Artificial Neural Networks for Individual Tracking and Characterization of Wake Vortices in LIDAR Measurements

Anton Stephan[1], Grigory Rotshteyn[2], Niklas Wartha[3], Frank Holzäpfel[4], Nicolas Petross[5], Lars Stietz[6]
Institut für Physik der Atmosphäre, Deutsches Zentrum für Luft- und Raumfahrt,
82234 Oberpfaffenhofen, Germany

A vortex detection and characterization strategy in LiDAR measurements (Light Detection and Ranging), consisting of two different parts, is presented in this study. First, a two-stage detection pipeline is implemented combining the computer vision deep neural network YOLO (You Only Look Once) and a regression convolutional neural network to detect vortices individually. An accuracy on the order of the instrument accuracy is achieved. Second, a new characterization method the so-called Projection Method is presented which has has the following important features. First, it is very accurate. Second, it yields valuable quantification of the accuracy of the results. Third, it is very fast and outperforms conventional methods by two to three orders of magnitude, hence it enables to process a high amount of data in a short time. Fourth, it provides much flexibility in choosing different models and compare the performance of the respective models. Fifth, it comes with a natural visualization of the underlying calculus. Sixth, it can be generalized to situations, where measurements provide a reduced and skewed image of the reality and certain structures or features have to be identified and characterized employing models.

I. Introduction

Pulsed coherent Doppler LiDARs (PCDLs) are currently considered to be one of the best technical facilities for the experimental investigation of mesoscale airflows. They are widely used in different situations from meteorological flows, like the turbulent boundary layer [1], to technogenic flows like the wake behind wind turbines [2] or aircraft wakes [3]. Due to the nature of that instrument measurements with a LiDAR have two major shortcomings. First, they only reveal the projection of the three-dimensional wind velocity vector to the laser beam direction. Second, they provide a smoothed or filtered version of the actual wind field. It is therefore a challenge to identify, interpret and characterize flow pattern of interest. Particularly, coherent structures like vortices that occur behind flying aircraft, wind turbines, buildings other objects are difficult to identify and characterize. In this study we present an effective and very fast method to first identify and locate vortices and second characterize them in terms of circulation and core radius in the framework of aircraft wake vortices. The methods presented here are easy to adapt for other applications. Vortices in a flow field can be characterized by their position, their strength in terms of circulation, by their core radius, and by their tangential velocity profile. The position of multiple vortices x_i in a slice when the full 3-d flow field is given, can be determined by

$$X_i = \frac{1}{\Gamma} \iint x_i \omega_x dy dz, \tag{1}$$

[1]Research Scientist(Anton.Stephan@dlr.de).
[2]Research Scientist.
[3]PhD Candidate.
[4]Senior Scientist, Associate Fellow AIAA.
[5]Graduate Student.
[6]Graduate Student.

where ω_x denotes the vorticity perpendicular to the slice. The circulation of each of the vortices, again assuming full 3-d information, can be calculated by the well known formula

$$\Gamma = \oint_{\partial A} \vec{u} \cdot d\vec{s} = \int_A \omega dA. \tag{2}$$

The core radius is defined as the maximum point of the tangential velocity profile. Higher order moments can be captured by matching certain velocity profiles to the tangential velocity profile. We will present a method on how to retrieve all these characteristics from LiDAR measurements, without making any additional assumptions.

As a special case, where range height indicator (RHI) scans are performed by a LiDAR device, we will pick out aircraft wake vortex measurements during landing. Of course, the presented methods easily transfer to other scan patterns and other technical or scientific applications.

Flying aircraft experience lift and therefore create trailing vortices that may dissipate very slowly and hence pose a risk for other aircraft if they encounter that coherent structure. At airports, where wake vortices descend is limited by the ground surface, aircraft separation rules are enforced to mitigate wake vortex encounter risk. LiDAR measurements became widely used when it comes to direct physical investigation of the phenomenon, sometimes complemented by numerical simulations. These measurements enable to study general wake vortex physics, but they can be also used to build situation awareness, if full necessary information is retrieved in real-time. An online monitoring system may help air traffic controllers to take right decisions in critical situations. Finally, if it comes to establish new separation rules, LiDAR measurements can help to prove a safety case.

II. A vortex detection pipeline with Artificial Neural Networks

The underlying data set was acquired during the Vienna measurement campaign with LiDAR measurements at Vienna International Airport from May 2019 until November 2019 by DLR [4]. The data contains the radial velocity RHI LiDAR scans. The scans in the LiDAR data set from Vienna include atmospheric effects, secondary vortices, noise, and measurement errors. The data set also contains corresponding labels generated with the method of radial velocities (RV-method) [3] with the vortex center locations in polar coordinates $R_t, \varphi_t \in \mathbb{R}_{\geq 0}$ and vortex circulation $\Gamma_t \in \mathbb{R}_{\geq 0}$. Hence labels for supervised training and evaluation are given.

We set up a prediction pipeline to obtain the individual vortices from the YOLOv4 prediction and input those into a regression CNN (Convolutional Neural Network). The regression net employs cut-out vortices from the LiDAR scans in Cartesian coordinates to further refine the results. Throughout this study, we use the terms *prediction* and *estimation* equivalently. In essence, prediction does not represent a prediction in time.

A. Data set

Since the Vienna measurement campaign was conducted to evaluate the effectiveness of plate lines in wake vortex mitigation, the data set contains measurements with and without plate line usage. Hence in some scans, plate line effects are present, which means, that the vortices are disturbed from their natural shape. In the following, a summary of the measurement campaign is given based on [5].

Figure 1 depicts the setup of the measurement instruments at the runway of the Vienna International Airport. During the campaign, two plate lines were installed consisting of respectively eight and nine plates of dimension $4.5\,\mathrm{m} \times 9\,\mathrm{m}$ displayed by red dashes. For the radial velocity measurement, at most three Leosphere Windcube 200S (1.543 μm) micro-PCDLs were used at once, positioned at three out of five possible positions (L1-L5). At positions A-C, additional meteorological instruments were placed. The runway is at the bottom of Fig. 1, and the aircraft approaches from the top. The average flight altitudes above ground at LiDAR planes are 40.8 m at L1, 45.8 m at L2, 54.3 m at L3, 64.8 m at L4, and 74.5 m at L5 with a standard deviation of 4.9 m [25]. The LiDAR measured the radial velocities at discrete points along each line of sight (LOS) perpendicular to the runway. Each LOS has 151 measuring points, called range gates, starting at a range of $R_{min} = 80\,\mathrm{m}$ to a range of $R_{max} = 530\,\mathrm{m}$ with a step size of $\Delta R = 3\,\mathrm{m}$. The minimum and maximum elevation angles used depend on the LiDAR position adjusted to the average flight altitudes at the different LiDAR positions. The elevation angle step size was $\Delta\varphi = 0.2^o$ for all positions. The corresponding minimum and maximum elevation angles used can be found in Table 2. A list of three-tuples can therefore describe a raw LiDAR scan

$$(R_j, \varphi_j, V_r(R_j, \varphi_j)) \in \mathbb{R}^3, \tag{3}$$

with $V_r(R_j, \varphi_j)$ measured at (R_j, φ_j). The polar coordinates (R_i, φ_j) represent at which the corresponding radial velocity $V_r(\varphi_i, R_j)$ was measured, with $i \in \{0, 1, \ldots, H_{\mathrm{lid}}\}$ and $j \in \{0, 1, \ldots, 150\}$.

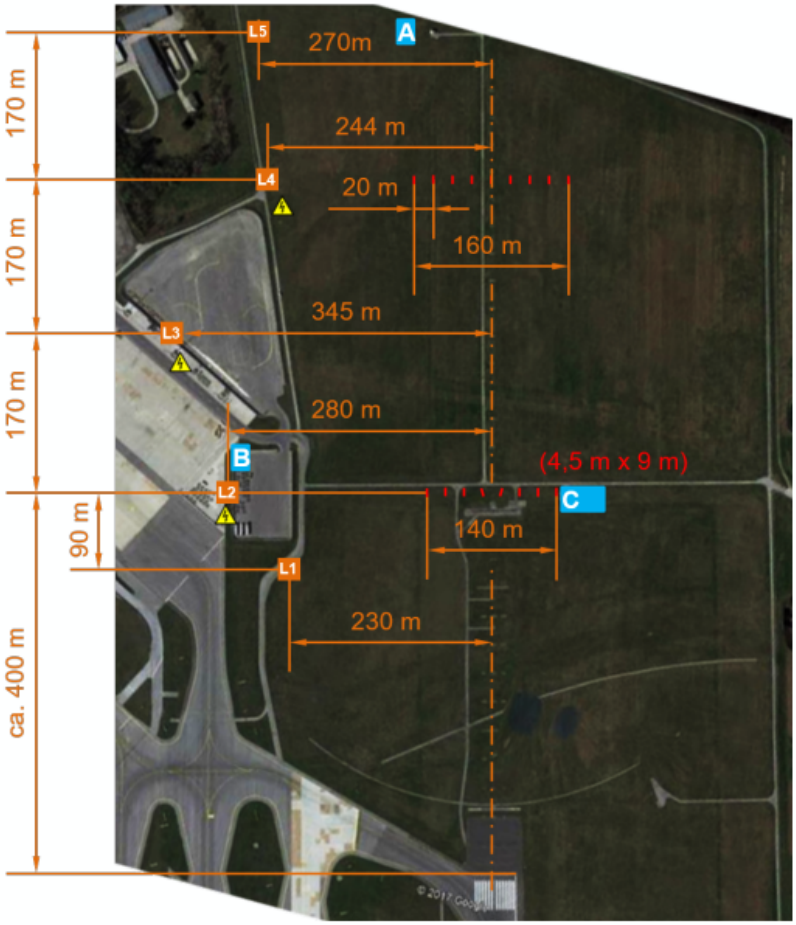
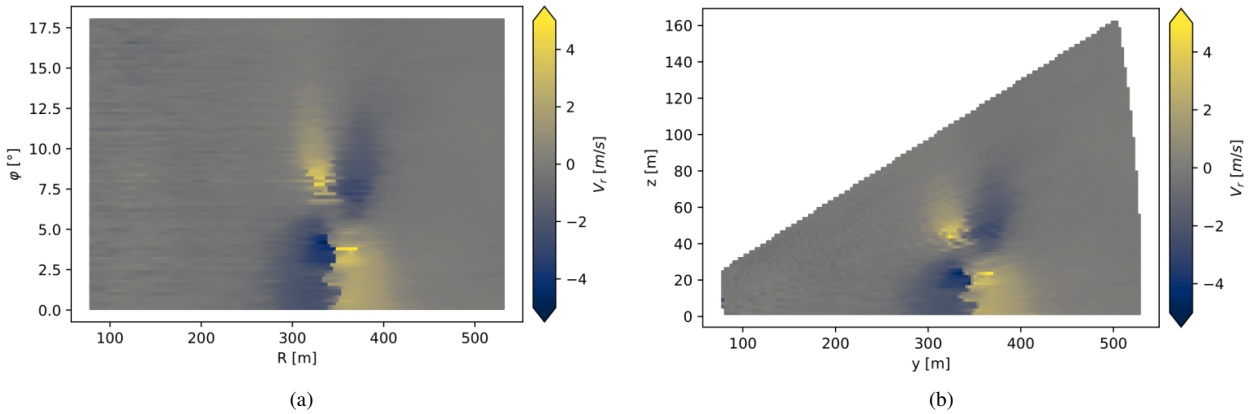


Fig. 1 Vienna measurement campaign setup of instrumentation with L1-L5 being the LiDAR positions, A-C being additional meteorological instruments, and the red dashes being the plates, from [4]



(a) (b)

Fig. 2 (a) Example of a raw LiDAR scan. (b) Example of a raw LiDAR scan transformed to Cartesian coordinates.

Table 1 The elevation angles covered correspond to each LiDAR position and the number of LOSs needed.

