ISSN 2075-1087, Gyroscopy and Navigation, 2025, Vol. 16, No.1, pp. 21-32.
Russian Text © The Author(s), 2025, published in Giroskopiya i Navigatsiya, 2025, No.1, pp. 36–51.

Algorithms of Runway Detection in Video Images in Neural Network Based UAV Landing

K. A. Chekanov

Ramenskoye Design Company, Ramenskoye, Russia
e-mail: chekanov1993@gmail.com
Received: November 27, 2024; reviewed: December 22, 2024; accepted: April 1, 2025

Abstract: Algorithms for detecting the runway contours in video images based on YOLOv8 neural network are presented, differing by the types of problems it is trained to (detection, segmentation, pose estimation). The accuracy and speed of these algorithms run on NVIDIA Jetson NANO computer module are analyzed. Using the analysis results, the best detection algorithm is selected based on certain parameters (speed, accuracy, range). The results confirm that the algorithm can be applied in onboard software of unmanned aerial vehicles (UAV).

Keywords: unmanned aerial vehicle (UAV), landing, machine vision, neural networks.

1. INTRODUCTION

Normally, the coordinates of UAV at all flight stages are determined by means of a satellite navigation system (SNS). It has high accuracy (especially in the differential mode), but is susceptible to artificial interference. A computer vision system can be a compact and autonomous solution, when the UAV is equipped with a video camera, and the navigation solution (coordinates, speed) is generated by analyzing the obtained image.

The task of determining an object's own position in the computer vision is called pose estimation [1]. Its solution can be based on various principles. For example, in the review article [2], video navigation algorithms are considered, where a digital map is used as a priori information, and the coordinates are determined by matching a photo image obtained in flight with the reference. The authors of [2] propose to divide the image-based navigation algorithms into the following categories:

1) algorithms involving the correlation-extreme approach;

2) algorithms using the feature points;

3) algorithms using the neural networks (NN).

At best, these algorithms ensure the accuracy at the SNS level (without differential mode), but in fact it is often lower: the average Euclidean distance (AED) relative to the SNS is usually 15–20 m. This error is acceptable for the modes where precise positioning is not required (for example, en-route flight), but it is absolutely inappropriate for landing (5.6 m in the horizontal plane [3]).

Another principle is based on determining the position of the camera relative to the subject with a priori known coordinates set in some global coordinate frame. Then the problem is solved by the photogrammetry methods [4], such as Perspective-n-Point (PnP) (i.e., perspective of the n-th number of points) [5].

In [6], it is proposed to use the PnP method to determine the UAV coordinates during gliding, relative to the infrared (IR) beacons installed near the runway. The coordinates of the IR-beacons are considered to be known a priori. The error in determining the UAV's own 3D coordinates at the final stage of gliding is less than 1 m. This result fully satisfies the requirements for landing errors, so the PnP method can be taken as the main one for generating a navigation solution in the proposed conditions. However, the beacons installation and maintenance complicate the ground infrastructure of the airfield. Therefore, the corner points of the runway can act as an alternative to visual markers, since their coordinates are also known a priori from the atlases of airfield navigation charts. The accuracy of the PnP method depends directly on the accuracy of the a priori points (the runway corners) found in the image.

This article is dedicated to the problem of runway recognition in a video during the UAV landing in the absence of SNS signals, i.e. when the UAV's own coordinates are unknown. Three algorithms using the YOLOv8 NN trained for different target functions are compared.

The input parameters for solving the problem are the color video image from the on-board camera and the a priori known coordinates of the runway, and the result should be the coordinates of the vertices of the runway contour seen in the image.

The structure of the work is as follows. Section 1 reviews the work related to the subject area and provides the reasons for the need to develop a new algorithm for runway recognition. In section 2, the problem statement is formalized and various options for NN operations with an image are considered: detection, segmentation, and pose estimation. The use of the latter two operations as part of runway recognition algorithms is described in the same section, and detection is discussed further. Section 3 describes a runway recognition algorithm implemented with NN-detection of an area of interest and subsequent post-processing: highlighting the contrasting corners of a given area, applying the BEV transformation to them, and searching for a rectangular shape most similar to a landing runway, using a genetic algorithm. In section 4, a method is proposed for collecting a learning dataset (using a flight simulator) and its automatic labeling. Section 5 describes simulation where the performance of the recognition algorithms based on all three approaches is assessed. The main conclusions are presented in the final section.

