Received: January 28, 2021. Revised: June 9, 2021. Accepted: June 27, 2021. Published: July 8, 2021.

Wake Vortex Detection by Convolutional Neural Networks

Nikolay Baranov[a], Boris Resnick[b]

[a]Dorodnicyn Computing Centre, FRC CSC RAS, Moscow, Russia
baranov@ians.aero

[b]Moscow State University, Moscow, Russia,
boris@resnick.ru

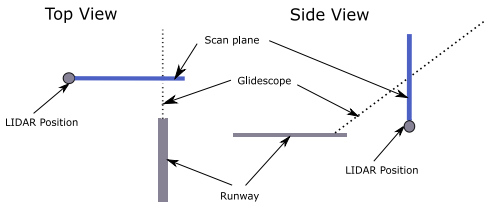

Fig. 1: Scanning Geometry Overview

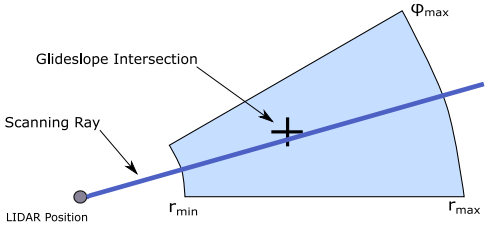

Fig. 2: Scanning Plane

Abstract- This paper describes one approach to use convolutional neural networks to detect wake vortices generated by aircraft based on scanning Doppler LIDAR measurements.

Keywords- Wake Vortex, LIDAR, Convolutional Neural Network, Object Detection

I. A Problem of Wake Vortex Detection

WAKE vortices generated by heavy aircraft may present a significant hazard for other aircraft, especially of a lighter category ([1]). To counter this hazard, aircraft operations in a densely occupied airspace require application of some additional separation distance, leading to reduced capacity. Although there is a number of modelling methods ([9], [10]) to predict vortex behaviour, strength and decay, direct detection bears a significant challenge.

Scanning Doppler LIDARs ([4], [7], [8]) are the one means to directly measure these wake vortices. A neural network approach to LIDAR-based real-time detection of wake vortex presence on glide slope is a task of this study.

II. Data acquisition with LIDAR

Neural network require a significant amount of data to perform training and validation. We use a comprehensive data set acquired by several LIDARs deployed in major airports. These airports exercise a representative traffic mix with more than 50% heavy and super-heavy aircraft.

Generally a LIDAR is installed in an immediate vicinity of a runway, with a nominal lateral offset of 300 meters off a center line and a nominal downwind offset of 900 meters off a threshold.

Scanning geometry takes into account a known manner of vortex behaviour. An induced turbulence is generated at aircraft's altitude, evolves into a distinct vortex structure several seconds later and then decays, gradually moving towards the ground.

LIDAR scans in a vertical plan orthogonal to an extended runway center line. Scanning elevation angles are generally set from zero (horizontal) (further designated as φ_{min}) to 30 degrees high (φ_{max}). This configuration ensures capturing a glide slope intersection point, as well as the most of vortex presence area.

LIDAR digital output, named the "raw data", can be represented as a set of scans. Each scan corresponds to a single pass of LIDAR scanner's reciprocating motion from the lowest elevation angle to the highest, or vice versa. At the each elevation angle (i.e along a scanning ray), LIDAR senses a projection of a wind speed (v) to a scan line at a set of distances. Scanning distances are determined by LIDAR capabilities and current geometry, with minimal distance (r_{min}) generally set to 100 meters, and maximum (r_{max}) set to 600 meters.

Thus, a LIDAR scan can be represented in discrete polar coordinates as follows:

$$\begin{aligned} &S = \{v(\varphi_i, r_j)\}, \\ &\varphi_{min} \le \varphi_i \le \varphi_{max}, \\ &r_{min} \le r_j \le r_{max}, \\ &i = 1..K, \quad j = 1..M. \end{aligned} \tag{1}$$

Numbers of measurements in a single scan $N = KM$ is determined by scan's angular (0.5°) and range (3 m) resolution, and timing constrains. A typical scan contains 10000 measurement points.

