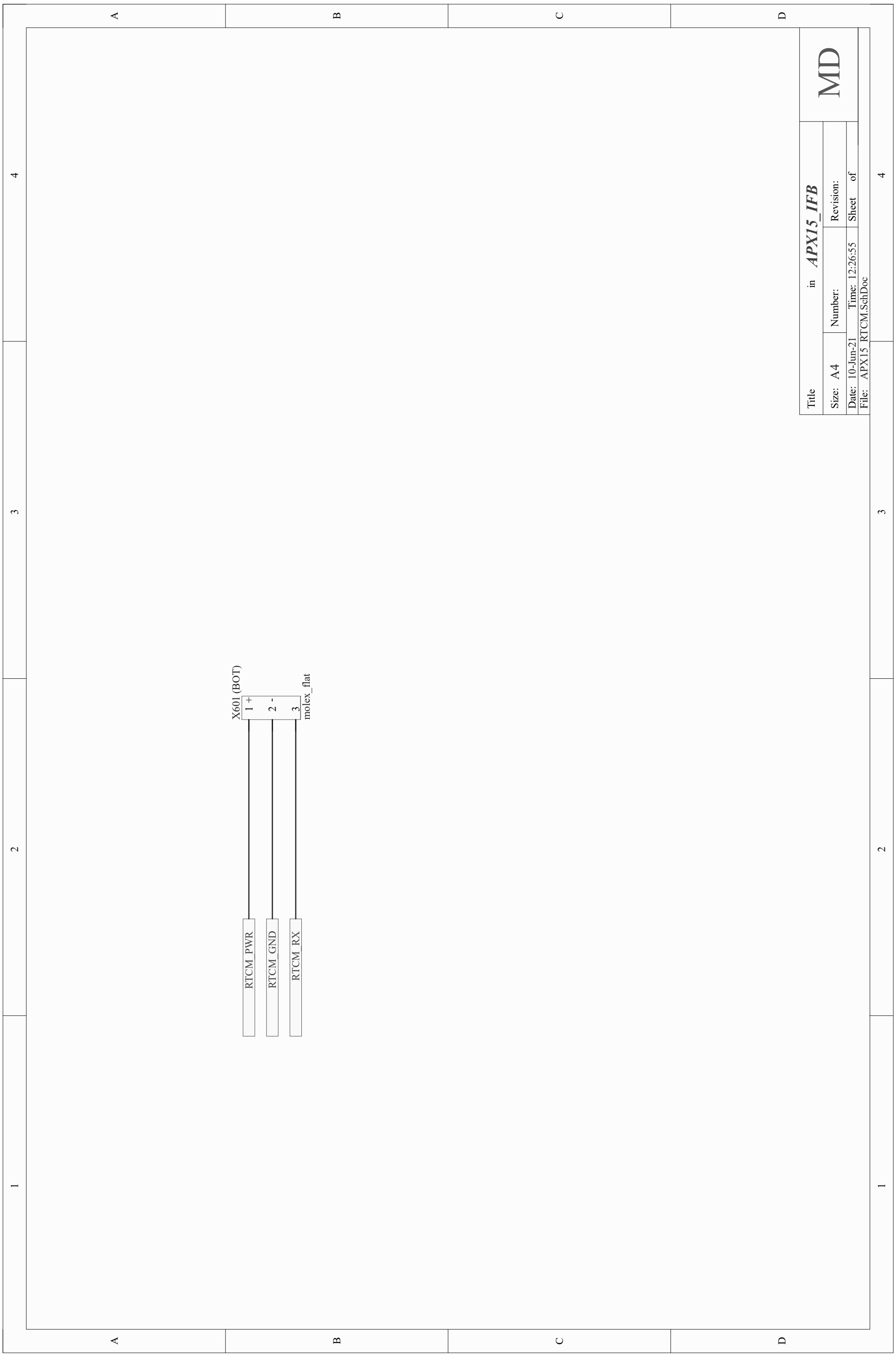

APX GND
APX_PWR_IN
APX RTK LED
APX USB VBUS
APX 5V OUT
APX USB D N
APX USB D P
APX PPS OUT
APX EVENT IN 1
APX COM 3 CTS
APX COM 1 RX
APX COM 3 RX
APX ETHER RD P
APX ETHER RD N
APX ETHER TD P
APX ETHER TD N
APX RESET IN
J201 (BOT)
SQT-122-01-L-D
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44
APX PWR LED
APX SAT LED
APX IMU NAV LED
APX EVENT IN 2
APX USB ID
APX DMI IN 1
APX COM 3 RTS
APX COM 1 TX
APX COM 3 TX
APX PWR OFF
APX DMI IN 2
Title APX_connector APX15_IFB
Size: A4
Number:
Revision:
Date: 10-Jun-21
Time: 12:24:58
Sheet of
File: APX15 Connector.SchDoc
MD
A
B
C
D
1
2
3
4