2. WORKS IN THE SUBJECT AREA CONCLUSION

Let us review the publications on runway recognition.

The paper [7] describes a study involving a flight simulator. For the convenience of recognition, contrasting landmarks free of IR radiation are placed in the corners of the runway. The algorithm of the runway search in the image is based on the classical computer vision techniques: binarization, Canny edge extraction, and Hough transform. According to the simulation, the recognition range was 1200 feet (~365 m).

In [8], the authors use the markings at the beginning of the runway to detect the runway in the image. It looks like the road marking of a pedestrian crossing, i.e., a series of contrasting rectangles. The runway markings are standardized, so their parameters can be used as a priori information for the PnP. The disadvantage of this solution is that the markings are hardly distinguishable at long distances. The authors point out that during the tests their algorithm started working from a distance of 718 m to the runway threshold.

In 2018, Airbus together with Wayfinder launched the Autonomous Taxi, Take-off and Landing (ATTOL) project [9]. Within its framework, an automatic landing system was introduced in 2020, which implemented runway recognition using artificial intelligence [10]; however, the specific type and architecture of the predictive model are not disclosed in open sources.

In 2019, a team from the Technical University of Munich presented a project of the C2Land automatic landing system [11]. To conduct field tests, the system was installed on a Diamond DA42 light-engine aircraft. In [12], the authors describe in detail the C2Land architecture and the computer vision unit. C2Land uses the readings from a strapdown inertial navigation system (SINS) and images from two cameras (optical and infrared ones) (Fig. 1). The runway is recognized in the images and the relative location of the aircraft is determined, which is then estimated in conjunction with other sensors.

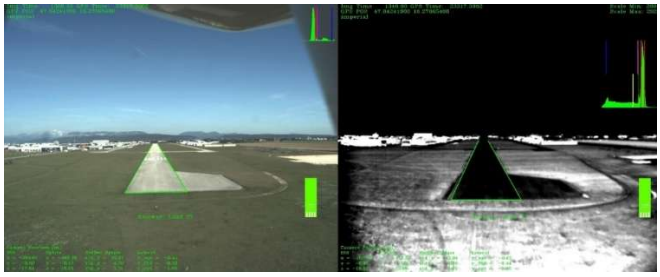

Fig. 1. Frames from optical (left) and infrared (right) cameras

In [13], the authors assessed the performance of C2Land. The infrared camera consistently recognized the runway from about 1500 m away, while the RGB camera did so from 500 m.

Let us estimate the minimum required distance of runway detection $D_{\min}$. According to the aviation regulations [14] based on the ICAO requirements, the concept of the minimum descent height (MDH) is introduced. This is an altitude below which the descent is prohibited if there is no stable visual contact with the runway. The go-around procedure starts at the calculated point of intersection of the descent trajectory (glide path) with the MDH level. The MDH calculation for an inaccurate instrumental approach takes into account the height of obstacles on the descent path and the fixed elevation above them, which can be up to 90 m, depending on various conditions [14]. We consider the ideal case when the height of obstacles compared to the elevation can be ignored, and calculate the distance of the MDH intersection with a standard glide

path [15] (slope 3°, the threshold exceeded by 15 m) relative to the runway threshold: D_{min} = (90 – 15) / tan3° ≈ 1430 m (Fig. 2).