LiDAR position	φ range	H_{lid} number of LOS beams
L1	$0° - 25°$	125
L2	$0° - 20°$	100
L3	$0° - 18°$	90
L4	$1° - 28°$	135
L5	$0° - 29°$	145

An example of a raw LiDAR scan can be seen in Fig. 2 (a) with the corresponding scan transformed to Cartesian coordinates in Fig. 2 (b). The grid in polar coordinates is equidistant. Hence, transforming to Cartesian coordinates, one loses this property.

YOLOv4 needs bounding box labels, thus we focus on scans in Cartesian coordinates. The input format required for YOLOv4 is that of images. Thus we have to transform the polar coordinates to an equidistant Cartesian grid, which requires data interpolation. This may lead to inaccuracies in the YOLOv4 detection result, especially for high range gate values, as the distance between two grid points at the same range gate but with different elevation angles increases with the range gate. Another source of inaccuracies is the time a single measurement takes. Changing the LOSs takes 50 ms. Hence a LiDAR scan cannot be instantaneous. This time difference results in radial velocities measured with a maximal time difference of 7.25 s. The LiDAR scans were initiated at either end of the elevation angle interval. Consequently, the resulting data set contains roughly an equal amount of data with a time distortion from top to bottom and vice versa. Thus this effect can be considered negligible.

B. Object Detection

The task of *object detection* is not only to classify objects in an image but also to localize those objects. ANNs (Artificial Neural Networks) performing object detection aim to mark existing objects in any image with a rectangular *bounding box* [6]. We define a bounding box as follows.

Definition II.1 (Bounding Box)**.** Let $(x_c, y_c, w, h) \in \mathbb{R}^4$, with (x_c, y_c) being the center point and (w, h) being the dimension defining the rectangular bounding box $\mathbf{B}$ as

$$\mathbf{B} \coloneqq \left\{ (x, y) \in \mathbb{R}^2 : x_c - \frac{w}{2} \leq x \leq x_c + \frac{w}{2}, y_c - \frac{h}{2} \leq y \leq y_c + \frac{h}{2} \right\}. \tag{4}$$

There are two main machine learning approaches to object detection. The first approach starts with a network that proposes regions for interest, the *region-proposal network* (RPN). A second network - of detection type - then classifies objects in those regions of interest [7]. Networks following this approach are called *two-stage* detectors. The second approach is the *one-stage* detectors. One-stage detectors accomplish regression and classification in a single shot. The most prominent one-stage detector is YOLO. We use the YOLOv4 version [8][1].

Because the one-stage detectors do not need an RPN, they are faster than the two-stage detectors enabling fast-time detection. In case of wake vortex measurements and wake vortex tracking fast-time detection is crucial. YOLOv4 outperformed other state-of-the-art object detection networks in terms of accuracy and speed [8].

Despite the difference of having two stages or only one to do object detection, modern object detection networks consist of three main parts, the *backbone*, the *neck*, and the *head* illustrated in Figure 3 [8]. The backbone network, in most cases, is pretrained for the ImageNet classification task [9]. The task of the backbone is to extract higher-level features of an image. The role of the neck is to collect feature maps from different stages of the backbone and is generally composed of several *top-down-paths* and *bottom-up-paths*. The idea is to let lower-level features and higher-level features interact. The head predicts the bounding boxes and classification.

C. Labels - Radial Velocity Method

The labels for the data set originate from the assessment of the measurement campaign, which used the RV method, described in [3]. The idea behind the RV method is first to estimate the center of the vortices and afterward estimate the

[1] Meanwhile YOLOv7 was published, which is even superior to YOLOv4.

circulations via minimizing a functional. The RV method is described as an example for two vortices. To estimate the range gate of the vortex center R_{C_i}, the maxima of

$$D(R_k) := \sum_{j=0}^{B_{\text{lid}}} V_r(R_k, \varphi_j)^2 \tag{5}$$

are sought [3]. The corresponding angles can be found by calculating the mean of the angle $\varphi_{\min}(R_{C_i})$ of minimal radial velocity and $\varphi_{\max}(R_{Ci})$ at the estimated range gate center [3]. A functional

$$\rho(\Gamma_i) = \sum_{j=0}^{B_{lid}} (V_r(R_{C_i}, \varphi_j) - \widetilde{V}_r(R_{C_i}, \varphi_j; \Gamma_1, \Gamma_2))^2 \tag{6}$$

is minimized to fit the best circulation. This functional (6) describes the difference between the radial velocities at the axis of the center range gate of the true LiDAR scan $V_r(R_{C_i}, \varphi_j)$ and a modeled scan $\widetilde{V}_r(R_{C_i}, \varphi_j; \Gamma_1, \Gamma_2)$ calculated theoretically with arbitrary circulations Γ_1 and Γ_2.

During the measurement campaign, 9 473 approaches were measured with approximately 20 scans per overflight [4]. Since the targets for the data set were constructed by the RV method, which is time-consuming, only a fraction of the data can be used for training, ending up with a data set of 16 349 samples in the complete data set used in three different variants. The total numbers of the samples per data set can be found in Table 2 with the corresponding splitting of the data set into training and validation data sets.

Table 2 The number of data samples contained in each data set used for training and validation of YOLOv4

Plate Line status	Train	Validation	Total number
up	5 994	958	6 963
down	8 428	969	9 386
both	14 422	1 927	16 349

D. YOLO

First, object detection is employed to detect wake vortices in the overall LiDAR scans, in order to tackle the problem of detecting different numbers of wake vortices in different LiDAR scans. The name YOLO is an abbreviation for "You Only Look Ones", which reflects the nature of a one-stage detector. We use the original YOLOv4[2] written in C/C++ [8].

1. Network architecture

Like all object detectors, YOLOv4 consists of three parts: the backbone, the neck, and the head (see Fig. 3 and Section II.B). As backbone YOLOv4 uses *CSPDarknet-53* [10]. In the neck, spatial pyramid pooling (SPP) [11] and

[2] `https://github.com/AlexeyAB/darknet`

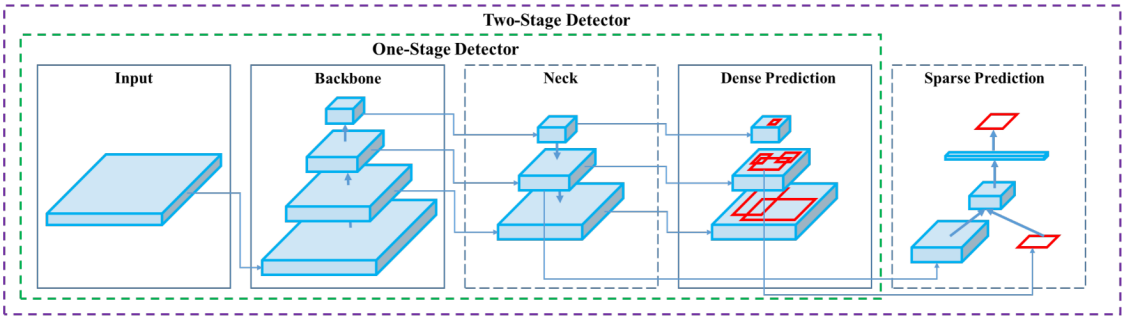


Fig. 3 Sketch of general object detectors (taken from [8]).

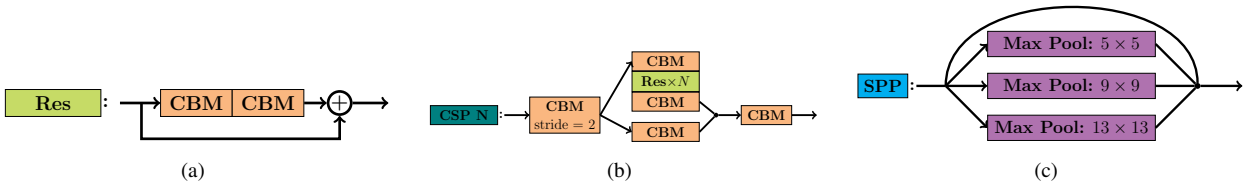


Fig. 4 The basic building blocks of the YOLOv4 Network. (a) Residual Block with CBM blocks. (b) CSP block with N residual blocks. (c) SPP block.

path-aggregation network (PAN) [12] are used. For the detection head, YOLOv3 [13] is used. The basic building blocks of YOLOv4 are illustrated in Figure 4.

CSPDarknet-53 is an update to Darknet-53 [13] incorporating so-called *cross-stage partial* (CSP) connections. A CSP block contains convolutional layers with batch normalization and Mish as an activation function, thus abbreviated with CBM (Convolutional layer with Batch normalization and Mish activating function). The first CBM layer in a CSP block uses a stride of two to downsample the input. After that, the output is copied and fed through another CBM layer, N residual blocks, illustrated in Figure 4 (a), and another CBM layer before being concatenated with the second copy which was only fed through a CBM layer. The concatenation is then again fed through a CBM layer. An illustration of that process can be found in Figure 4.

The PAN block combines features extracted at lower layers with features at higher layers. It uses bottom-up paths to make low-layer information easier to propagate [12]. In Figure 5, PAN is marked by the red connections. The difference between the PAN YOLOv4 uses to the originally proposed PAN is the usage of the concatenation of feature maps instead of addition [8].

The SPP block added after the CSPDarknet-53 backbone is used to significantly increase the network's receptive field and separate out the most significant context features [8]. This is accomplished using three different max pooling layers, each with a different pooling size, namely 5×5, 9×9, and 13×13. The SPP block is illustrated in Figure 4 (c).

Instead of the Mish activation function, YOLOv4 uses LReLU (leaky rectified linear unit) in the neck and the head of the network. After the convolution, batch normalization is also performed in the neck and head. Those blocks are abbreviated by CBL.

The detection head used is based on the anchor box idea from YOLOv3 [8, 13]. The functionality of the detection mechanism will be explained in more detail in the next section. The entire network of YOLOv4 can be seen in Figure 5.

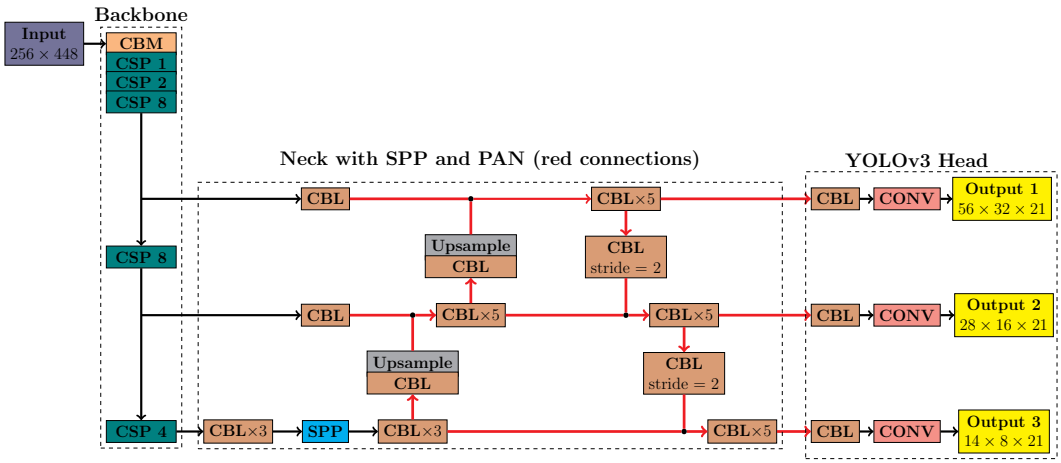
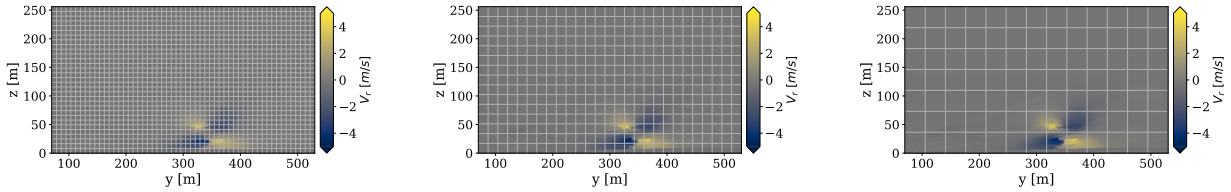


Fig. 5 An illustration of the architecture of the YOLOv4 network. When the paths split up, the output is copied. When two paths join, the dot represents the concatenation of the inputs.