III. Vortex Detection Algorithms

In this study we want to estimate a number of vortices present and their location using a single LIDAR scan. As a vortex is a dynamic 3D object, a LIDAR can capture only a limited amount of vortex information. As one scan is fast enough, we assume that a single scan represents a single section of a vortex volume by a scanning plane. A

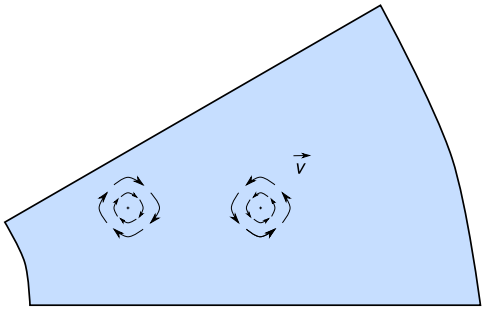

Fig. 3: Wind Speed Field for Vortex Sections

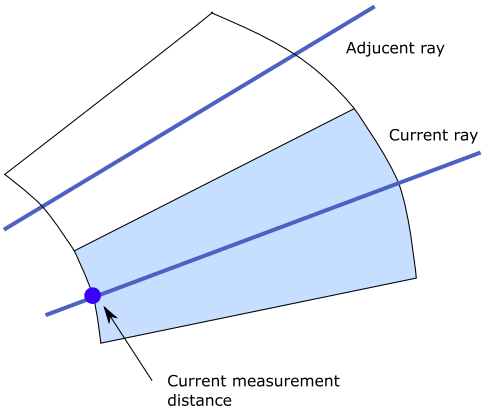

Fig. 4: Elementary Image Cell

nature of a vortex is such that the major part of a wind speed vector projects to the scan plane.

Existing *ad hoc* algorithms ([2], [3]) are based on direct analysis of $v_{i,j}$ value behaviour. Specifically, high gradient values of v, together with change of sign (i.e. direction) along the adjacent ray, constitute the presents of a vortex. This approach has both advantages and disadvantages.

As a main advantage, these algorithms can be used to derive quantitative characteristics of a vortex: coordinates and circulation strength. However, these methods work under two assumptions: firstly, a vortex has a delineated circular structure, and, secondly, radial wind speed distribution has a particular shape.

The last assumption is not precisely correct near a ground plane, when its interference with a vortex creates secondary vortex structures, hence overall vortex geometry deviates from a circular shape.

To address this problem, we follow an alternative approach which is less sensitive to *a priori* vortex structure.

IV. LIDAR Scans as Images

For a human, the primary tool to assess a vortex presence on a LIDAR scan is to examine scan's 2D color rendering — a conventional digital image. As modern convolutional neural networks (CNN) reach human ability in image recognition, in our study we try to leverage these capabilities for wake vortex detection. As with all neural networks, there are several basic steps required:

- Acquire a data set, split into training, test and validation subsets.
- Label the training set and the test set.
- Train the network.
- Run inference over a validation set and assess the performance.

A. Scan Conversion

First, we need to define a deterministic way to transform a LIDAR scan to an image. As a reference grid we choose a rectangular area in a scan plane. CNNs require a fixed image resolution.

We assume the following scan parameters and a linear resolution of 1 meter:

$$\begin{aligned} &\varphi_{min} = 0^\circ, \quad \varphi_{max} = 30^\circ, \\ &r_{min} = 100, \quad r_{max} = 600, \\ &\delta = 1. \end{aligned} \tag{2}$$

We choose our linear resolution to be several times higher than a nominal physical resolution. Hence, horizontal resolution in pixels is given by:

$$W = (r_{max} - r_{min})\,/\delta = 500.$$

Minimal vertical resolution calculates as

$$H = (r_{max} - r_{min})\sin(\varphi_{max}) = 250.$$

So we use the following cartesian grid:

$$(x_p, y_q),\ p = 1..W,\ q = 1..H.$$

Next, we extrapolate a single LIDAR-measured value $v_{i,j}$ to an elementary scan element, a "cell". A cell is formed by adjacent rays and consequent measurement distances as depicted on Figure A.

The cell is defined by four grid points $(x_l^{i,j}, y_l^{i,j})$, $l = 1..4$. Assuming that φ_j is a current scan ray elevation, d_i is a current measurement distance, $\Delta\varphi$ is an angle between the rays, we derive these points:

$$\begin{aligned} x_1^{i,j} &= d_i \cos\varphi_j - \Delta\varphi, \\ x_2^{i,j} &= d_i \cos\varphi_j + \Delta\varphi, \\ x_3^{i,j} &= d_{i+1} \cos\varphi_j + \Delta\varphi, \\ x_4^{i,j} &= d_{i+1} \cos\varphi_j - \Delta\varphi, \\ \\ y_1^{i,j} &= x_1^{i,j} \tan\varphi_j - \Delta\varphi, \\ y_2^{i,j} &= x_2^{i,j} \tan\varphi_j + \Delta\varphi, \\ y_3^{i,j} &= x_3^{i,j} \tan\varphi_j + \Delta\varphi, \\ y_4^{i,j} &= x_4^{i,j} \tan\varphi_j - \Delta\varphi. \end{aligned} \tag{3}$$