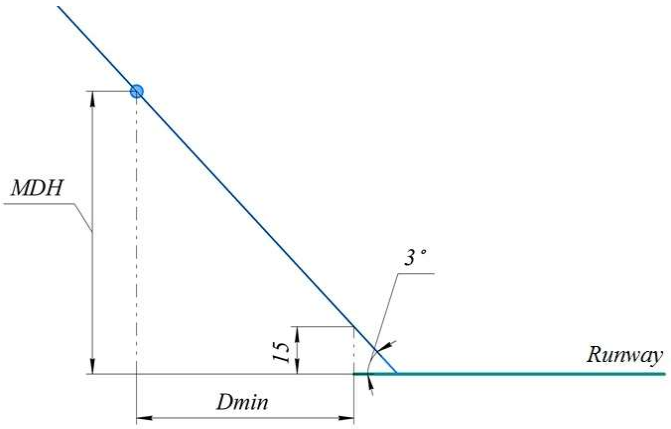

Fig. 2. Illustration to calculation of the minimal distance of runway recognition.

In the papers mentioned above, C2Land reaches the highest distance of recognition, but it practically coincides with the minimum value. Analysis of open sources has shown that there are no solutions with runway recognition distance significantly exceeding D_{min}. At the same time, the problem under study is relevant, and large companies such as Airbus are engaged in its solution. The difficulty of recognition lies in the fact that the runway size in the frame is not constant; it changes as the aircraft is approaching. In order to achieve a better recognition distance, some algorithms based on the use of a NN are proposed and analyzed in this paper.

3. NEURAL NETWORK IN RUNWAY RECOGNITION TASK

As was mentioned above, the corner points are of the main interest in the runway recognition problem in terms of the navigation solution generation. Therefore, we will introduce the assumption that the runway in the image is correctly approximated by an irregular tetragon. The task is to find the runway corners in an image obtained from a camera mounted on the UAV (resolution, focus, and other camera characteristics are known). The developed algorithm that solves this problem is based on the YOLOv8 NN [16]. This architecture was chosen because the YOLO family shows the best performance when working in real time due to the search for objects in one scan of the image [17].

The selected NN within the framework of the developed algorithms can be used for solving the three tasks shown in Fig. 3.

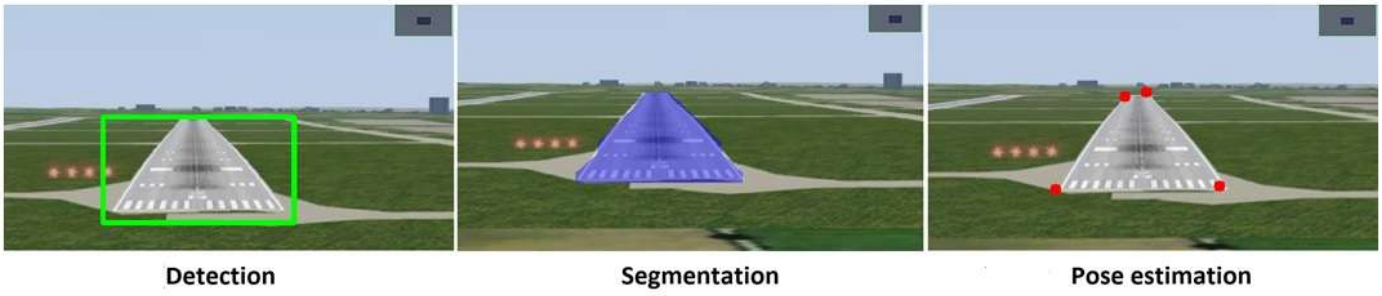

Fig. 3. Illustration to the tasks solved by YOLOv8.

Detection is the simplest task for a NN. The coordinates of the bounding rectangle, which is actually the area of interest, are formed on its output layer. To find the corner points of this area, some additional algorithms are required, which will be discussed in the next section.

In the segmentation task, a mask containing the runway is formed on the NN output layer (Fig. 4a). To detect the runway corners, the following algorithm is used: the mask outline along its perimeter is formalized by an approximating contour selected using the Douglas–Peucker algorithm [18], and then an oriented rectangle of the minimum area is described around the contour (Fig. 4b). According to the direction of its orientation, a rectangular 2D coordinate frame (CF) is established in the center, and in each of its quadrants the contour point furthest from the abscissa axis is selected (Fig. 4c).