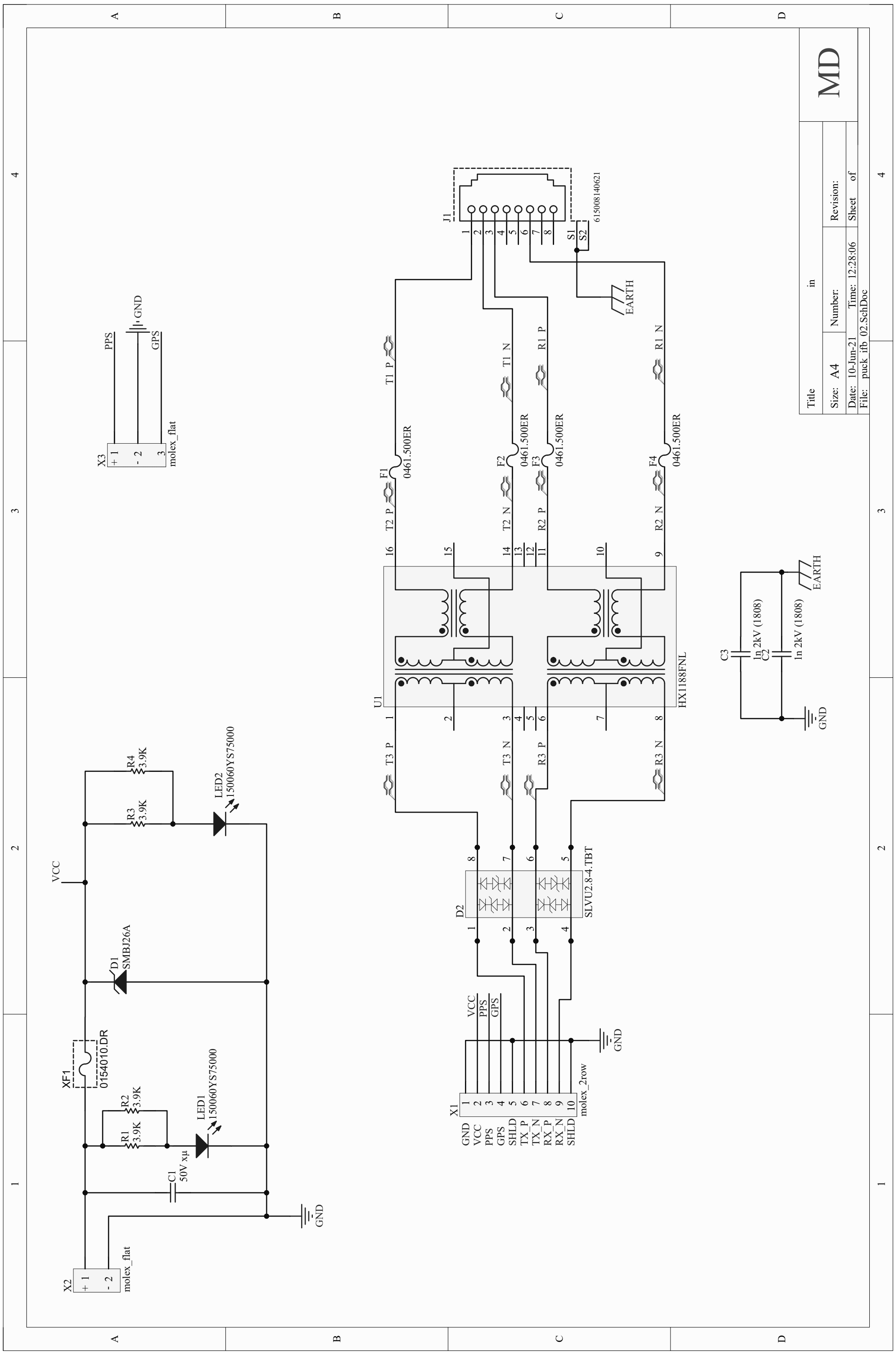

X301
+ 1
- 2
molex_flat
C302
2µ2
R301
470R
C303
1µ
R302
6k19
R303
6k19
R310
6k19
R311
6k19
D1
RED 2V
XF301
0154010.DR
D9
SMBJ26A
VCC
R304
6k19
R305
6k19
R312
6k19
R313
6k19
D2
RED 2V
R306
3.9K
R314
3.9K
R317
3.9K
nSHDN
R318
210k
Q301
Q302
SQ4050EY-T1_GE3
SQ4050EY-T1_GE3
U301
VIN
UV
OV
SHDN
VOUT
GATE
FAULT
GND
LTC4365CTS8#TRMPBF
nFAULT
R307
6k19
R308
6k19
R315
6k19
R316
6k19
D3
YELLOW 2V
R309
21k5
C304
1µ
C301
100µ
PWR_IN
GND
Up is start/normal. Down is stop
3V3
R321
147k
R322
147k
SW301
JS202011SCQN
R319
147k
R320
21k5
C305
1µ
U302F
SN74LV14APWR
U302E
SN74LV14APWR
U303C
SN74LV125APWR
APX_PWR_OFF
R323
147k
R324
21k5
C306
1µ
Equivalent to an OR gate
U302B
SN74LV14APWR
U302A
SN74LV14APWR
U304C
SN74LV08APWRG4
U302C
SN74LV14APWR
nSHDN
V divider 5 -> 3.3 and weak pulldown
APX_5V_OUT
R329
100k
R326
210k
Turns on when power at chip is good
R325
210k
U304D
SN74LV08APWRG4
nFAULT
R327
470R
R328
470R
D4
GREEN 2V
U305
TLF80511TFV33ATMA1
IN
OUT
TAB
C307
0.1µ
C308
1µ
C309
1µ
C310
4.7µ
U302G
SN74LV14APWR
C311
0.1µ
U304E
SN74LV08APWRG4
C312
0.1µ
U303E
SN74LV125APWR
C313
0.1µ
U303D
SN74LV125APWR
U304B
SN74LV08APWRG4
U304A
SN74LV08APWRG4
U302D
SN74LV14APWR
U303B
SN74LV125APWR
U303A
SN74LV125APWR
Title
in
APX15_IFB
Size: A4
Number:
Revision:
Date: 10-Jun-21
Time: 12:25:31
Sheet of
File: APX15_PWR.SchDoc
MD