Then scanning plane coordinates are transformed to

the image coordinates as following:

$$\begin{aligned} H_{min} &= r_{min}\cos(\varphi_{max}), \\ H_{max} &= r_{max}, \\ V_{min} &= r_{min}\sin(\varphi_{min}), \\ V_{max} &= r_{max}\sin(\varphi_{max}), \\ p &= \frac{W(x - H_{min})}{H_{max} - H_{min}}, \\ q &= \frac{H(y - V_{min})}{V_{max} - V_{min}}. \end{aligned} \tag{4}$$

Futher, we denote this transform as $(p, q) = T(x, y)$.

All grid points in the cell are set to the value corresponding to the current measurement $v_{i,j}$. We need to convert the measurement, a single real value, to a color triplet. The most common RGB color model is fit this purpose. General approach would be to use a standard graphics library. However, this process may or may not be deterministic, and not always well-documented. To address this problem, we apply four steps:

- Remove influence of the ambient wind by substructing an average wind speed value V.
- Truncate wind speed values by a reasonable maximum, for which we choose $v_{max} = 30m/s$.
- Apply a linear transform of a wind speed range to a visual light wavelength range in nanometers ($\lambda_{min} = 380$, $\lambda_{max} = 750$), which is naturally converted into a color.
- Apply a transform based on a formula presented in [12]: $(R, G, B) = \mathcal{C}(\lambda)$.

To remove the ambient wind and truncate, we apply:

$$\begin{aligned} V &= \frac{1}{N}\sum_{i=1}^{K}\sum_{i=1}^{M} v_{i,j}, \\ \overline{v} &= \max\left\{\frac{v}{V},\ v_{max}\right\}. \end{aligned} \tag{5}$$

Then the range is transformed as:

$$\begin{aligned} \hat{v}_{i,j} &= \frac{\overline{v}_{i,j} + v_{max}}{2v_{max}}, \\ \lambda_{i,j} &= \lambda_{min} + \hat{v}_{i,j}\left(\lambda_{max} - \lambda_{min}\right). \end{aligned} \tag{6}$$

The final color applied to a cell corresponding to $v_{i,j}$ equals

$$(R^{i,j}, G^{i,j}, B^{i,j}) = \mathcal{C}(\lambda_{i,j}).$$

Applying there values to all pixels within a polygon $T(x_l^{i,j}, y_l^{i,j})$, $l = 1..4$ for all cells $i = 1..K$, $j = 1..M$, we acquire a final image, denoted as a tensor

$$\mathcal{I} = \begin{bmatrix} (R_{1,1}, G_{1,1}, B_{1,1}) & \dots & (R_{1,H}, G_{1,H}, B_{1,H}) \\ \dots & \dots & \dots \\ (R_{W,1}, G_{W,1}, B_{W,1}) & \dots & (R_{W,H}, G_{W,H}, B_{W,H}) \end{bmatrix}, \tag{7}$$

where $p = 1..W$, $q = 1..H$.

An example is given by Figure 5.

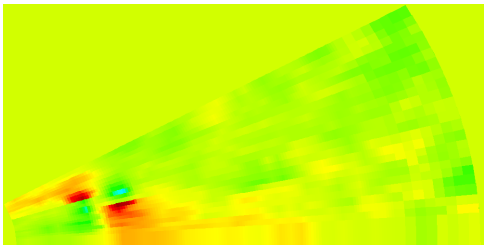

Fig. 5: An example of LIDAR Scan Rendering

V. Training and Test Images Generation

Our initial data set contains LIDAR scan with a variety of noise, background disturbance and scanning actifacts, and the most common measurements do not contain any vortices at all. On the other hand, to train a neural net, we require an extensive set of images with coordinates of a vortex are labeled appropriately. As there exist a significant number of vortex shapes and sizes, a manual selection was disregarded in the scope of this paper as too time-consuming. Also, in an absence of a robust *ad hoc* detection algorithm, automated labelling is also problematic.

To address these concerns, we apply the following procedures:

- LIDAR measurements modelling.
- Ambient conditions measurements labelling.
- Image blending.