The pose estimation task is originally intended for identifying the feature points in an image of the human body. In this mode, the NN can be further learned to search for other feature points, such as runway corners. This approach is interesting because the final result, i.e., the coordinates of the runway corner points, will be generated on the NN output layer without any additional processing.

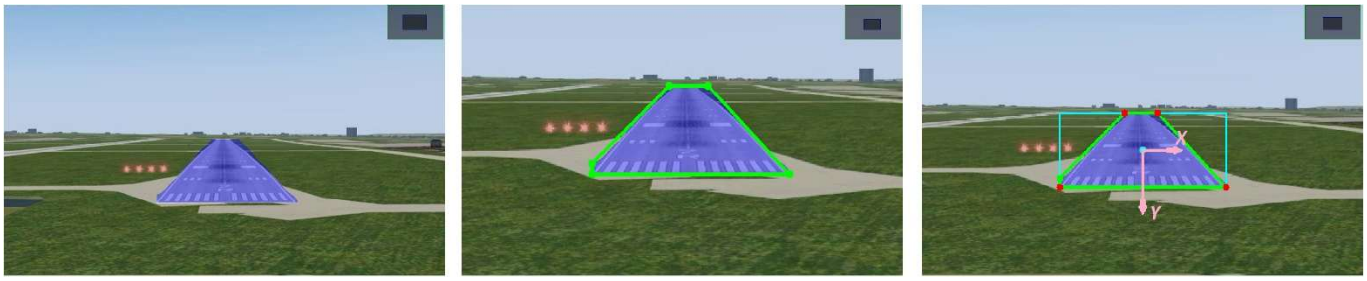

Fig. 4. Runway corners detection using a segmenting NN

4. RUNWAY RECOGNITION USING A DETECTING NEURAL NETWORK

As was noted earlier, the detecting NN forms a rectangular area of interest, containing the desired object. We will select all the contrasting corners in this area using the Shi-Tomasi detector [19] (Fig. 5).

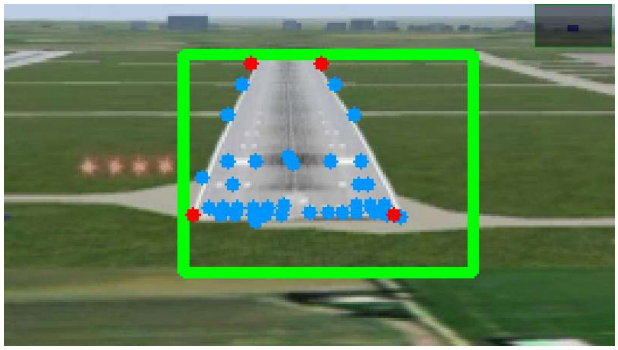

Fig. 5. Corners selection by the Shi-Tomasi detector.

A cloud of feature points marked in blue is formed at the detector output. It is necessary to identify four points in the cloud that relate to the runway corners (marked in red). It is difficult to do this in the initial frame, because when shot from different angles, the runway can take the shape of a trapezoid or a parallelogram (in other words, an irregular tetragon). In reality, the runway has a strictly rectangular shape and retains it when viewed from above. This fact will help us in the future, so we will use projective transformations to bring the image in the frame to the top view.

Correction of Projective Distortions and BEV Transformation

Homography is a special case of perspective transformation, namely the perspective transformation of a plane, i.e., mapping the points of a 2D flat surface onto the photomatrix screen plane. We assume that the runway surface is close to flat. Figure 6 shows the perspective correction problem statement: the perspective of camera position *1* needs to be brought to position *2* (top view), where *P* is some point belonging to the area of interest.