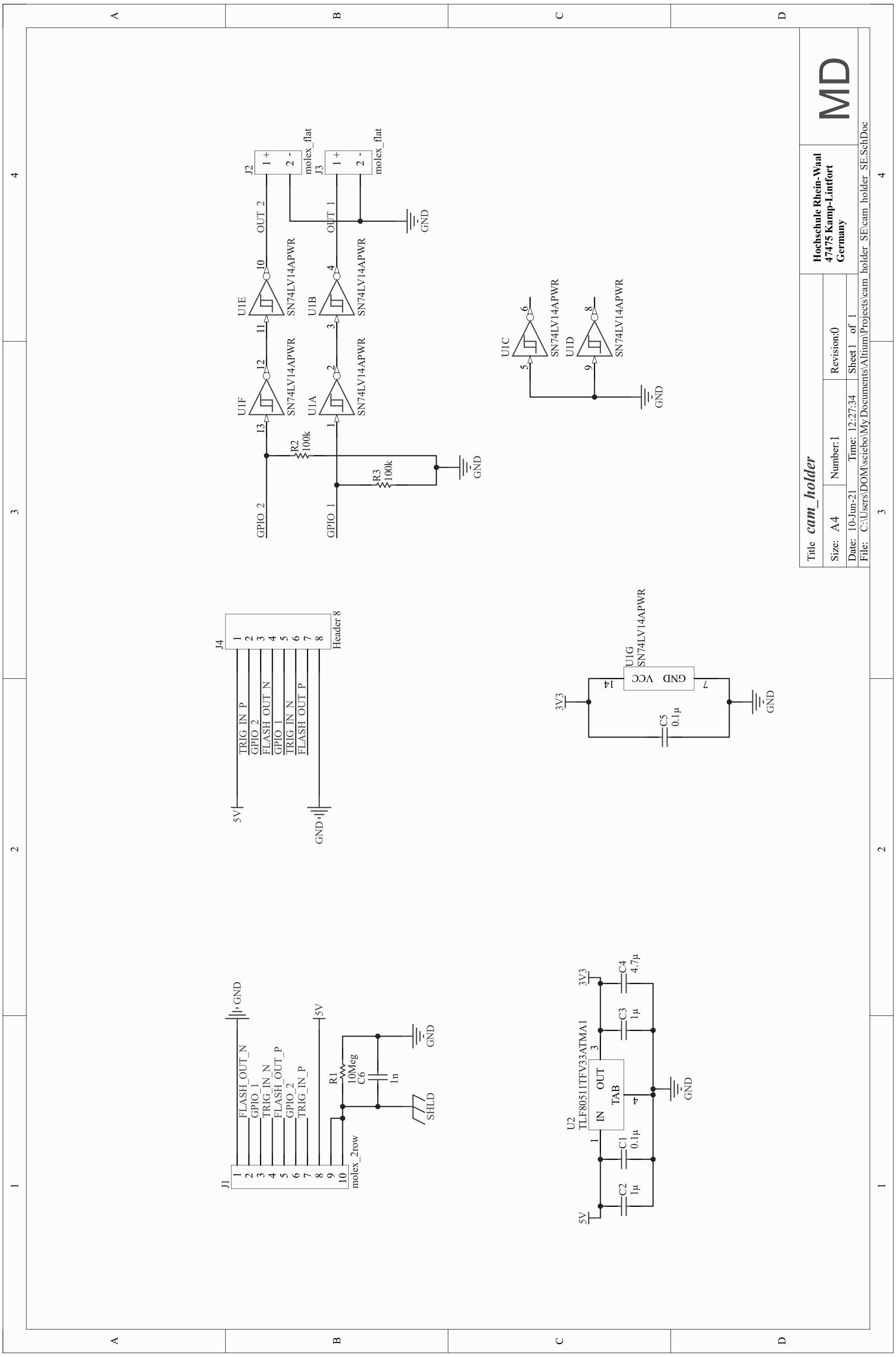

R4
470R
D8
YELLOW 2V
R3
470R
D7
RED 2V
R2
470R
D6
GREEN 2V
R1
470R
D5
RED 2V
APX_IMU_NAV_LED
APX_SAT_LED
APX_RTK_LED
APX_PWR_LED
APX_GND
Title
in APX15_IFB
Size: A4
Number:
Revision:
Date: 10-Jun-21
Time: 12:25:57
Sheet of
File: APX15_LEDS.SchDoc
MD
A
B
C
D
1
2
3
4