A. Modelling

To avoid extensive manual vortex detection and labelling, we use a model described in [5], [6] to generate a comprehensive set of modelled LIDAR measurements for certain vortex parameters.

Modelling is two-phased. First, we model the vortex as a 3D wind speed field, and then a LIDAR measurement model used to convert this field into a set of measurements which simulate a real scan defined by (1).

In order in acquire a representative vortex collection, each of the vortex parameters is varied in a predefined range by a predefined step value or chosen from a list, yielding a set of parameter's values. Combined, these sets form a cartesian product, and for each element a LIDAR scan is modelled.

The following parameters are used:

- Aircraft type. Values: Boeing B767, B777, B787, Airbus A380, A350, A340, A330, A319, A320. Number of elements $C_1 = 9$.
- Scan delay from scanning plane intersection time. Values: 10, 12, 15 seconds. $C_2 = 3$.
- Vortex generation height. Values: 75 meters, 60 meters. $C_3 = 2$.
- Ambient wind speed. Minimum $-2m/s$. Maximum $4m/s$. $C_4 = 7$.

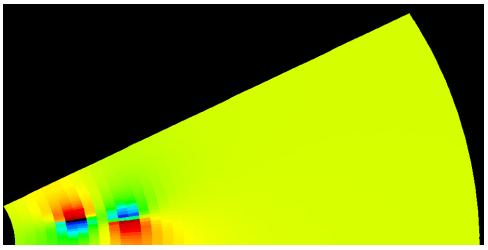

Fig. 6: Modelled LIDAR Scan Rendering

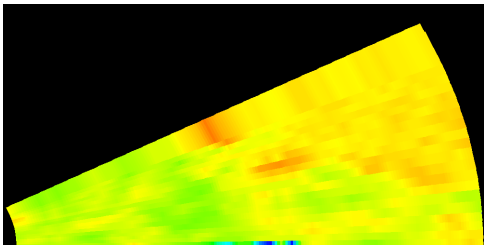

Fig. 7: Ambient Conditions LIDAR Scan Rendering

- Wind direction. Minimum 0°. Maximum 0.3°. $C_5 = 8$.

Thus, we generate $C_g = \prod_{i=1}^{5} C_i = 3024$ different vortex models. Each model is rendered as an image using the same procedure as described in Section A.

Using a notation defined by (7), this image set is defined as $\mathcal{I}^{gen,g},\ g = 1..C_g$.

An example generated vortex image is presented on Figure 6.

In addition to the scan data, our model yields two pairs of vortex coordinates $(X_1, Y_1),\ (X_2, Y_2)$ in a scan plane, and vortex radii $R_1,\ R_2$.

B. *Ambient Conditions Scans*

As one can see from the above example, although the model can simulate a vortex itself, a generated image lacks adequate representation of ambient wind conditions. Instead of using a corresponding model, this study adopts a different approach. We select a set of LIDAR scans which do not contain any vortices. These scans were taken in diverse conditions, reflecting various wind environments. This selection is performed manually with a help of a labelling tool. Somewhat arbitrarily we choose $C_a = 50$ different scans.

An example of a selected scan $\mathcal{I}^{amb,g},\ g = 1..C_a$, is presented on Figure 7.

C. *Image Blending*

Finally, we generate a set of images suitable to be used as neural network inputs.

First, a basic background image $\mathcal{I}^{back}$ is generated as follows:

$$\mathcal{I}^{back}_{p,q} = \mathcal{C}(0),\ \forall p = 1..W,\ \forall q = 1..H$$

.

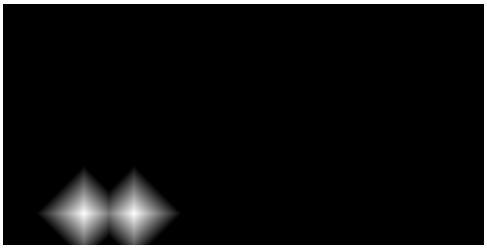

Fig. 8: Blending Mask

Second, for every generated image $\mathcal{I}^{gen,g},\ g = 1..C_g$ we randomly select an ambient condition image $\mathcal{I}^{amb,a^*},\ a^* \in \{1, \ldots, C_a\}$ and generate the final images using the following blending procedure.

Third, a blending mask is derived from the set of vortex coordinates, in such a way so the target image contains both generated vortex and ambient conditions. The weight of a generated information is higher when a distance is lower.