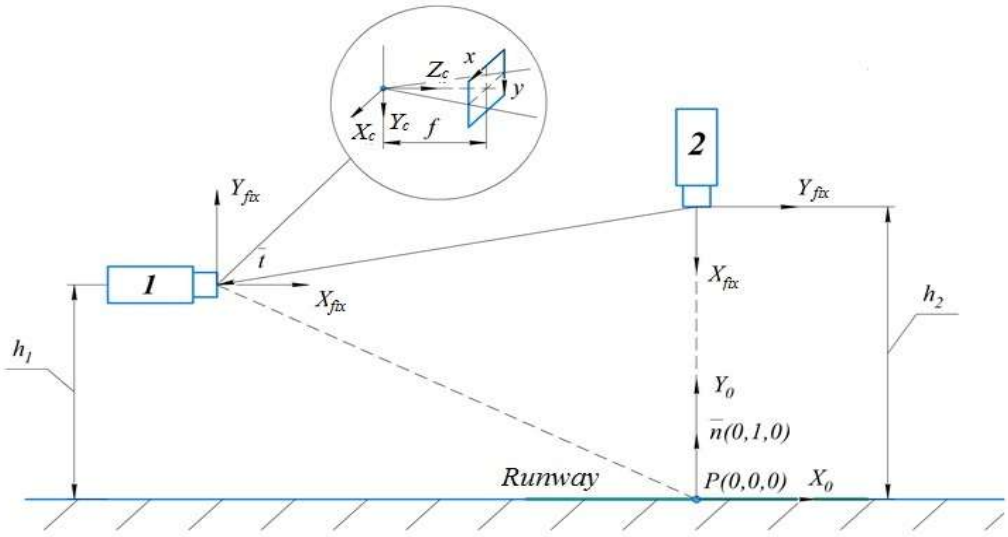

Fig. 6. Perspective correction

The transformation that brings a perspective to a top view is called Bird's Eye View (BEV) [20]. Its homography matrix is written as

$$P_{S2} = H \cdot P_{S1},$$
$$H = D_{S2}^{K2} \cdot D_{C2}^{fix2} \cdot \overline{D}_{fix2}^{fix1} \cdot D_{fix1}^{K1} \cdot D_{C1}^{S1}, \quad (1)$$

where P_{S1}, P_{S2} are the screen coordinates of point *P* in positions *1* and *2*, respectively; *H* is the homography

matrix; D_{C1}^{S1} is the matrix of transition from screen coordinates *1* to the camera coordinate frame CF *1*; D_{fix1}^{C1} is the matrix of transition from the camera CF *1* to the fixed CF *1*; $\overline{D}_{fix2}^{fix1}$ is the rotation matrix with displacement from the fixed CF *1* to the fixed CF *2*; D_{C2}^{fix2} is the matrix of transition from the fixed CF *2* to the camera CF *2*; D_{S2}^{C2} is the matrix of transition from the camera CF *2* to the screen coordinates *2*.

The pinhole is used as the camera model. Its coordinate transformation matrix is traditionally [1] written as

$$D_{S2}^{C2} = (D_{C1}^{S1})^{-1} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (2)$$

where f_x, f_y is the camera focal point in the horizontal and vertical directions of the photomatrix (we assume by default that $f_x = f_y = f$); c_x, c_y are the screen center coordinates.

In order to not distort the classical form of the matrix of camera parameters (2), the camera CF and the fixed CF were presented as different coordinate frames. Their mutual transformation is carried out by the matrix

$$D_{C2}^{fix2} = (D_{fix1}^{C1})^{-1} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}. \quad (3)$$

Detailed derivation of matrix $\overline{D}_{fix2}^{fix1}$ is described in [21]. Using the notation adopted in this paper, the final result can be written as follows:

$$\overline{D}_{fix2}^{fix1} = D_{fix2}^{fix1} + \frac{t_2 n^T D_0^{fix1}}{h_1}, \quad (4)$$

where $D_{fix2}^{fix1} = D_{fix2}^{0} D_0^{fix1}$ is the rotation matrix between the fixed coordinate frames; D_0^{fix1} is the matrix of rotation from the 1st fixed CF to the normal Earth CF related to point *P*; D_{fix2}^{0} is the matrix of rotation from the normal CF to the 2nd fixed CF; h_1 is the height of the 1st position of the camera relative to the runway; $n^T = [0,1,0]$ is the normal to the runway surface; t_2 is the displacement of the 1st fixed CF relative to the 2nd fixed CF in the 2nd fixed CF.