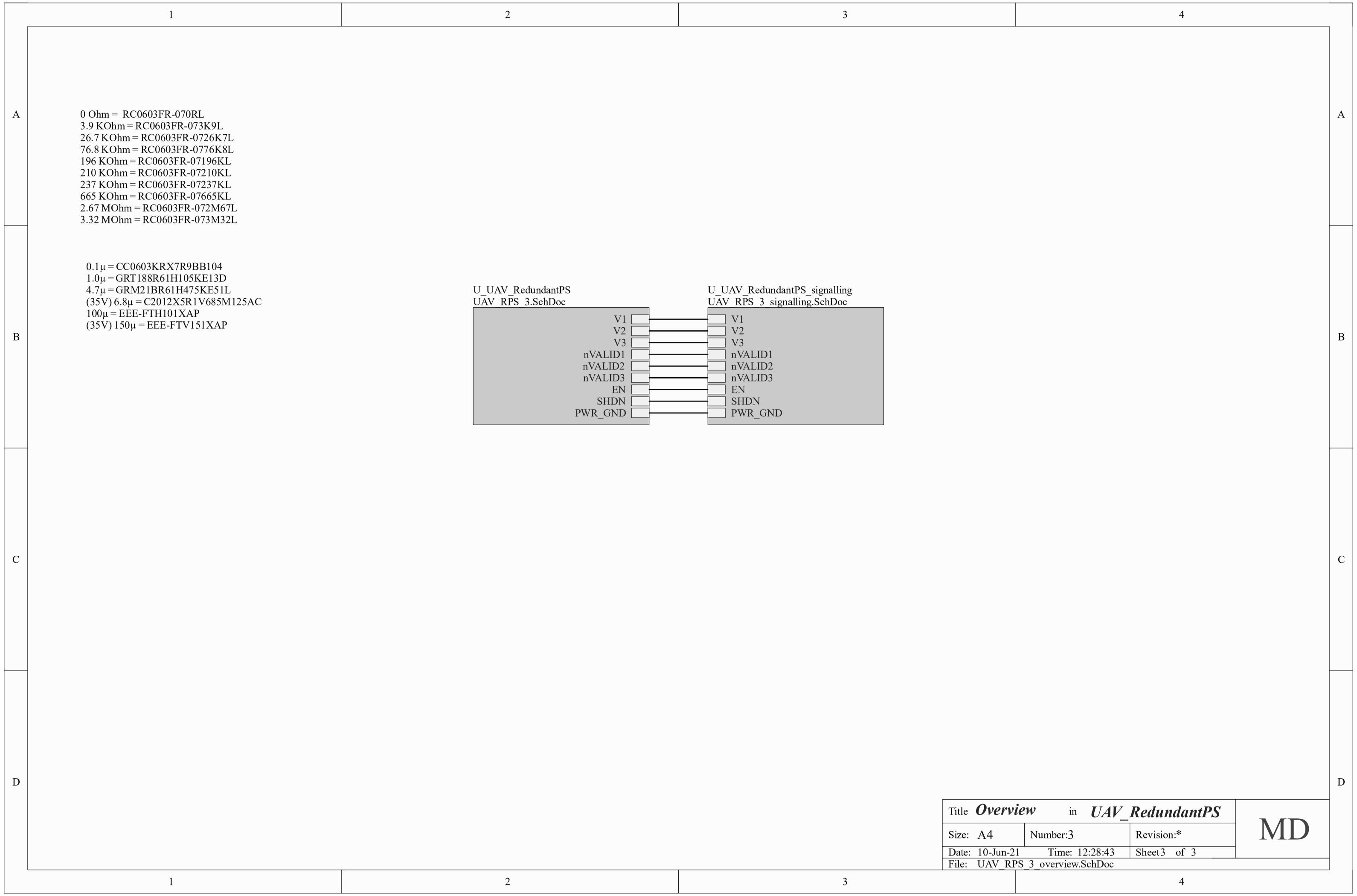

J701
615008140621
1 2 3 4 5 6 7 8 S1 S2
P4_N
P4_P
P3_N
P3_P
R701 49R9
R702 49R9
R703 49R9
R704 49R9
R705 49R9
R706 49R9
R707 75R
R708 75R
C704 1n 2kV (1808)
TX_N
MT_P
TX_P
RX_N
MT_N
RX_P
F701 0461.500ER
F702 0461.500ER
F703 0461.500ER
F704 0461.500ER
TDX_N
TDX_P
RDX_N
RDX_P
U701
HX1188FNL
TD_N
TD_P
RD_N
RD_P
D701
SLVU2.8-4.TBT
ETH_TD_N
ETH_TD_P
ETH_RD_N
ETH_RD_P
ETH_GND
C701 1n 2kV (1808)
C702 1n 2kV (1808)
C703 1n 2kV (1808)
EARTH
Title APX15_IFB
Size: A4
Number:
Revision:
Date: 10-Jun-21
Time: 12:26:18
Sheet of
File: APX Ethernet.SchDoc
MD
A B C D
1 2 3 4


X401 (BOT)
1 +
2 -
3
molex_flat
APX_PPS
APX_GPS_GND
APX_GPS_TX
Title
in APX15_IFB
Size: A4
Number:
Revision:
Date: 10-Jun-21
Time: 12:26:36
Sheet of
File: APX15_GPS_OUT.SchDoc
MD
A
B
C
D
1
2
3
4


A
B
C
D
1
2
3
4
X601 (BOT)
1 +
2 -
3
molex_flat
RTCM_PWR
RTCM_GND
RTCM_RX
Title
in APX15_IFB
Size: A4
Number:
Revision:
Date: 10-Jun-21
Time: 12:26:55
Sheet of
File: APX15_RTCM.SchDoc
MD


MD
Title
in
Size: A4
Number:
Revision:
Date: 10-Jun-21
Time: 12:28:06
Sheet of
File: puck ifb 02.SchDoc
J1
615008140621
S1
S2
EARTH
T1 P
T1 N
R1 P
R1 N
F1
F2
F3
F4
0461.500ER
T2 P
T2 N
R2 P
R2 N
U1
HX1188FNL
T3 P
T3 N
R3 P
R3 N
D2
SLVU2.8-4.TBT
C3
C2
1n 2kV (1808)
GND
X3
molex_flat
PPS
GPS
VCC
R1
R2
R3
R4
3.9K
LED1
LED2
150060YS75000
D1
SMBJ26A
XF1
0154010.DR
C1
50V xµ
X2
X1
molex_2row
SHLD
TX_P
TX_N
RX_P
RX_N