(a) (b) (c)

Fig. 6 Three different grids used in the detection head of YOLOv3. (a) Fine grid: 56×32**. (b) Medium grid:** 28×16**. (c) Coarse grid:** 14×8**.**

YOLOv3 Head

As detection head YOLOv4 uses the YOLOv3 detection head [8]. The prediction in the YOLOv3 detection head is made at three different scales, such that we have three outputs [13]. Each detection head's prediction is based on a grid of different sizes such that objects of different sizes can be detected. The grid dimensions can be found in Table 3 and a LiDAR scan with the corresponding grids in Figure 6.

To predict the position and dimension of a bounding box, YOLOv3 uses *anchor box priors* [13] with width $p_w \in \mathbb{N}$ and height $p_h \in \mathbb{N}$. At each scale, YOLOv3 uses three anchor box priors, such that a total of nine anchor box priors are used. Different object shapes can be represented by using different anchor box priors. For each anchor box and grid cell, an offset, as well as a class and *objectness*, is predicted. In our case, there are two classes: port and starboard. The objectness predicts the intersection over union (IoU) (7) of the ground-truth and the proposed box [14]. A metric to measure how much a predicted box and a ground-truth box match IoU is defined as follows.

Definition II.2 (Intersection over Union [15])**.** Let $\mathbf{B}$ be a bounding box defined by (x, y, w, h) and $\mathbf{B}_{\text{gt}}$ be another bounding box defined by $(x_{\text{gt}}, y_{\text{gt}}, w_{\text{gt}}, h_{\text{gt}})$, corresponding to Definition II.1. The intersection over union (IoU) of $\mathbf{B}$ and $\mathbf{B}_{\text{gt}}$ is then defined by

$$\text{IoU}(\mathbf{B}, \mathbf{B}_{\text{gt}}) \coloneqq \frac{|\mathbf{B} \cap \mathbf{B}_{\text{gt}}|}{|\mathbf{B} \cup \mathbf{B}_{\text{gt}}|}. \tag{7}$$

The operation $|\mathbf{B}|$ corresponds to the cardinality of the finite set $\mathbf{B}$, which is understood as the number of elements a set contains. With IoU and the probability of an object existing in the predicted bounding box $\mathbf{B}$ objectness is defined by[16]

$$C(\mathbf{B}) \coloneqq P(\text{object})\text{IoU}(\mathbf{B}, \mathbf{B}_{\text{gt}}). \tag{8}$$

The offset prediction of YOLOv3 is made by predicting coordinates t_x, t_y, t_w, t_h representing the center, width, and height offset, respectively. Furthermore, the objectness t_o and a probability for each class t_{c_i} is activated by the logistic function σ to represent a probability. The bounding box is then calculated with respect to the center of a grid cell, defined by the offset of c_x, c_y from the origin at the top left corner, by

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \\ C(B) &= \sigma(t_o) \\ P(\text{class}_i) &= \sigma(t_{\text{class}_i})\ i \in \{1, 2\}. \end{aligned} \tag{9}$$

At the testing time, the objectness is multiplied by a conditional class probability $P(\text{class}_i|\text{object})$ to give a class-specific confidence score. In that way, the probability of class i appearing in the bounding box and the quality of bounding box fitting is encoded [16]. Figure 7 is a sketch of the positioning and scaling of an anchor box to a predicted bounding box.

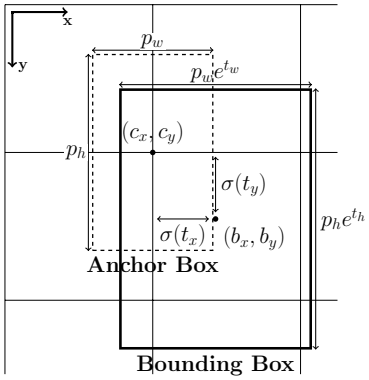


Fig. 7 The process of transforming bounding box predictions from an anchor box prior to a predicted bounding box. (Inspired by [14])

With that knowledge, we can calculate the output dimension of each detection head. Let the detection head divide the image into a $S_1 \times S_2$ grid and let each grid cell predict B_{num} bounding boxes. Let furthermore be C_{num} be the number of classes to predict. The dimension of the detection head output can then be calculated by

$$S_1 \times S_2 \times [B_{\text{num}}(\underbrace{4}_{\text{bounding box coordinates}} + \underbrace{1}_{\text{confidence score}} + C_{\text{num}})]. \tag{10}$$

In our case, the detection heads are after the 139th layer, the 150th layer, and the 161st layer. Each detection head uses three anchor boxes, i.e., $B_{\text{num}} = 3$. Recalling that we aim to distinguish between the port and starboard vortices, two classes are used, i.e., $C_{\text{num}} = 2$. In Table 3, the output dimensions for each detection head are calculated according to equation (10).

Table 3 The dimension of the output tensors from the three different detection heads and the corresponding anchor boxes.

Detection Head Layer	Output Tensor Dimensions	Anchor Box Dimensions
139	$56 \times 32 \times 21$	12×16, 19×36, 40×28
150	$28 \times 16 \times 21$	36×75, 76×55, 72×146
161	$14 \times 8 \times 21$	142×110, 192×243, 459×401

Since each grid cell predicts three bounding boxes, some large objects or objects near the border of multiple cells can be well localized by multiple cells [16]. The tool to mitigate that problem is non-maximum suppression (NMS). YOLOv4 uses greedy NMS [8]. The idea behind greedy NMS is to select a bounding box with the highest objectness and suppress all other bounding boxes that have an IoU above a given threshold $T_{\text{nms}} \in [0, 1]$[17]. To assign a predicted bounding box to the correct ground-truth box, an IoU threshold of 0.231 is used [8].

2. YOLOv4 Loss Function and Accuracy Measurement

The loss function for all versions of YOLO can be split into three parts. The first part evaluates the bounding box prediction, the second part evaluates the objectness prediction, and the last part evaluates the class predictions. The loss function used by YOLOv4 is an updated version of the loss function from YOLOv3. For the bounding box regression, instead of the MSE (Mean Square Error) loss, complete IoU (CIoU) [15] loss is used [8]. The CIoU loss not only considers the MSE between the bounding box predictions but considers the overlapping area, the distance between center points, and the aspect ratio [15].

Definition II.3 (CIoU Loss [15])**.** Let $\mathbf{B}_{\text{p}} = B(\mathbf{b}, w_{\text{p}}, h_{\text{p}})$ be a predicted bounding box and $\mathbf{B}_{\text{gt}} = B(\mathbf{b}_{\text{gt}}, w_{\text{gt}}, h_{\text{gt}})$ the ground-truth bounding box. Furthermore, let the center of the bounding boxes be $\mathbf{b}$ and $\mathbf{b}_{\text{gt}}$, respectively. The

consistency of the aspect ratio is measured by

$$v \coloneqq \frac{4}{\pi^2}\left(\arctan\frac{w_{\mathrm{gt}}}{h_{\mathrm{h_gt}}} - \arctan\frac{w_{\mathrm{p}}}{h_{\mathrm{h_p}}}\right)^2. \tag{11}$$

The CIoU loss is then defined, with a trade-off parameter $\alpha \in \mathbb{R}_{\geq 0}$ and the diagonal $c \in \mathbb{R}_{\geq 0}$ of the smallest enclosing box covering $\mathbf{B}_{\mathrm{p}}$ and $\mathbf{B}_{\mathrm{gt}}$, by

$$\mathcal{L}_{\mathrm{CIoU}}(\mathbf{B}, \mathbf{B}_{\mathrm{gt}}) \coloneqq 1 - \mathrm{IoU}(\mathbf{B}, \mathbf{B}_{\mathrm{gt}}) + \frac{\|\mathbf{b} - \mathbf{b}_{\mathrm{gt}}\|^2}{c^2} + \alpha v. \tag{12}$$

The objectness is evaluated with binary-cross entropy loss [18] split up into the cases of an object being present in a grid cell for each bounding box and the case of no object being present. This is done to force the objectness to be zero if no object is present. Finally, the classification task is also measured with binary-cross entropy loss [18].

The training of YOLOv4 was performed on the TUHH (Technical University of Hamburg) cluster using an NVIDIA A100-80 GPU. The training of YOLOv4 took aproximately 5 h for the proxy data set and 7 h for the LiDAR data set.

To evaluate object detectors, we count the true positive (TP), false positive (FP), and false negative (FN) predictions. In classification problems *recall* and *precision* are used as a measure of success. Recall measures the ratio of TP predictions out of all positive samples, i.e., a low recall hints at many FN predictions [19, p. 87]. Precision measures the ratio of TP predictions out of all samples predicted as positive, i.e., a low precision hints at many FP predictions [19, p. 87]. The formulas to calculate precision and recall are [19, p. 86]

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}. \tag{13}$$

To assign a class to a bounding box, YOLO uses a class probability, i.e., for each class, a probability of that class being present in a bounding box is predicted. To finally label that bounding box with a class, a threshold $T \in [0, 1]$ is used at which probability a class prediction is used. To find the best threshold and the best compromise between high recall and high precision, a precision-recall curve is used . The precision-recall curve plots the precision against the recall for all thresholds $T \in [0, 1]$ [19]. The average precision (AP) for a given class now is defined as the area under the precision-recall curve [20]. For each class we want to detect, the AP is calculated. The mean average precision (mAP) is the mean of the AP per class and the metric used to evaluate YOLOv4. The mAP can be calculated for different IoU thresholds at which a prediction is considered positive also to reflect the quality of the bounding box.

3. Data Preprocessing

Before we can use the LiDAR data sets for the training of YOLOv4, the labels and scans have to be converted to match the architecture of YOLOv4 and the required input data type.

As a label, YOLOv4 needs the center position of a vortex, a width $w_{\mathrm{B}} \in \mathbb{R}_{\geq 0}$, and height $h_{\mathrm{B}} \in \mathbb{R}_{\geq 0}$ which we both set to be the initial vortex spacing $b_0 = \pi/4B$, with the aircraft's wing span B. We employ b_0 as the width and height of the bounding box as label. That choice ensures only one vortex center being present in each bounding box, even shortly after aircraft passage, contributing to our goal of individual vortex characterization. The underlying raw LiDAR data is in the form of polar coordinates, see Fig. 2 (a). LiDAR scans of polar coordinates displayed in a grid visually distort the vortices. Hence scans in polar coordinates are not suited for the usage of b_0 as bounding box dimensions. We therefore only focus on LiDAR scans in the form of Cartesian coordinates, see Fig. 2 (a).

E. YOLOv4 Training

Since using a pre-trained version of Faster R-CNN (Region Based Convolutional Neural Networks) on the Microsoft Common Objects in Context data set (COCO) [21] gave promising results [22], we use a version of YOLOv4 pre-trained on the COCO data set as well. COCO data set consists of 328 000 images containing 2 500 000 labeled instances of common objects from 91 classes [21].

The data set can be categorized into three different data set groups. One data set group with only scans having the plate lines used, one data set group without using plate lines and the combined data set group. To evaluate which training works best for YOLOv4, we train it with those three data set groups separately. The three resulting YOLOv4 models generated by training with each training data set group, are validated with each validation data set group. We employ the mAP@50 as accuracy metric, as it is the standard choice for YOLOv4.