All rotation matrices included in (4) are calculated using the attitude angles generated by the onboard SINS. The height h_1 can be measured with a barometric altimeter. In (4), displacement t_2 remains unknown. The coordinates of point *P* in the normal CF related to the position *1* are:

$$\begin{bmatrix} P_{X1_g} \\ P_{Y1_g} \\ P_{Z1_g} \end{bmatrix} = D_0^{fix1}(\gamma, \vartheta, \psi = 0) \begin{bmatrix} P_{X1} \\ P_{Y1} \\ P_{Z1} \end{bmatrix}, \quad (5)$$

where P_{X1g}, P_{Y1g}, P_{Z1g} are the coordinates of point *P* in the normal CF (from Fig. 7 $P_{Y1_g} = -h_1$); γ, ϑ, ψ are roll, pitch and yaw; P_{X1}, P_{Y1}, P_{Z1} are the coordinates of point *P* in the 1st fixed CF.

The term ψ = 0 emphasizes that all CF are aligned with the current heading of the UAV (further we will write D_0^{fix1} meaning the rotations only in roll and pitch), which means that this angle can be ignored. The coordinates in the fixed CF can be expressed in terms of the point viewing angles the tangents of which are determined by the formulas

$$\tan\mu = \frac{P_x - c_x}{f}, \quad \tan\varphi = \frac{(P_y - c_y)\cdot\cos\mu}{f}, \quad (6)$$

where μ, φ are the azimuth and elevation angle, respectively; P_x, P_y are the screen coordinates of point *P*.

The coordinates from the fixed CF are combined with the viewing angles by formulas

$$\tan\mu = \frac{P_{Z1}}{P_{X1}}, \quad \tan\varphi = \frac{P_{Y1}}{\sqrt{P_{X1}^2 + P_{Z1}^2}}. \quad (7)$$

We express the coordinates from (7) in terms of the viewing angles and P_{Y1}:

$$\begin{cases} P_{Z1} = P_{X1}\cdot\tan\mu \\ \tan\varphi = \dfrac{P_{Y1}}{\sqrt{P_{X1}^2 + (P_{X1}\cdot\tan\mu)^2}} \end{cases} \rightarrow \begin{cases} P_{Z1} = P_{X1}\cdot\tan\mu \\ \tan\varphi = \dfrac{P_{Y1}}{P_{X1}\sqrt{1 + (\tan\mu)^2}} \end{cases} \rightarrow$$
$$\rightarrow \begin{cases} P_{Z1} = P_{X1}\cdot\tan\mu \\ P_{X1} = \dfrac{P_{Y1}\cos\mu}{\tan\varphi} \end{cases} \rightarrow \begin{cases} P_{Z1} = \dfrac{P_{Y1}\sin\mu}{\tan\varphi} \\ P_{X1} = \dfrac{P_{Y1}\cos\mu}{\tan\varphi} \end{cases}. \quad (8)$$

Now we substitute the result of (8) in (5):

$$\begin{bmatrix} P_{X1_g} \\ P_{Y1_g} \\ P_{Z1_g} \end{bmatrix} = D_0^{\text{св1}} \begin{bmatrix} \frac{P_{Y1}\cos\mu}{\tan\varphi} \\ P_{Y1} \\ \frac{P_{Y1}\sin\mu}{\tan\varphi} \end{bmatrix} \rightarrow \begin{bmatrix} P_{X1_g} / P_{Y1} \\ -h_1 / P_{Y1} \\ P_{Z1_g} / P_{Y1} \end{bmatrix} = D_0^{\text{св1}} \begin{bmatrix} \frac{\cos\mu}{\tan\varphi} \\ 1 \\ \frac{\sin\mu}{\tan\varphi} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}. \quad (9)$$

If we put the P_{Y1} coordinate outside the brackets and divide both sides of the equation by this value, it turns out that everything is known on the right side. To shorten the record, we will replace the right sides with $b_{1\ldots3}$. We find the coordinates of point P from (9):

$$\begin{bmatrix} P_{X1_g} \\ P_{Y1_g} \\ P_{Z1_g} \end{bmatrix} = \begin{bmatrix} -h_1\left(\frac{b_1}{b_2}\right) \\ -h_1 \\ -h_1\left(\frac{b_3}{b_2}\right) \end{bmatrix}. \quad (10)$$