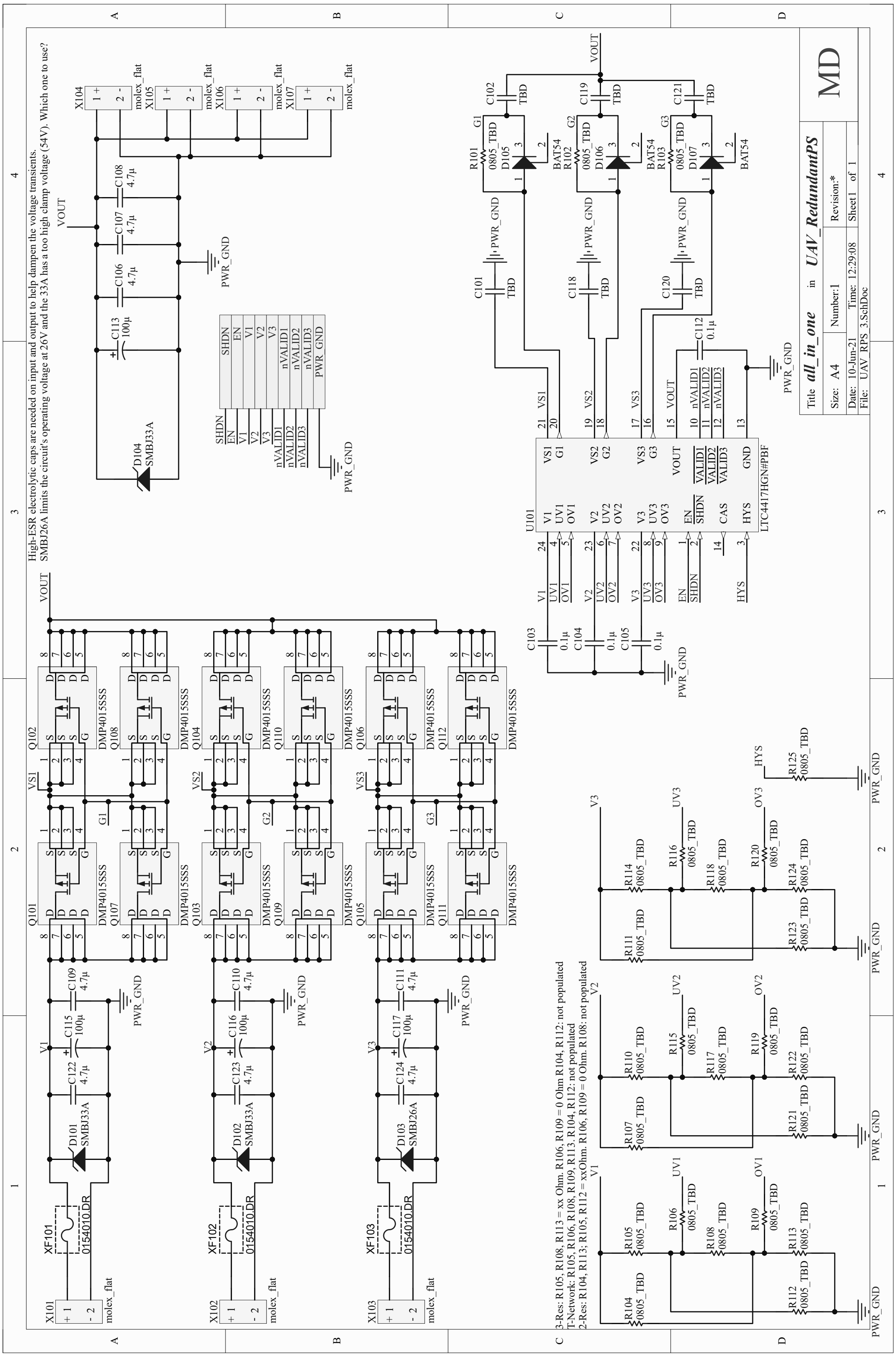

Title cam_holder
Size: A4
Number:1
Revision:0
Date: 10-Jun-21
Time: 12:27:34
Sheet 1 of 1
File: C:\Users\DOM\sciebo\My Documents\Altium\Projects\cam_holder_SE\cam_holder_SE.SchDoc
Hochschule Rhein-Waal
47475 Kamp-Lintfort
Germany
MD
J1
molex_2row
FLASH_OUT_N
GPIO_1
TRIG_IN_N
FLASH_OUT_P
GPIO_2
TRIG_IN_P
5V
GND
R1
10Meg
C6
1n
SHLD
J4
Header 8
TRIG_IN_P
GPIO_2
FLASH_OUT_N
GPIO_1
TRIG_IN_N
FLASH_OUT_P
J2
J3
molex_flat
OUT_2
OUT_1
GPIO_2
GPIO_1
R2
100k
R3
100k
U1A
U1B
U1C
U1D
U1E
U1F
U1G
SN74LV14APWR
VCC
GND
C5
0.1µ
3V3
U2
TLF80511TFV33ATMA1
IN
OUT
TAB
C1
0.1µ
C2
1µ
C3
1µ
C4
4.7µ
A
B
C
D
1
2
3
4