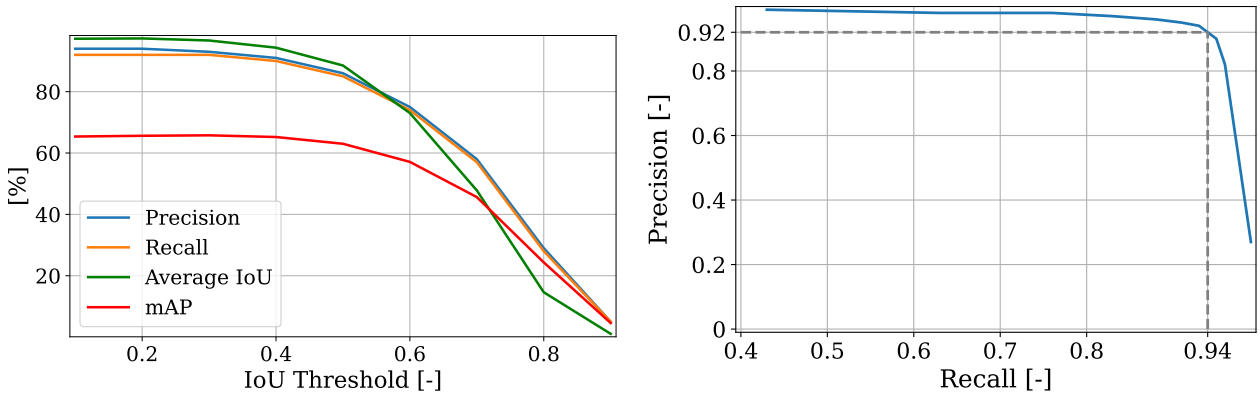


Fig. 8 (a) A graph of the precision and recall, as well as a graph of the average IoU and the mAP with respect to the IoU threshold used. (b) Precision-Recall curve with the optimal precision-recall pair marked.

The mAP for all different scenarios can be found in Table 4. The YOLOv4 model trained with the combined data set group - containing both plate line scenarios - has the best mAP for all validation data sets. We thus continue improving that model. In the further course of this work, we only consider the data set group containing both plate line scenarios for training and validation. For further training, we use YOLOv4 pre-trained on COCO as it was also utilized by [23].

Table 4 The mAP of YOLOv4 for the three different validation data sets after training with the respective training data sets. The highest mAP for each validation data set is marked in green.

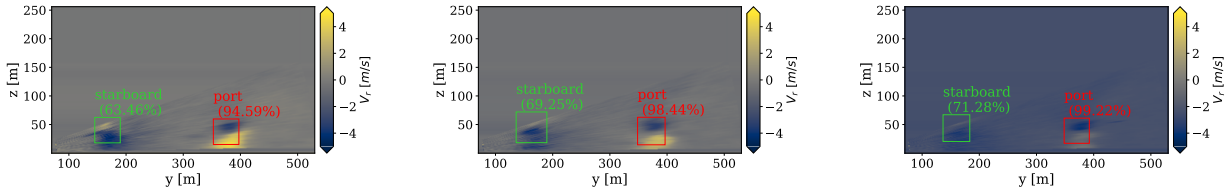
Pre-trained on COCO			
Training Data Set \ **Validation Data Set**	**Both**	**Down**	**Up**
Both	87.96%	86.79%	89.15%
Down	84.61%	85.55%	83.96%
Up	83.47%	84.22%	82.92%

1. Detection Parameter Setting

Since we use the YOLOv4 bounding box predictions as input for the regression network, we have to investigate the parameters used for detection. In particular, the confidence threshold used for detection has to be considered. We use the regression CNN to enhance the vortex center prediction. Hence a lower IoU can be considered for validation. So far, we have always used the mAP for an IoU threshold of 0.5 to evaluate our model. In Figure 8 (a), precision, recall, mAP, and average IoU for different values of IoU thresholds can be seen. Since the precision and recall are the highest at an IoU threshold of 0.25, we evaluate this case further for different confidence thresholds. In Figure 8 (b), the Precision-Recall curve is plotted for an IoU threshold of 0.25. The best point is marked at a precision of 0.92 and a recall of 0.94, with the confidence threshold being $T_c = 0.25$.

If we favor recall over precision, a confidence threshold of 0.2 would be the choice. Given that a lower recall explains increasing numbers of TP predictions and decreasing numbers of FN predictions, more vortices are captured. In the case of wake vortex prediction, the conservative choice, of capturing more vortices, is preferred. Therefore we apply a confidence threshold of 0.2 to create the bounding box predictions that are fed into the regression net.

Considering the RV method creating the ground-truth labels also has inaccuracies, we assume the confidence threshold of 0.2 to be the better choice and continue to use it for detection. Figure 9 shows an example of wrong YOLOv4 prediction according to the RV label. The first scan is predicted correctly according to the RV label. In the two consecutive scans, the RV method no longer detected the starboard vortex, but YOLOv4 has. As previously discussed, the vortex trajectory is a hyperbola without crosswind, but with a crosswind, the upwind vortex can be expected to stall



(a) (b) (c)

Fig. 9 Three consecutive LiDAR scans from the same overflight with YOLOv4 bounding box predictions. (a) Correctly predicted bounding box according to RV method. (b)-(c) False detection of starboard vortex according to RV method.

over the runway [24]. This is the case seen in Figure 9. Hence we assume the YOLOv4 prediction to be correct and that the RV method could not detect the starboard vortex. Therefore, leads to the assumption that the actual mAP, precision, and recall are higher than evaluated as we validate using labels generated by the RV method.

2. Accuracy

To evaluate the accuracy of the YOLOv4 network with a confidence threshold of 0.2 even further, we assume that the center of the bounding box matches the center of the predicted vortex. The rationale is that the vortex center is used for the bounding box center label. We use the validation data set for all those tests containing both plate line scenarios. After evaluating the validation data set, we find the mean confidence to be 84.54% and the median confidence to be 91.63%. With that result, we can confirm that the bounding box predictions are of good quality even with a low confidence threshold. To evaluate the accuracy of the center prediction from YOLOv4, we use the Euclidean distance of the bounding box center to the vortex center label, i.e., MADE (mean absolute distance error) Eq. (15). Furthermore, we separately evaluate the MAE (mean absolute error) Eq. (14) in the y- and z-directions to understand which coordinate prediction is more accurate. The results can be found in Table 5. We can see that the accuracy of the z-coordinate predictions is significantly higher. This can be explained by the larger velocity gradient present in the z-direction, caused by the perpendicular measuring of radial velocities only measuring the velocities in the direction of the LiDAR's LOS. Based on these values, we can evaluate the enhancement of vortex center predictions by the following regression CNN.

Table 5 MADE as well as MAE for y- and z-coordinates evaluated separately for each plate line scenario.

Plate Line Scenario	MADE	MAE (y)	MAE (z)
Both	6.26 m	5.86 m	1.41 m
Up	6.54 m	6.18 m	1.43 m
Down	5.96 m	5.52 m	1.39 m

The difference between the MADE with plate lines and without is 0.58 m, the MAE difference in y-coordinates is 0.66 m, and for z-coordinates, it is 0.04 m. Those differences are negligible, normalized by the minimal initial vortex separation $b_0 = 26.8$ m for aircraft A320 and A20N giving a percentage error. We end up with a rounded percentage error of 0.02 for the MADE and the MAE in y-coordinates and an even lower percentage error for the z-coordinates.

To further evaluate the quality of the vortex center prediction, we compare each prediction's absolute distance error(ADE) with the circulation strength target Γ_t of the respective vortex in Figure 10. We can see that YOLOv4 is more accurate at predicting the vortex center of stronger vortices. Those vortices, in most cases, belong to aircraft of the heavy category.

F. Regression Network

The regression network is employed after the YOLOv4 bounding box prediction on the predicted LiDAR scan sections, where vortices can be found. The task is to enhance the localization for each vortex.

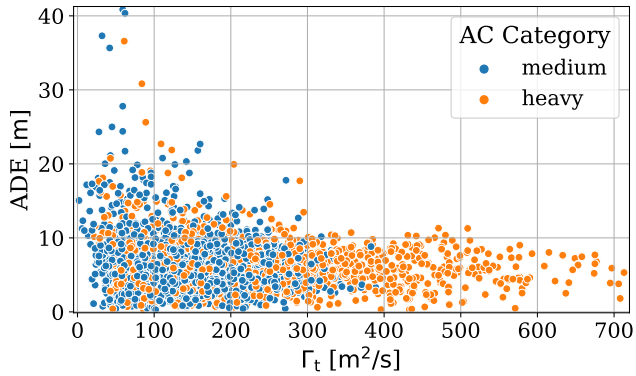


Fig. 10 Relationship between the ADE of the YOLOv4 prediction and circulation Γ_t of the respective vortex.

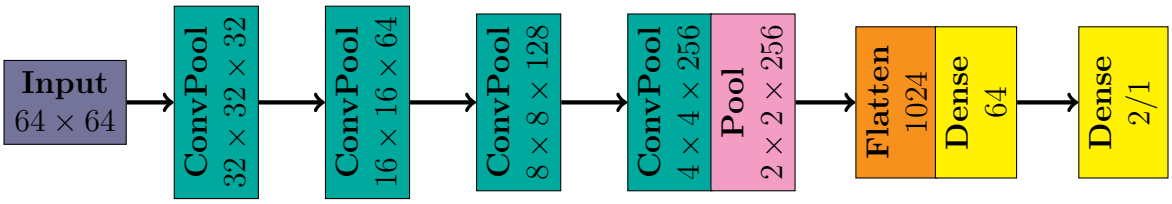


Fig. 11 The CNN used for the regression tasks with the output dimensions of each block. The final output dimension depends on whether the vortex center or vortex circulation strength is predicted.

1. Network Architecture

As base architecture, we use the CNN from [25], given that it achieved promising results on complete LiDAR scans and is assumed to perform even better on single vortices. The CNN consists of blocks of convolutional layers followed by max pooling layers, so-called *ConvPool* blocks. Four of such blocks are used, with the convolutional layers having a filter size of 3×3 and the pooling layers having a filter size of 2×2 and a stride of 2. The first layer uses 32 filters which number is subsequently doubled [25]. A graph of the CNN is depicted in Figure 11.

2. Training and Accuracy Measurement

We use ADAM [26] as an optimizer and MSE as a loss function to train the CNN. To avoid overfitting we include an early stopping mechanism to stop training if no improvement of the validation loss is seen for 30 epochs.

The machine-learning package we employ for implementing CNN regression is Keras [27]. Keras is a high-level API (Application Programming Interface) based on Tensorflow [28]. We use Keras version 2.9.0 and Tensorflow version 2.9.1., which were the latest versions available at the time of starting the work on this manuscript. The training of the regression networks was performed on the HOREKA cluster using an NVIDIA A100-40 GPU and took, on average, 30 min for each network.

The first metric we use is the mean absolute error (MAE) defined for a set $\mathbf{D}$ of N prediction-label pairs $(\hat{\mathbf{y}}_i, \mathbf{y}_i)$ with $i \in \{1, 2, \ldots, N\}$ by

$$\mathrm{MAE}(\mathbf{D}) = \frac{1}{N} \sum_{i=1}^{N} |\hat{\mathbf{y}}_i - \mathbf{y}_i|. \tag{14}$$

The MAE can be calculated for vectors using the 1-norm or for scalars with the absolute value. In the case of vortex center prediction, the MAE can be considered for the two coordinates separately to get a feeling for which coordinate prediction is more precise. If we replace the absolute error in the sum of (14) with the 2-Norm, we obtain the mean absolute distance error (MADE) which can be used to evaluate the overall distance error. The MADE is defined by

$$\mathrm{MADE}(\mathbf{D}) = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|. \tag{15}$$

3. Data Preprocessing

As the predicted bounding boxes can vary in size, but the regression CNN needs a constant input dimension, we have to choose a sufficient dimension as input. The bounding box size depends mainly on the initial vortex separation b_0, given that we labeled the data based on that. The largest initial vortex separation of the data set is $b_0 = 62.7$ m (A380). Therefore, a maximal width and height of 64 m, including a safety factor, is assumed to be sufficient.