In image grid coordinates, vortex coordinates on g-th generated image calculate as:

$$\begin{aligned}(P_1^g, Q_1^g) &= T\left(X_1^g, Y_1^g\right),\\ (P_2^g, Q_2^g) &= T\left(X_2^g, Y_2^g\right).\end{aligned} \tag{8}$$

Assuming D_M is a Manhattan distance function

$$D_M(p, q, p_0, q_0) = |p - p0| + |q - q0|,$$

a mask calculates as a sum of the first vortex mask

$$\mathcal{M}^{g,1}_{p,q} = \max\left\{0, 1 - 0.01 D_M(p, q, P_1^g, Q_1^g)\right\},$$

and the second vortex mask

$$\mathcal{M}^{g,2}_{p,q} = \max\left\{0, 1 - 0.01 D_M(p, q, P_2^g, Q_2^g)\right\},$$

so the final mask equals

$$\mathcal{M}^g = \mathcal{M}^{g,1} + \mathcal{M}^{g,2}. \tag{9}$$

An example of a blending mask is shown on Figure 8.

As the last step of input data preparation, for $g = 1..C_g$ we generate a set of training input images as

$$\mathcal{I}^{input,g} = \mathcal{M}\mathcal{I}^{gen,g} + (1 - \mathcal{M})\mathcal{I}^{amb,a^*} + \mathcal{I}^{back}.$$

An example of an input image is shown on Figure 9.

Additionally, each input image is augmented with labelling data in PASCAL VOC XML format. Labelling data contain vortex positions in image grid coordinates. A simple rectangular label is used for each of two vortices. To contruct these labels, we use vortex radii $R_{1,2}$ yielded by the modelling procedure. Two label rectangles $\left(p^{1,2}_{min}, q^{1,2}_{min}, p^{1,2}_{max}, q^{1,2}_{max}\right)$ are defined by:

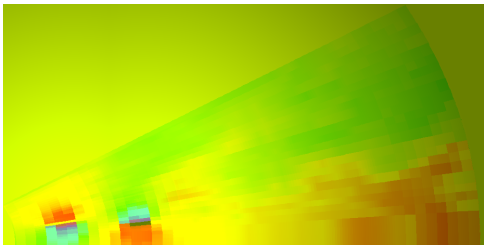

Fig. 9: Blended Scan Rendering

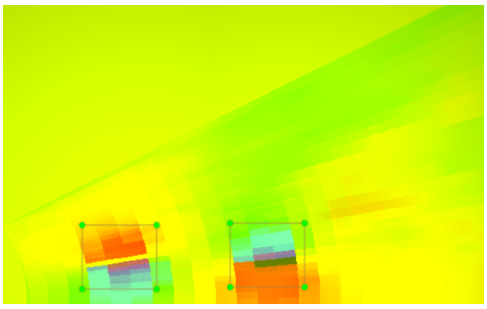

Fig. 10: Rectangular Labels for the Vortices

$$
\begin{aligned}
X^{1,2}_{min} &= X^{1,2} - R_{1,2}, \\
X^{1,2}_{max} &= X^{1,2} + R_{1,2}, \\
Y^{1,2}_{min} &= Y^{1,2} - R_{1,2}, \\
Y^{1,2}_{max} &= Y^{1,2} + R_{1,2}, \\
\left(p^{1,2}_{min}, q^{1,2}_{min}\right) &= T\left(X^{1,2}_{min}, Y^{1,2}_{min}\right), \\
\left(p^{1,2}_{max}, q^{1,2}_{max}\right) &= T\left(X^{1,2}_{max}, Y^{1,2}_{max}\right).
\end{aligned}
\tag{10}
$$

An example of label rectangles is shown on Figure 10.

VI. Neural-Network Based Detection Models

There are multiple image recognition architectures [11] readily available to train on compatible data set. For this study, we use a well-known Faster R-CNN method and Inception V2 architecture, pre-trained on the COCO data set.

Neural network training use a procedure described in [13]:

- Tensorflow library is configured to use GPU acceleration.
- Blended images data and labelling data are converted to TensorFlow data records.
- Training parameters are set:
  - Number if classes: 1 ('Vortex');
  - Number of steps: 200000;
  - Training data set: 80% of samples ($0.8C_g$);
  - Testing data set: 20% of samples.

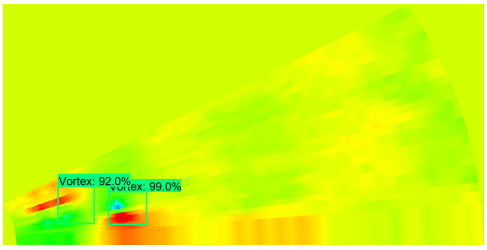

Fig. 11: The Vortices Detected on a Scan

- Training sequence is launched.
- A target trained model is exported.