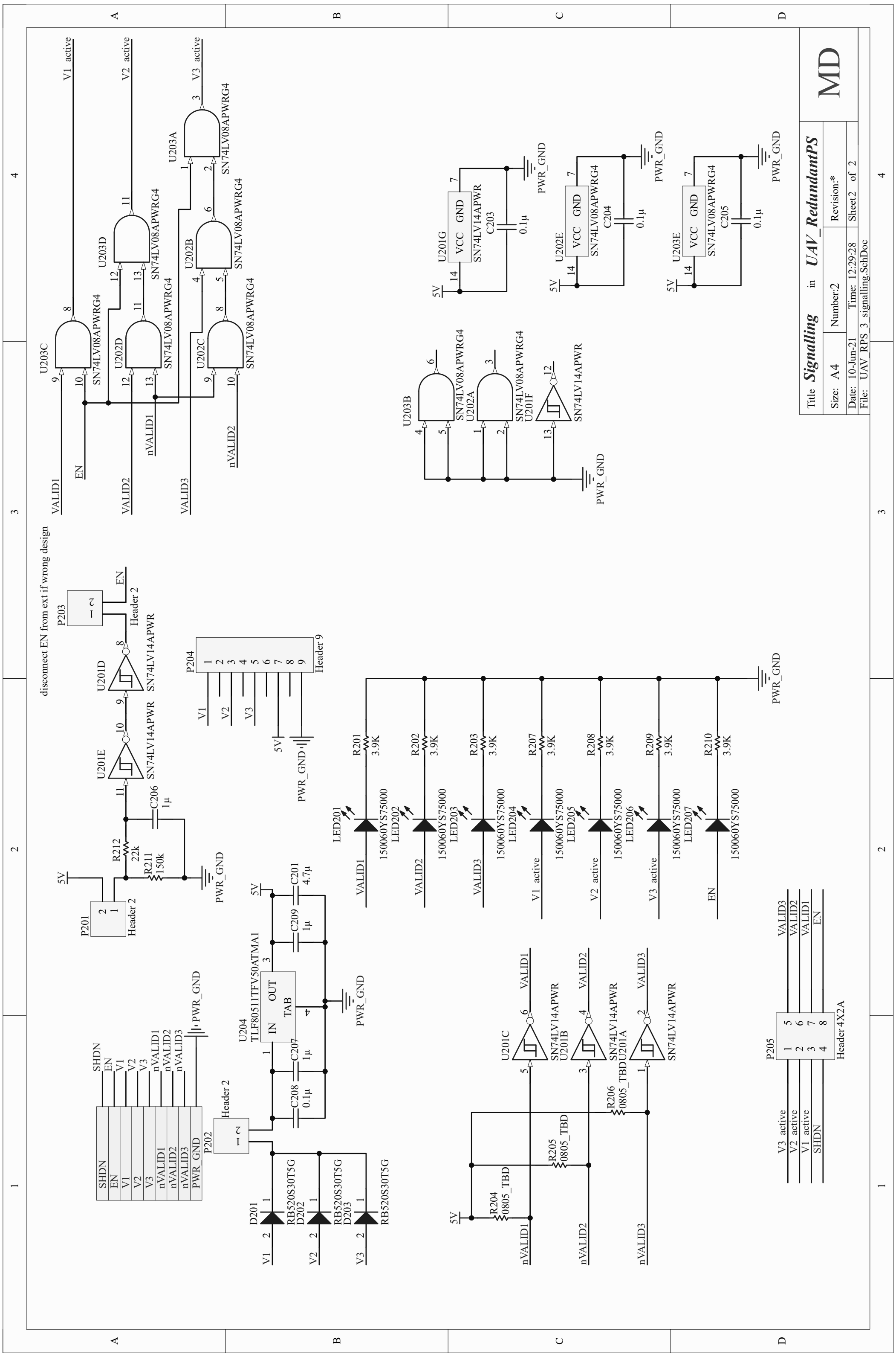


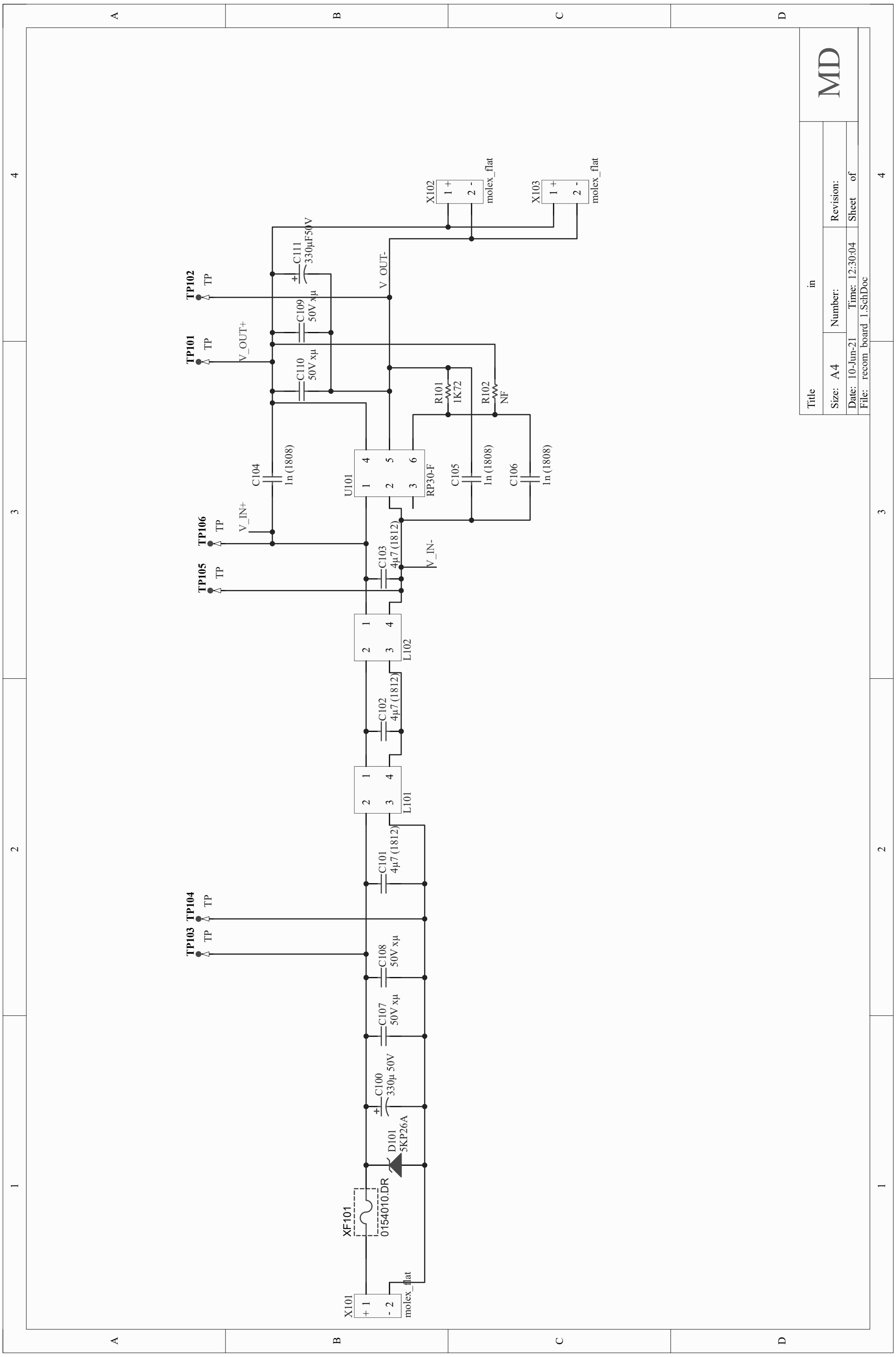
0 Ohm = RC0603FR-070RL
3.9 KOhm = RC0603FR-073K9L
26.7 KOhm = RC0603FR-0726K7L
76.8 KOhm = RC0603FR-0776K8L
196 KOhm = RC0603FR-07196KL
210 KOhm = RC0603FR-07210KL
237 KOhm = RC0603FR-07237KL
665 KOhm = RC0603FR-07665KL
2.67 MOhm = RC0603FR-072M67L
3.32 MOhm = RC0603FR-073M32L