4. Localization

Since we do not use data assimilation (DA), the vortex center of a perfectly cropped vortex is always in the center of the input data. Hence the CNN only trains to predict that the vortex center coincides with the center of the input data for perfectly cropped vortices. We, therefore, can not get benchmark values on how a network would behave on perfect bounding box predictions.

The results of the CNN for localization can be found in Table 6. To evaluate the improvement in comparison to the YOLOv4 center prediction, we use Table 5. The CNN improved the overall MADE with the CNN by 3.18 m, which is approximately half the error in comparison to solely using YOLOv4. If we take a closer look at which improvement accounts for the decrease in the localization error, we see that the MAE for the y-coordinate was more than halved from 5.86 m to 2.29 m. The MAE of the z-coordinate increased in the CNN prediction by 0.16 m. Hence the MADE improvement is due to a better y-coordinate prediction. The slightly higher MAE in z-coordinates of 0.16 m for both cases is negligible as normalized by the minimal initial vortex separation $b_0 = 26.8$ m is only, rounded to three decimals, 0.006.

Table 6 MADE as well as MAE for y- and z-coordinates separately for each plate line scenario by the CNN regression.

Plate Line Scenario	MADE	MAE (y)	MAE (z)
Both	3.08 m	2.29 m	1.57 m
Up	3.10 m	2.30 m	1.58 m
Down	3.06 m	2.27 m	1.56 m

Again we can see that the prediction in the case of no plate line usage is insignificantly better. While for YOLOv4 center prediction, we had a MADE difference between the up and down cases of 0.58 m, we now have a difference of 0.04 m. Hence we can assume that the plate line scenario has no significant impact on the quality of the vortex center prediction. The relationship of the ADE and the circulation strength target Γ_t looks similar to the one of YOLOv4 (see Figure 10) and can be seen in Figure 12. The main difference is that we now have a lower mean, and we see less outliers.

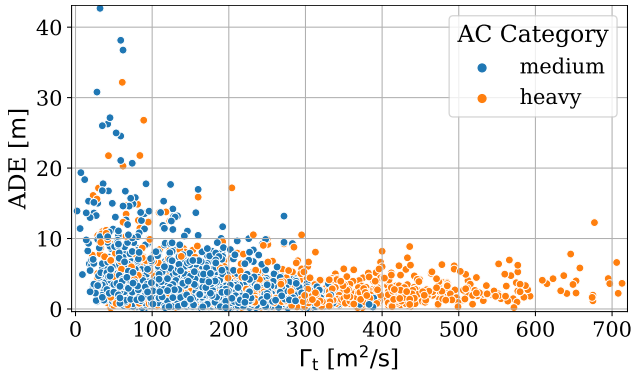


Fig. 12 Relationship between the ADE of the localization CNN and circulation Γ_t of the respective vortex.

G. Complete Prediction Pipeline

After training, the prediction time of the complete pipeline can be evaluated. An example of one LiDAR scan being processed can be found in Figure 13. Although the bounding box prediction is made on an image converted from a

LiDAR scan, we use the radial velocities and not the pixel intensities for illustration purposes. The original input LiDAR scan has previously been presented in Figure 2 and converted to Cartesian coordinates in Figure 13.

The pipeline includes the preprocessing and transforming of a LiDAR scan to an image. After that, the bounding box prediction by YOLOv4 is made, illustrated in Figure 13 (a). Based on the bounding boxes, the vortices get cropped from the original scan, Fig. 13 (b), 13 (c) and preprocessed according to previously explained feature engineering. Those vortices are fed into the circulation prediction CNN and the localization prediction CNN. The predictions of the CNNs can be found in Fig. 13 (d) and Fig. 13 (e), for the starboard and port vortex, respectively. The combination of the YOLOv4 prediction and the CNN prediction is depicted in Fig. 13 (f).

The average time this pipeline takes is 0.13 seconds on the HoreKa supercomputer with an NVIDIA A100-40 GPU and an Intel Xeon Platinum 8368 CPU. The CNN-only approach took 0.16 seconds for evaluation but was also performed without the usage of a GPU [25].

H. Comparison with the state-of-the-art

Since the labels for our data set were created with the RV method, we can only give a qualitative comparison. We must expect the accuracy of our approach to be, at most, the one used for labeling the data set. Given that the labels created with the RV method, might contain inaccuracies, and the data used also contains inaccuracies due to the nature of LiDAR measurements, a natural accuracy limit is given.

The traditional wake vortex characterization methods, the RV and VE methods, predict the vortex center in polar coordinates. Only the VE method provides an error in terms of absolute distance that we can compare our approach with. Those errors are only theoretical estimations [3, 29]. The median ADE for the VE method is 7.91 m [25]. YOLOv4 has a median ADE of 5.78 m, and the YOLOv4+CNN approach can reduce this to a median ADE of 2.24 m. As the RV method has an accuracy similar to the VE method for the vortex center estimation, we can compare both methods to our approach. The median ADE is of the same magnitude, so that we can say the prediction is of similar quality as the RV and VE method.

A comparison of the prediction of the herein presented approach - first using YOLOv4 and then using a CNN - with the results of the CNN from [25] can be seen in Table 7. The prediction accuracies were increased in all categories independent of both, the plate line scenario and the vortex class.

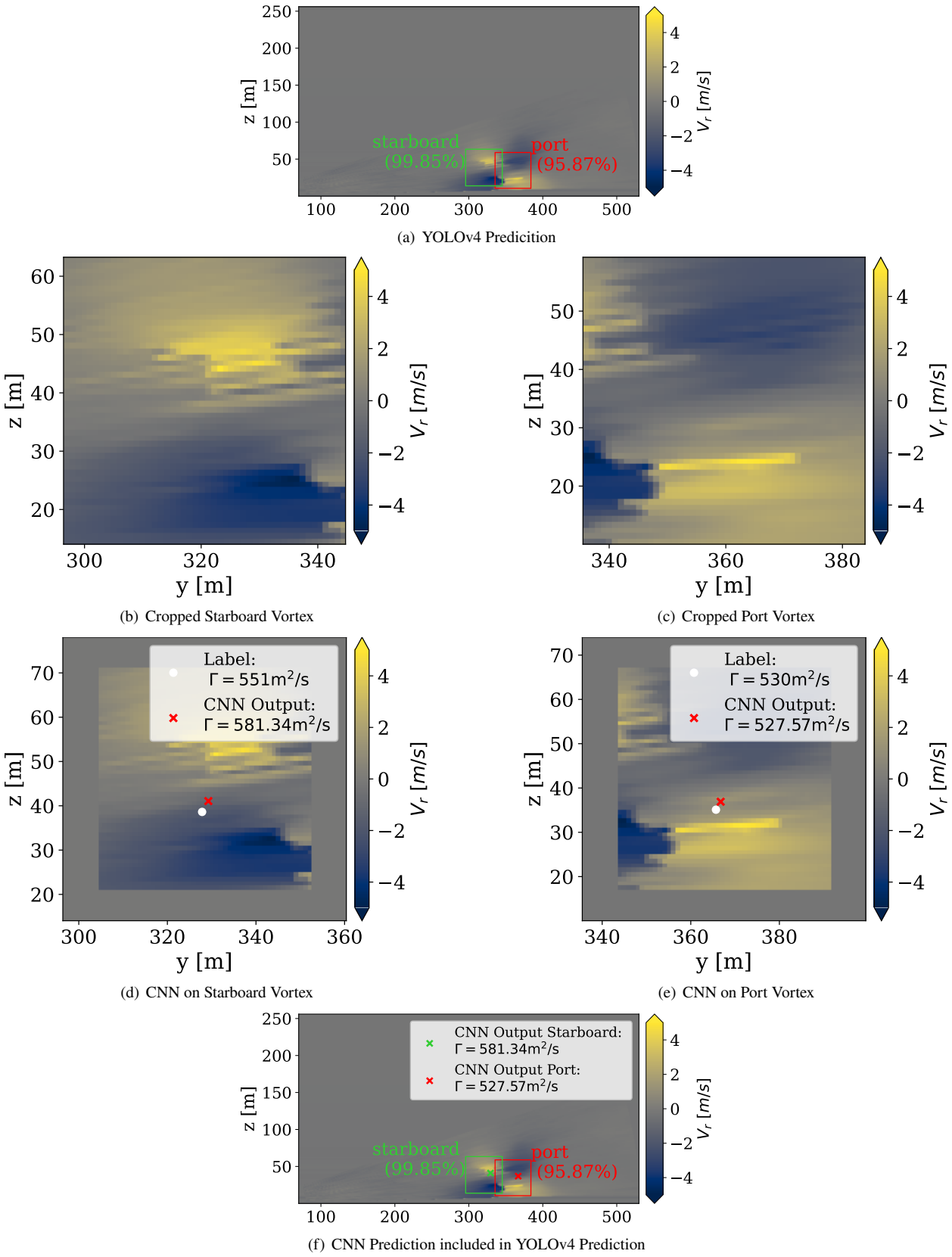
Table 7 Comparison of YOLOv4+CNN with the CNN approach from [25] for both plate line scenarios and vortex classes separately.

	Plate Lines Up		**Plate Lines Down**	
ANN	**Port MADE** [m]	**Starboard MADE** [m]	**Port MADE** [m]	**Starboard MADE** [m]
YOLOv4	7.59	5.30	6.64	4.94
YOLOv4+CNN	3.26	2.91	3.23	2.82
CNN-only [25]	46.90	46.70	21.89	22.70

The hazard detection can also be compared by using the precision value. The precision of the CNN-only approach was 88.6% [25]. With the new approach, we could increase the precision by 7.51% to gain a precision of 96.11%.

III. Vortex characterization with the Projection Method

After having identified the vortex position on LiDAR scans, the second task is to identify vortex characteristics, the vortex circulation and the vortex core radius. We will present a new direct method in this paper, which will be compared to the state-of-the-art RV method [3]. The method is called Projection Method (PM) since it operates with projections in an appropriate L_2 space. The PM method is a direct method, which - in contrast to the RV method - provides a best estimate for the core radius. It does not require training data and it is very fast. Further, the PM naturally comes with an accuracy measure for the estimated wake vortex parameters. First we will introduce the method in a general abstract L_2 framework, which easily is transferable to other problems, especially for identifying structures in LiDAR scans or other measurements where only parts of the full information are revealed. Later we will present applications to the presented data set.



(a) YOLOv4 Predicition

(b) Cropped Starboard Vortex

(c) Cropped Port Vortex

(d) CNN on Starboard Vortex

(e) CNN on Port Vortex

(f) CNN Prediction included in YOLOv4 Prediction

Fig. 13 An illustration of the complete prediction pipeline excluding the initial preprocessing step.

A. Mathematical framework

We start with a lidar scan represented in Cartesian coordinates, see Fig. 13 (a). As we have seen in the previous chapter, the prediction pipeline gives us a fairly precise estimation of the positions of wake vortices occurring in the scan together with a bounding box. Let us assume to have the cut-out vortex, see Fig. 13 (a). Let $n \times n = N$ be the size of the bounding box in terms of the pixels used in the interpolated scan. Then the cut out vortex can be represented by a point in $\mathbb{R}^N$. Let V_r be the image of a cut-out vortex, then $V_r \in \mathbb{R}^N$.