A. Detection Results

To validate our network, we apply an inference procedure based on the exported model to real LIDAR scans. The model outputs an estimate of detected vortex bounding box coordinates. Then we overlay these coordinates on a scan render to visualized detection results. A detection example is presented on Figure 11.

VII. Future Research

This study uses a straighforward approach to detect vortices based to translation of a scan to an image. There are several ways to develop this approach further:

- More Wake Models: there are multiple emerging object detection models and neural network architectures. A study is required to assess whether these are more efficient to detect vortices.
- Polar Coordinates Detection: a LIDAR scan is naturally described in polar coordinates. However, a conventional CNN uses cartesian coordinates. There are models which address this. One can investigate whether these are applicable to vortex detection.
- Dynamic Approach: A vortex is a dynamic 4D object. In this study, we analyse a vortex section in isolation based on a single LIDAR scan. Considering several adjacent sections in a single model may enhance detection quality.

References

[1] F. Holzäpfel, "Analysis of potential wake vortex encounters at a major European airport", Aircraft Engineering and Aerospace Technology, Vol. 89, No. 5 (2017), pp. 634–643.

[2] V.A. Banakh, I.N. Smalikho, "Aircraft Wake Vortex Parametrization Based on 1.5-μm Coherent Doppler Lidar Data", EPJ Web of Conferences 119 14002 (2016).

[3] D. P. Delisi, G. C. Greene, R. E. Robins, D. C. Vicroy, F. Y. Wang, "Aircraft Wake Vortex Core Size Measurements", 21st AIAA Applied Aerodynamics Conference, 23-26 June 2003. AIAA 2003-3811.

[4] Proctor, F.H., 1996: "Numerical Simulation of Wake Vortices Measured During the Idaho Falls and Memphis Field Programs", AIAA Paper 96-2496, 14th AIAA Applied Aerodynamic Conference, New Orleans.
[5] Matayoshi N., Yoshikawa E., Yamamoto M., "Wake Vortex Measurement Using Pulsed Doppler Lidar', 19th Coherent Laser Radar Conference 2018.
[6] H. Gao, J. Li, P.W. Chan, K.K., "Hon Parameter retrieval of aircraft wake vortex based on its max–min distribution of Doppler velocities measured by a Lidar", J. of Engeneering. Vol. 2019, Iss. 20. P.p. 6852-6855.
[7] L. Thobois, J.-P. Carioul, V. Cappellazzo, C. Musson and V. Treve, "Comparison and validation of wake vortex characteristics collected at different airports by different scanning lidar sensors", EPJ Web of Conferences (2018), Vol. 176, paper 06002.
[8] F. Holzäpfel, T. Gerz, F. Köpp, E. Stumpf, M. Harris, R. I. Young and A. Dolfi-Bouteyre, "Strategies for Circulation Evaluation ofAircraft Wake Vortices Measured by Lidar", J. of Atmospheric and Oceanic Technology (2003), Vol. 20, No. 8, pp. 1183–1195.
[9] F. H. Proctor, D. W. Hamilton, and J. Han, "Wake Vortex Transport and Decay in Ground Effect: Vortex Linking with the Ground", AIAA Paper No. 2000–0757 (2000).
[10] M. Lin, W. Huang, Z. Zhang, C. Xu and G. Cui, "Numerical study of aircraft wake vortex evolution near ground in stable atmospheric boundary layer", Chinese Journal of Aeronautics (2017), Vol. 30, Iss. 6, pp. 1866–1876.
[11] Jonathan Huang et al. "Speed/accuracy trade-offs for modern convolutional object detectors", 2017.
[12] Mihai, Dragoş and Străjescu, "From Wavelength to RGB Filter", U.P.B. Sci. Bull., Series D, V. 69, N.2, 2007.
[13] GitHub Resource. "Training an Object Detector using Tensorflow API on Ubuntu 16.04 (GPU)", URL: https://github.com/Khaivdo/How-to-train-an-Object-Detector-using-Tensorflow-API-on-Ubuntu-16.04-GPU/. Accessed on 07 March 2021.

Contribution of individual authors to the creation of a scientific article

Nikolay Baranov lead the study and implemented vortex data generation algorithms

Boris Resnick implemented input images generation algorithm and a learning framework