0.1µ = CC0603KRX7R9BB104
1.0µ = GRT188R61H105KE13D
4.7µ = GRM21BR61H475KE51L
(35V) 6.8µ = C2012X5R1V685M125AC
100µ = EEE-FTH101XAP
(35V) 150µ = EEE-FTV151XAP

Title ***Overview*** in ***UAV_RedundantPS***			MD
Size: A4	Number:3	Revision:*	
Date: 10-Jun-21 Time: 12:28:43		Sheet3 of 3	
File: UAV_RPS_3_overview.SchDoc			


High-ESR electrolytic caps are needed on input and output to help dampen the voltage transients.
SMBJ26A limits the circuit's operating voltage at 26V and the 33A has a too high clamp voltage (54V). Which one to use?
3-Res: R105, R108, R113 = xx Ohm. R106, R109 = 0 Ohm R104, R112: not populated
T-Network: R105, R106, R108, R109, R113. R104, R112: not populated
2-Res: R104, R113; R105, R112 = xxOhm. R106, R109 = 0 Ohm. R108: not populated
X101
X102
X103
X104
X105
X106
X107
molex_flat
XF101
XF102
XF103
0154010.DR
D101
D102
SMBJ33A
D103
SMBJ26A
D104
SMBJ33A
C122
C123
C124
C115
C116
C117
C109
C110
C111
C106
C107
C108
C113
4.7µ
100µ
Q101
Q102
Q103
Q104
Q105
Q106
Q107
Q108
Q109
Q110
Q111
Q112
DMP4015SSS
V1
V2
V3
VS1
VS2
VS3
G1
G2
G3
VOUT
PWR_GND
SHDN
EN
nVALID1
nVALID2
nVALID3
U101
LTC4417HGN#PBF
UV1
OV1
UV2
OV2
UV3
OV3
CAS
HYS
GND
C101
C102
C103
C104
C105
C112
C118
C119
C120
C121
TBD
0.1µ
R101
R102
R103
0805_TBD
D105
D106
D107
BAT54
R104
R105
R106
R107
R108
R109
R110
R111
R112
R113
R114
R115
R116
R117
R118
R119
R120
R121
R122
R123
R124
R125
A
B
C
D
1
2
3
4
Title all_in_one in UAV_RedundantPS
Size: A4
Number:1
Revision:*
Date: 10-Jun-21
Time: 12:29:08
Sheet1 of 1
File: UAV_RPS_3.SchDoc
MD


Title Signalling in UAV_RedundantPS
Size: A4
Number:2
Revision:*
Date: 10-Jun-21
Time: 12:29:28
Sheet2 of 2
File: UAV_RPS_3_signalling.SchDoc
MD
disconnect EN from ext if wrong design
U203C SN74LV08APWRG4
U203D SN74LV08APWRG4
U203A SN74LV08APWRG4
U202D SN74LV08APWRG4
U202B SN74LV08APWRG4
U202C SN74LV08APWRG4
U203B SN74LV08APWRG4
U202A SN74LV08APWRG4
U201F SN74LV14APWR
U201G SN74LV14APWR
U202E SN74LV08APWRG4
U203E SN74LV08APWRG4
C203 0.1µ
C204 0.1µ
C205 0.1µ
V1_active
V2_active
V3_active
VALID1
VALID2
VALID3
nVALID1
nVALID2
nVALID3
EN
SHDN
5V
PWR_GND
P203 Header 2
U201D SN74LV14APWR
U201E SN74LV14APWR
C206 1µ
R212 22k
R211 150k
P201 Header 2
P204 Header 9
P202 Header 2
U204 TLF80511TFV50ATMA1
IN OUT TAB
C201 4.7µ
C209 1µ
C207 1µ
C208 0.1µ
D201 RB520S30T5G
D202 RB520S30T5G
D203 RB520S30T5G
R201 3.9K
R202 3.9K
R203 3.9K
R207 3.9K
R208 3.9K
R209 3.9K
R210 3.9K
LED201 150060YS75000
LED202 150060YS75000
LED203 150060YS75000
LED204 150060YS75000
LED205 150060YS75000
LED206 150060YS75000
LED207 150060YS75000
U201C SN74LV14APWR
U201B SN74LV14APWR
U201A SN74LV14APWR
R204 0805_TBD
R205 0805_TBD
R206 0805_TBD
P205 Header 4X2A
A B C D
1 2 3 4