The vortex image measured by a LiDAR originates from a flow field, consisting of three velocity components in reality $\mathbf{u} = (\mathbf{u}, \mathbf{v}, \mathbf{w})^{\mathbf{T}}$. Hence, we can assume that the real underlying data is a point in the reality space $\mathbf{R} = \mathbb{R}^{\mathbf{3N}}$. The LiDAR measurement can be described as a mapping L from $\mathbf{R}$ to its subspace $\mathbf{U} = \mathbb{R}^{\mathbf{N}}$.

$$L : \mathbf{R} \longrightarrow \mathbf{U} \tag{16}$$

L is linear and it can be expressed as the composition of a projection P to the LiDAR beam unit vector $\mathbf{b}$ and a convolution C, which can be interpreted as a filter, with a range-gate weighting function (RWF) ρ :

$$L = C \circ P, \tag{17}$$

with

$$\tilde{V}_r = P\mathbf{u} = \mathbf{b} \cdot \mathbf{u} = \mathbf{u} \sin\gamma \cos\varphi + \mathbf{v} \cos\gamma \cos\varphi + \mathbf{w} \sin\varphi \tag{18}$$

and

$$V_r = C\tilde{V}_r(r_0) = \int_{-\infty}^{\infty} \rho(s)\tilde{V}_r(r_0 + s)ds, \tag{19}$$

where we interpret the mappings P and C on every coordinate in $\mathbf{R}$.

The task is to characterize vortices in terms of circulation 2 and core radius in the reality space bfR. Since the scans only provide a projection of the velocity field (followed by a filter), it is impossible to derive the circulation 2 directly, without any additional assumptions. Further, the core radius is defined for rotationally symmetric vortices, while we can expect tilted and disturbed ones due to interactions with other fluid structures.

Therefore an assumption on the vortex structure has to be made. We make the following

Definition III.1 (Vortex model)**.** Let $b(\Gamma, r_c) : \mathbb{R}_+ \to \mathbb{R}_+$ be be a continuous Function, depending on two Parameters Γ and r_c, with the following properties

$$(i) \quad b(\Gamma, r_c)(r) \sim O(r) \qquad \text{for } r \to 0 \tag{20}$$

$$(ii) \quad b(\Gamma, r_c)(r) \sim O(1/r) \qquad \text{for } r \to \infty \tag{21}$$

$$(iii) \quad b(\Gamma, r_c)(r) = \Gamma \cdot b(1, r_c)(r) \qquad \text{b is linear in } \Gamma \tag{22}$$

$$(iv) \quad \max b(\Gamma, r_c)(r) = b(\Gamma, r_c)(r_c) \tag{23}$$

We call b a vortex model.

Then $b(\Gamma, r_c)$ defines a vortex $B(\Gamma, r_c)(v_r, v_\theta, v_z)$ in space in cylindrical coordinates (r, θ, z) with the velocity components (v_r, v_θ, v_z) by

$$v_r = 0, \quad v_\theta = b(\Gamma, r_c), \quad v_z = 0 \tag{24}$$

We define the vortex in the measurement space $\mathbf{U}$ to be

$$\mathbf{B}(\Gamma, r_c) = LB(\Gamma, r_c) \tag{25}$$

Examples for vortex models are the Lamb-Oseen vortex model b_{LO}, the Burnham-Hallock vortex model b_{BH} and the Rankine vortex model b_{RA} with:

$$b_{LO}(\Gamma, r_c) = \frac{\Gamma}{2\pi r}\left(1 - e^{-1.2526\left(\frac{r}{r_c}\right)^2}\right) \tag{26}$$

$$b_{BH}(\Gamma, r_c) = \frac{\Gamma}{2\pi r}\frac{r^2}{r^2 + r_c^2} \tag{27}$$

$$b_{RA}(\Gamma, r_c) = \frac{\Gamma}{2\pi}\begin{cases} r/r_c^2 & r \le r_c, \\ 1/r & r > r_c \end{cases}. \tag{28}$$

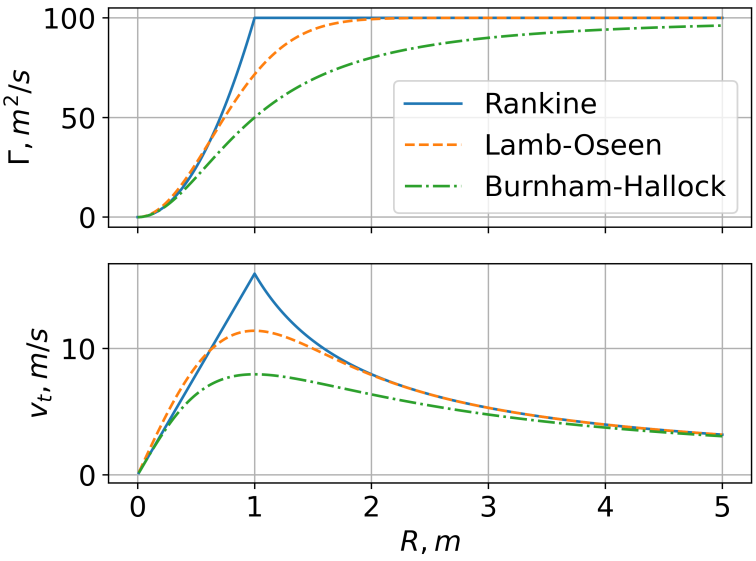


Fig. 14 Circulation distribution $\Gamma(r)$ and radial velocity distribution $V_\theta(r)$ for $\Gamma = 100$ and $r_c = 1$ m, for three different vortex models.

The circulation in a disc with radius r of a vortex associated with a vortex model can be easily computed with

$$\Gamma(r) = \oint_{\partial A} \vec{u} \cdot d\vec{s} = 2\pi r V_\theta(r). \tag{29}$$

This yields

$$\Gamma_{LO}(r) = \Gamma\left(1 - e^{-1.2526\left(\frac{r}{r_c}\right)^2}\right) \tag{30}$$

$$\Gamma_{BH}(r) = \Gamma\frac{r^2}{r^2 + r_c^2} \tag{31}$$

$$\Gamma_{RA}(r) = \Gamma\begin{cases} r^2/r_c^2 & r \le r_c, \\ 1 & r > r_c \end{cases}. \tag{32}$$

Figure 14 depicts the circulation and radial velocity of the different vortex models.

Definition III.2 (Model manifold)**.** We define the model manifold $\mathcal{M}_b \subset \mathbf{U}$, associated with a vortex model b to be the set of points

$$\mathcal{M}_b = \{\mathbf{B}(\Gamma, r_c), \quad \Gamma, r_c \in \mathbb{R}_+\} \tag{33}$$

Since $\mathcal{M}_b$ is a two dimensional differentiable manifold, as it can be locally parameterized by Γ and r_c, we also call it the model surface.

B. Method of Projections

With the Method of Projections we present an efficient way to calculate the circulation and the core radius of a vortex in a Lidar measurement assuming a certain vortex model, that represents the real vortex. This method also provides a measure of how good the assumed vortex model represents the measured wind field.

Let us assume, that the identified vortex in the measurement v_r has the shape of a vortex associated with a vortex model with the fixed radius r_c. It therefore can be written in the form

$$V_r = \mathbf{B}(\Gamma, r_c) \tag{34}$$

Now, if we project V_r to $\mathbf{B}(1, r_c)$ we get with Eq. 22:

$$(V_r, \mathbf{B}(1, r_c)) = (\mathbf{B}(\Gamma, r_c), \mathbf{B}(1, r_c)) = \Gamma\,(\mathbf{B}(1, r_c), \mathbf{B}(1, r_c))\,. \tag{35}$$

Hence,

$$\Gamma = (V_r, \mathbf{B}(1, r_c))\,/||\mathbf{B}(1, r_c)||^2. \tag{36}$$

Equation 36 provides the Circulation of a vortex, assuming a certain core radius.

To calculate the core radius r_c, we maximize the projection of V_r to the normalized vectors $\mathbf{B}(1, r)/||\mathbf{B}(1, r)||$. Hence we get the following

Algorithm III.1 (Projection Method)**.** To calculate the core radius and circulation of a vector field V_r assuming a certain vortex model we perform the following two steps:

$$\textbf{Step 1:} \quad \max_r\,(V_r, \mathbf{B}(1, r)/||\mathbf{B}(1, r)||) \Longrightarrow r_c \tag{37}$$

$$\textbf{Step 2:} \quad \Gamma = (V_r, \mathbf{B}(1, r_c))\,/||\mathbf{B}(1, r_c)||^2 \tag{38}$$

The presented algorithm is essentially a projection of a point V_r, which has to be evaluated to the model surface $\mathcal{M}_b$ for a certain vortex model b. It is clear from the definition, that given a certain vortex model, the projection $\mathbf{B}(\Gamma_0, r_c)$ will be the best approximation of V_r. A a measure for the accuracy of the vortex model will be the distance of V_r to the model surface $\mathcal{M}_b$. We make the following

Definition III.3 (Model accuracy)**.** The normalized distance d of a point $V_r \in \mathbf{U}$ to the model surface $\mathcal{M}_b$, which is given by the distance of V_r to the projection to $\mathcal{M}_b$

$$d = ||\mathbf{B}(\Gamma_0, r_c) - V_r||/N \tag{39}$$

is called model accuracy.

Since it may happen, that different sizes of the bounding boxes are involved we normalize the distance by the number of points of the cut out vortex image.

C. Application of the Projection Method

In the following sections we would like to present results of the Projection Method with increasing complexity in each section.

1. Employed Vortex models

We apply the Projection Method to the processed data set with different vortex models. The different vortex models are listed in the first row of Table 8.

Simplified Models

For the first three we assume that the convolution C contributing to the LiDAR operator L is the identity, $C = id$ hence $L = P$ is a pure projection. This assumption is mostly used in methods for vortex circulation evaluation. Here we analyse three different vortex models b_{BH}, b_{LO} and b_{RA}, which we call *B.-H. simple*, *L.-O. simple* and *Rankine simple*. The projection stencils for the *B.-H. simple* model can seen in Fig. 15 (a) - (c). For a small core radius the high radial velocities are concentrated in the center of the image. With increasing core radius the areas of high velocities get larger.

Models including LiDAR convolution

Further we investigate the vortex model including the full LiDAR convolution Eq. (19) in both steps of the algorithm Eq. (37) and (38). We call this model *B.-H. full*. The projection stencils for the *B.-H. full* model can seen in Fig. 15 (d) - (f). Since the convolution acts like a filter in the direction of the LiDAR beam, the vortex peaks are flattened in radial direction. Compared to the vortex radial velocity field V_r, see Fig. 13 (b) and (c), this seems to be potentially a better fit,

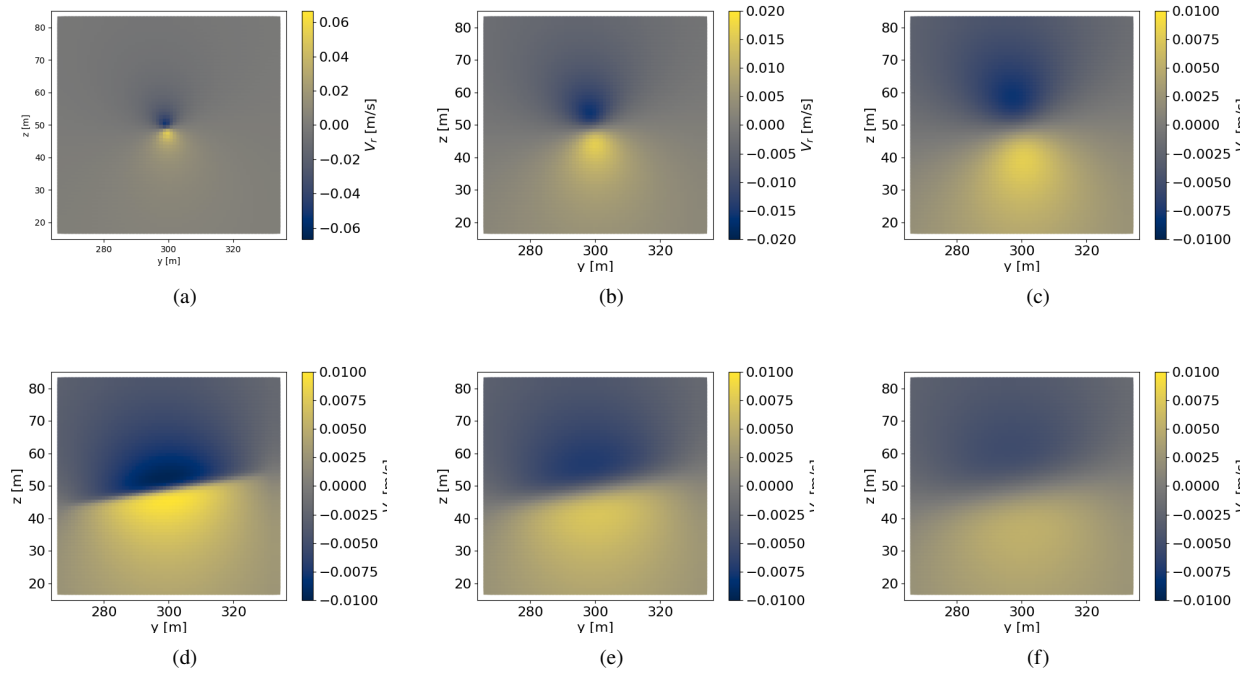


Fig. 15 Three Burnham-Hallock vortex unit templates for a simplified LiDAR (a) - (c) and a full LiDAR (d) - (f) for different core radii, $\mathbf{B}(1, 1.5)$ (a), (d), $\mathbf{B}(1, 5)$ (b), (e), $\mathbf{B}(1, 10)$ (c), (f) at a generic position of $(x, y) = (300\,\mathrm{m}, 50\,\mathrm{m})$.

than the simplified LiDAR model. The calculation of the convolution Eq. (19) is quite time consuming, which increases the computation time by a factor of 50.

Due to the time consumption of the *B.-H. full* model, we also add a *B.-H. partial* model, where only in step 2 of the PM algorithm, (38), the LiDAR convolution Eq. (19) is applied.

2. Performance Comparison

The results of the PM applied to the entire data set can be seen in Fig. 16. Here we plot the estimation of either the circulation of the core radius with one model against the other. Fig. 16 (a) - (c) compares the estimated circulation with the different vortex models.

First let us observe, that the results for circulation differ considerably, Fig. 16 (a). In almost all cases the circulation estimation with the *L.-O. simple* model is smaller than the estimation with the *B.-H. simple* model, on average, the circulation achieved using the Lamb-Oseen is about 71% of the circulation in the *B.-H. simple*. In case of the Rankine vortex model, the difference is even more pronounced. On average, the *Rankine* simple model yields circulation values of about 57% of the circulation found by the *B.-H. simple*. On the contrary, the effect of the LiDAR simulation is found to be small, see Fig. 16 (c). The dots scatter around the identity line. On average the *B.-H. full* yields circulation values of about 96% of the circulation values found by the *B.-H. simple* model.

Due to these large deviations, depending on the assumed vortex model, we have to investigate the accuracy of the fits. For that we introduced the l_2-distance d in Def. 39. The distribution of d is plotted in Fig. 17 (a) and (b) compared to the distance, the RV method would yield. While we observe that the PM method yield much better approximations of the flow field compared to the RV method in terms of distribution peak and spread, we see that the actual model is not important in the overall statistics. Even in the closeup Fig. 17 (b) the different models lead to minor differences.

From this we conclude that although the circulation estimations are crucially different, the approximation distance does not score one vortex above the other, at least if the entire data set is included. That means that in the case if models are used, the estimated circulation is highly dependant of the involved model. Thus it is inevitable to always refer to a certain vortex model when presenting a circulation estimation in LiDAR scans. It is further necessary to ensure that the same corresponding vortex model is used in subsequent investigations, like encounter simulations or flight simulators,

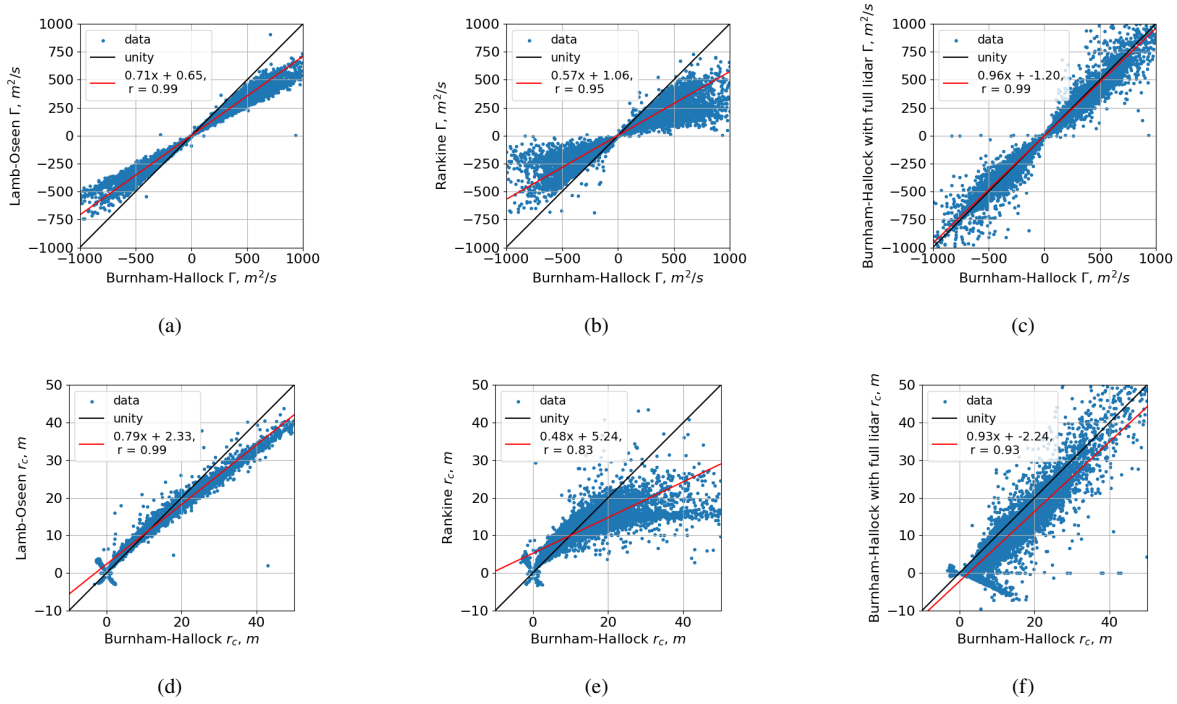


Fig. 16 Circulation (a) - (c) an core radius (d) - (f) estimation for different vortex models with linear fit.

as the reported circulation values may be invalid for other vortex models!

Evaluating the vortex core radius in Fig. 16 (d) - (f), we observe that the difference in the models is not linear. Therefore, the liner fit is somewhat misleading. Comparing *B.-H. simple* with *L.-O. simple* and *Rankine simple* for small core radii below 10 m the *B.-H. simple* is underestimating, while overestimating for core radii above 20 m. The *B.-H. full* model does not deviate too much yielding on average core radii of about 93%. This outcome was expected since the LiDAR convolution seems to enlarge the model vortex, cf. Fig. 15.

3. Application to data subsets

In this section we will analyse why the different models seem to perform equally well, and whether this depends on the data set. Figure 17 (a) and (b) indicates that large deviations from the vortex model like disturbances due to turbulence or high CNR mainly lead to the approximation error, such that the influence of the vortex model is negligible. To investigate this hypothesis we set up two filters of the analysed data.

Filter 1: Scans with plate lines erected are excluded. Since the plate lines lead to a accelerated wake vortex decay, we may assume, that the vortex is less coherent in that case.

Filter 2: Scans with plate lines erected and scans with a mean CNR value above the median are excluded. Due to different weather situations and a different amount of aerosols the LiDAR signal has changeable quality. With that filter we try to focus on the best quality scans.

Figure 17 (c) and (d) shows that an exclusion of the plate lines (filter 1) increases approximation accuracy, even to a point where the different vortex models split up. Note, that the performance of the Rankine vortex model is the worst, *B.-H. simple* and *L.-O. simple* perform similarly well and the LiDAR including models even better.

The effect of filter 2 can be seen in Fig. 17 (e) and (f). The accuracy is further increasing in terms of distribution peak as well as the spread of the vortex models. This proves that the LiDAR simulated vortex model performs better, which is masked by scans in turbulent environments or scans with a high CNR.

Table 8 lists the mean, median and standard deviation of the error distribution in case that filter 2 is applied. We observe a clear superiority of the *B.-H. full* model in terms of mean and median, while the standard deviation is comparable.

Figure 18 shows how the approximation accuracy distribution changes, when filter 1 and filter two are applied. Table 9 list the quantitative performance in terms of mean, median and standard deviation. Clearly the removal of

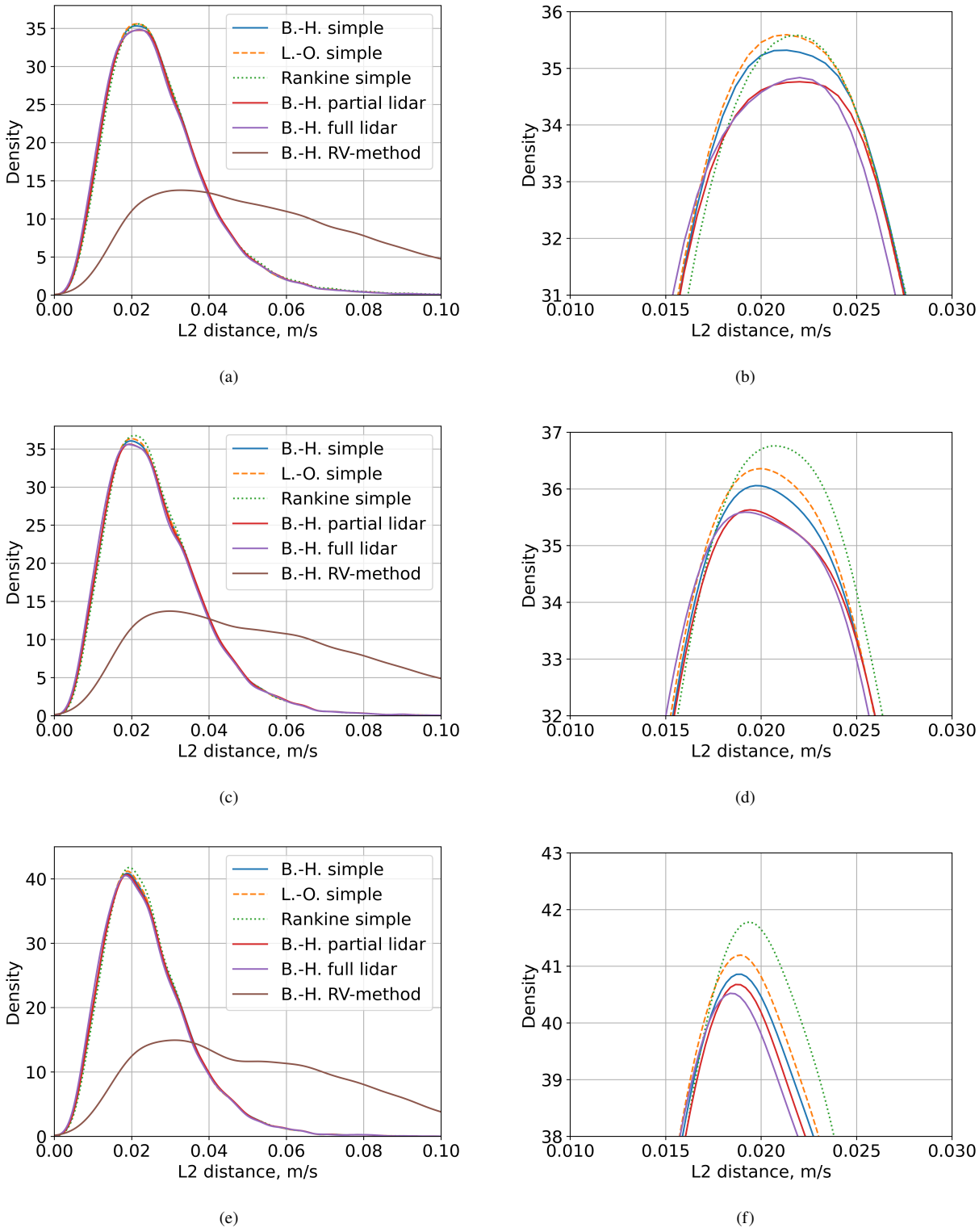


Fig. 17 Approximation accuracy for different models compared with the RV method. No filter on the data (a), plate lines excluded (c) and plate lines and high CNR values excluded (e). Close-up image on the right.