A
B
C
D
1
2
3
4
X101
+ 1
- 2
molex_flat
XF101
0154010.DR
D101
5KP26A
C100
330µ 50V
C107
50V xµ
C108
50V xµ
TP103
TP
TP104
TP
C101
4µ7 (1812)
L101
1
2
3
4
C102
4µ7 (1812)
L102
1
2
3
4
TP105
TP
C103
4µ7 (1812)
V_IN-
TP106
TP
V_IN+
C104
1n (1808)
U101
RP30-F
1
2
3
4
5
6
C105
1n (1808)
C106
1n (1808)
R101
1K72
R102
NF
TP101
TP
V_OUT+
C110
50V xµ
C109
50V xµ
TP102
TP
C111
330µF50V
V_OUT-
X102
1 +
2 -
molex_flat
X103
1 +
2 -
molex_flat
MD
Title
in
Size: A4
Number:
Revision:
Date: 10-Jun-21
Time: 12:30:04
Sheet of
File: recom_board_1.SchDoc

B. LIDAR COVERAGE VS DISTANCE

The current Section claims that, *all other things being equal, the chance to hit a point with a lidar pulse increases roughly linearly with flying altitude*. One's first instinct might be that because the spacing between lidar pulses increases with height, therefore lowering the pulse density, the above statement should be false. However laser pulses are not simply points, they have a non-zero width (w) and height (h), both of which increase with distance. And as it turns out, the area covered by laser pulses increases quadratically, while the point density decreases only linearly.

Consider the following scenario: a lidar unit that emits laser pulses at the rate of f_{pulse} is being moved in a straight line with constant speed v and we measure the area covered by all of its pulses $A_{\text{total}} = \sum A_{\text{pulse}}$ at a distance R. By comparing A_{total} to the area of a cylinder of radius R (A_{cyl}), we can make statements about the chance of points on the cylinder's surface to be hit by a laser pulse.

The area of a single pulse at a distance R can be calculated as $A_{\text{pulse}} = (w_0 + Rc_w)(h_0 + Rc_h)$, where c_w and c_h are constants describing the beam divergence in the cross-track and along-track directions respectively. The above can be simplified as $A_{\text{pulse}} \approx Rc_wRc_h$ when R is sufficiently large that the initial footprint size does not matter. In the case of the puck, the initial dimensions can be safely ignored from about $R \approx 10$ m. So the area of a single pulse becomes $A_{\text{pulse}} \approx R^2c_wc_h$. For a given interval t, the lidar emits $N = f_{\text{pulse}}t$ pulses. So the total area covered by laser pulses becomes $A_{\text{total}} \approx f_{\text{pulse}}tR^2c_wc_h$. For the same interval, the lidar moves a distance of $l = vt$ metres. So the area of the cylinder to be considered becomes $A_{\text{cyl}} = 2\pi Rl = 2\pi Rvt$. The ratio of the areas (η) becomes:

$$\eta \approx \frac{A_{\text{total}}}{A_{\text{cyl}}} = \frac{f_{\text{pulse}}\cancel{t}R^{\cancel{2}}c_wc_h}{2\pi\cancel{R}v\cancel{t}} \tag{B.1}$$

$$\eta \approx \frac{f_{\text{pulse}}c_wc_h}{2\pi v}R \tag{B.2}$$

Plugging in the numbers for the puck ($f_{\text{pulse}} = 3 \times 10^5$, $c_w = 3.0 \times 10^{-3}$, $c_h = 1.6 \times 10^{-3}$) and a flying speed $v = 5.0$ m/s, the following η are obtained for $R = 20$ and $R = 100$ m respectively:

$$\eta_{\text{20m}} \approx 0.92 \qquad\qquad \eta_{\text{100m}} \approx 4.59 \tag{B.3}$$

Thus it is shown that at a flying height of 100 m, points on ground fairly close to the imaginary cylinder surface are hit on average by 4.6 pulses, vs only 0.9 pulses when flying at 20 m.

C. MISCELLANEOUS

C.1. NMEA SENTENCES

“NMEA” refers to NMEA0183: a specification of data formats and electrical interfaces to be used in marine electronics. The standard has been developed by the National Marine Electronics Association. It has since been broadly adopted by GNSS-receivers even outside the field of marine navigation. In the current work, all mentions of “NMEA” refer distinctively to the data specification. The standard itself is proprietary and costs upwards of 1000 USD (NMEA, 2021), however GNSS manufacturers (Trimble, n.d.) and other parties (Raymond, 2021) publish information that is sufficient for decoding and understanding NMEA messages. The following shortly explains their structure.

NMEA sentences are a class of messages generated by GNSS receivers. Besides the position and time solution obtained by the receiver, they can contain a myriad of other information, varying from current datum to position error statistics. The NMEA sentences are essentially text strings which follow a clearly defined structure: they always begin with a dollar sign or an exclamation mark, contain a variable number of fields delimited by commas, and end with five characters: an asterisk, a two-character checksum and CRLF.[56] The first field always contains five capital letters: two for talker-ID, meaning the satellite constellation used, and three for message type. Usual talker-IDs are GP/GL (when only GPS or GLONASS is used) and GN (GNSS, when at least two constellations are used). The message type determines the number and expected content of fields in the rest of the sentence. For instance, a message of type “ROT” contains two fields: rate of turn in degrees/minute and a data valid indicator, while an “RMC” message contains ten fields conveying information about position, velocity and time.

---

[56] CR refers to “carriage return” and LF to “line feed”—control codes in ASCII which essentially mean “go to the beginning of the line” and “advance to next line”. They are used together or separately to delimit lines in text files.

DECLARATION OF AUTHENTICITY

I, Marcel Dogotari, hereby declare that the work presented herein is my own work completed without the use of any aids other than those listed. Any material from other sources or works done by others has been given due acknowledgement and listed in the reference section. Sentences or parts of sentences quoted literally are marked as quotations; identification of other references with regard to the statement and scope of the work is quoted. The work presented herein has not been published or submitted elsewhere for assessment in the same or a similar form. I will retain a copy of this assignment until after the Board of Examiners has published the results, which I will make available on request.