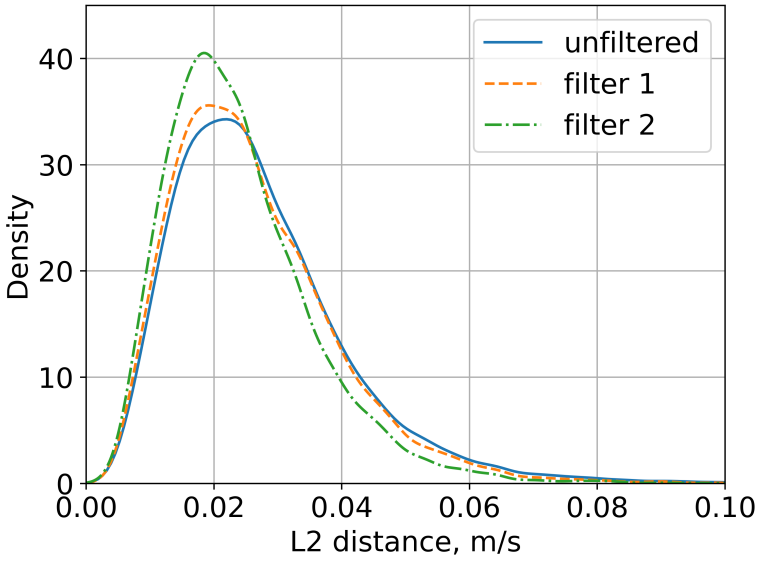


Fig. 18 Approximation accuracy for the Burnham-Hallock simple model with different filters applied.

turbulent and noisy scans lead to a reduction of the approximation distance and lower spread.

Another source of the approximation distance might be the influence of the other vortices in the scan as well as the influence of the image vortices at the ground, which will be investigated in the near future.

Table 8 Mean, median and standard deviation of the error distributions of the different vortex models as well as the reference case calculated with the RV method. Filter 2 applied.

	mean	median	std. deviation
B.-H. simple	0.0244	0.0225	0.0117
L.-O. simple	0.0244	0.0224	0.0116
Rankine simple	0.0246	0.0226	0.0116
B.-H. partial lidar	0.0244	0.0224	0.0117
B.-H. full	0.0240	0.0220	0.0117
RV-method	0.0531	0.0496	0.0274

IV. Conclusion

This work presented a new comprehensive strategy to track and to characterize vortices in LiDAR scans. We divided the task into two parts, first the determination of the vortex position with a two-stage neural network approach. Second, the characterization of the vortices in terms of circulation, core radius, and (partly) tangential velocity profile empoying the Projection Method.

We employed the Computer Vision tool YOLOv4 as the first stage for vortex detection and provided specific metrics to measure the accuracy. For the vortex position, a mean average precision of about 88% was achieved. As a second stage, a CNN regression network was applied, leading to a position detection accuracy of about 3 m which is actually the instrument resolution distance.

In the second part, we introduced a new calculus, how to apply models to retrieve properties and moments of the coherent structures of the underlying flow field. This framework naturally yields an algorithm that provides the moments of interest by projections of the field to model stencils. Therefore it outperforms other methods by orders of magnitude in computation speed. The calculus provides a metric for the approximation quality. We could show that the Projection

Table 9 Mean, median and standard deviation of the error distributions of *B.-H. simple* vortex model for different filters.

	mean	median	std. deviation
unfiltered	0.0275	0.0248	0.0156
filter 1	0.0262	0.0239	0.0130
filter 2	0.0240	0.0220	0.0117

Method increases the approximation quality more than 50%.

Further, we analysed the effects of different vortex models and concluded that while providing comparatively good approximation accuracy, the circulation estimates differ significantly. This is often masked by turbulence and noise in the scans, as we have shown applying different filters.

Finally, with coherent data we see a difference in the model performance. To make it even clearer the influence of the other vortices in the RHi scan as well as the influence of the image vortices at the ground will be investigated in the near future.

Acknowledgements

We greatly acknowledge the valuable ideas and contributions from Prof. Dr. Daniel Ruprecht from the Institute of Mathematics at the Hamburg University.

This project has received funding within the framework of the Single European Sky Air Traffic Management Research Joint Undertaking "Safely Optimized Runway Throughput" project (VLD3-W2 SORT) within the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 874520 as well as from the DLR, German Aerospace Center (DLR) project "Wetter und Disruptive Ereignisse."

References

[1] Robey, R., and Lundquist, J. K., "Behavior and mechanisms of Doppler wind lidar error in varying stability regimes," *Atmospheric Measurement Techniques*, Vol. 15, No. 15, 2022, pp. 4585–4622. https://doi.org/10.5194/amt-15-4585-2022, URL https://amt.copernicus.org/articles/15/4585/2022/.

[2] Wildmann, N., Gerz, T., and Lundquist, J. K., "Long-range Doppler lidar measurements of wind turbine wakes and their interaction with turbulent atmospheric boundary-layer flow at Perdigao 2017," *Journal of Physics: Conference Series*, Vol. 1618, No. 3, 2020, p. 032034. https://doi.org/10.1088/1742-6596/1618/3/032034, URL https://dx.doi.org/10.1088/1742-6596/1618/3/032034.

[3] Smalikho, I., Banakh, V., Holzäpfel, F., and Rahm, S., "Method of radial velocities for the estimation of aircraft wake vortex parameters from data measured by coherent Doppler lidar," *Opt. Express*, Vol. 23, No. 19, 2015, pp. A1194–A1207. https://doi.org/10.1364/OE.23.0A1194, URL https://opg.optica.org/oe/abstract.cfm?URI=oe-23-19-A1194.

[4] Holzäpfel, F., Stephan, A., Rotshteyn, G., Körner, S., Wildmann, N., Oswald, L., Gerz, T., Borek, G., Floh, A., Kern, C., Kerschbaum, M., Nossal, R., Schwarzenbacher, J., Strobel, M., Strauss, L., Weiß, C., Kauczok, S., Schiefer, C., Czekala, H., Maschwitz, G., and Smalikho, I., "Mitigating Wake Turbulence Risk During Final Approach via Plate Lines," *AIAA Journal*, Vol. 59, No. 11, 2021, pp. 4626–4641. https://doi.org/10.2514/1.J060025, URL https://doi.org/10.2514/1.J060025.

[5] Holzäpfel, F., Stephan, A., Rotshteyn, G., Körner, S., Wildmann, N., Oswald, L., Gerz, T., Borek, G., Floh, A., Kern, C., et al., "Mitigating wake turbulence risk during final approach via plate lines," *AIAA Journal*, Vol. 59, No. 11, 2021, pp. 4626–4641.

[6] Zhao, Z.-Q., Zheng, P., Xu, S.-T., and Wu, X., "Object Detection With Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 30, No. 11, 2019, pp. 3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865.

[7] Du, L., Zhang, R., and Wang, X., "Overview of two-stage object detection algorithms," *Journal of Physics: Conference Series*, Vol. 1544, No. 1, 2020, p. 012033. https://doi.org/10.1088/1742-6596/1544/1/012033, URL https://dx.doi.org/10.1088/1742-6596/1544/1/012033.

[8] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M., "YOLOv4: Optimal Speed and Accuracy of Object Detection," , 2020.

[9] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "ImageNet Classification with Deep Convolutional Neural Networks," *Commun. ACM*, Vol. 60, No. 6, 2017, p. 84–90. https://doi.org/10.1145/3065386, URL https://doi.org/10.1145/3065386.

[10] Wang, C.-Y., Mark Liao, H.-Y., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., and Yeh, I.-H., "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 1571–1580. https://doi.org/10.1109/CVPRW50498.2020.00203.

[11] He, K., Zhang, X., Ren, S., and Sun, J., "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 37, No. 9, 2015, pp. 1904–1916. https://doi.org/10.1109/TPAMI.2015.2389824.

[12] Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J., "Path Aggregation Network for Instance Segmentation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768. https://doi.org/10.1109/CVPR.2018.00913.

[13] Redmon, J., and Farhadi, A., "YOLOv3: An Incremental Improvement," , 2018. https://doi.org/10.48550/ARXIV.1804.02767.

[14] Redmon, J., and Farhadi, A., "YOLO9000: Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525. https://doi.org/10.1109/CVPR.2017.690.

[15] Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., and Ren, D., "Distance-IoU loss: Faster and better learning for bounding box regression," *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34, 2020, pp. 12993–13000. https://doi.org/10.1609/aaai.v34i07.6999.

[16] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. https://doi.org/10.1109/CVPR.2016.91.

[17] Bodla, N., Singh, B., Chellappa, R., and Davis, L. S., "Soft-NMS — Improving Object Detection with One Line of Code," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5562–5570. https://doi.org/10.1109/ICCV.2017.593.

[18] Chollet, F., *Deep learning with Python*, Manning Publications Company, Birmingham, 2017.

[19] Lindholm, A., Wahlström, N., Lindsten, F., and Schön, T. B., *Machine Learning - A First Course for Engineers and Scientists*, Cambridge University Press, 2022. URL https://smlbook.org.

[20] Everingham, M., Van Gool, L., Winn, J., and Zisserman, A., "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, Vol. 88, 2010. https://doi.org/10.1007/s11263-009-0275-4.

[21] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., "Microsoft coco: Common objects in context," *European conference on computer vision*, Springer, 2014, pp. 740–755. https://doi.org/10.1007/978-3-319-10602-1_48.

[22] Baranov, N., and Resnick, B., "Wake vortex detection by convolutional neural networks," *European Journal of Electrical Engineering and Computer Science (EEACS)*, Vol. 3, 2021, pp. 92–97.

[23] Weijun, P., Yingjie, D., Qiang, Z., Jiahao, T., and Jun, Z., "Deep Learning for Aircraft Wake Vortex Identification," *IOP Conference Series: Materials Science and Engineering*, Vol. 685, No. 1, 2019. https://doi.org/10.1088/1757-899x/685/1/012015.

[24] Stephan, A., Holzapfel, F., and Misaka, T., "Hybrid simulation of wake-vortex evolution during landing on flat terrain and with plate line," *International Journal of Heat and Fluid Flow*, Vol. 49, 2014, pp. 18–27.

[25] Wartha, N., Stephan, A., Holzäpfel, F., and Rotshteyn, G., "Characterizing aircraft wake vortex position and strength using LiDAR measurements processed with artificial neural networks," *Opt. Express*, Vol. 30, No. 8, 2022, pp. 13197–13225. https://doi.org/10.1364/OE.454525, URL https://opg.optica.org/oe/abstract.cfm?URI=oe-30-8-13197.

[26] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimization," , 2017.

[27] Chollet, F., et al., "Keras," , 2015. URL https://github.com/fchollet/keras.

[28] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," , 2015. URL https://www.tensorflow.org/, software available from tensorflow.org.

[29] Köpp, F., Rahm, S., Smalikho, I., Dolfi, A., Cariou, J.-P., Harris, M., and Young, R. I., "Comparison of Wake-Vortex Parameters Measured by Pulsed and Continuous-Wave Lidars," *Journal of Aircraft*, Vol. 42, No. 4, 2005, pp. 916–923. https://doi.org/10.2514/1.8177.