Since the camera at position *2* is located exactly above point P, the coordinates of camera *1* relative to *2* in the normal CF related to position *2* will be determined by the formula

$$t_0 = \begin{bmatrix} -P_{X1_g} \\ h_2 - h_1 \\ -P_{Z1_g} \end{bmatrix} \quad (11)$$

Next, to determine the desired displacement t_2, it is necessary to rotate the vector t_0 using matrix D_2^0. We assume that the camera in position *2* has the attitude angles $\gamma = 0$, $\vartheta = -90°$:

$$t_2 = D_2^0 \cdot \begin{bmatrix} -P_{X1_g} \\ h_2 - h_1 \\ -P_{Z1_g} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -P_{X1_g} \\ h_2 - h_1 \\ -P_{Z1_g} \end{bmatrix} = \begin{bmatrix} h_2 - h_1 \\ P_{X1_g} \\ -P_{Z1_g} \end{bmatrix} \quad (12)$$

The user can adjust the height h_2 depending on the task. Thus, all the matrices in equation (1) are defined, which makes it possible to calculate the homography. After multiplying the initial frame by the homography matrix, we obtain the result shown in Fig. 7.

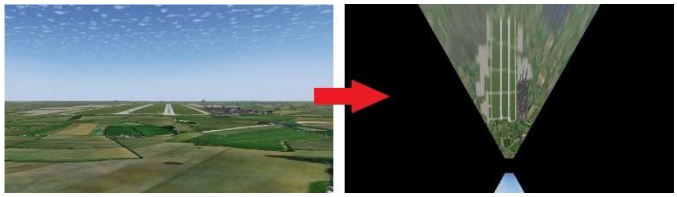

Fig. 7. The result of perspective correction.

It should be noted that Fig. 7 demonstrates the correctness of formulas described in this section, while changing the perspective of the entire frame does not make practical sense. Within the framework of the algorithm being developed, the BEV transformation is applied only to the coordinates of the feature points.

Selection of Feature Points. Genetic Algorithm

After re-projecting the feature points to the top view, the task is to select four points that form a rectangle with the geometric dimensions closest to the runway (the runway characteristics are taken from the aeronautical atlas). To determine the linear distance between the points being viewed, it is necessary to calculate the unit value of pixel. To do this, let us study Fig. 8.

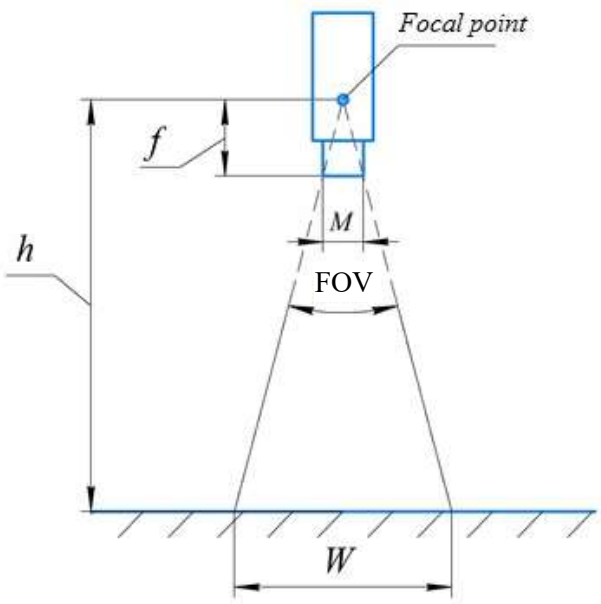

Fig. 8. Determining the pixel unit value: FOV = the field of view, W = the field of view width, M = photomatrix width.

The pixel unit value can be found from the similarity of triangles:

$$\frac{W}{M} = \frac{h \cdot ps}{f}, \quad (13)$$

where ps is the linear size of one pixel (we assume that the pixel is square-shaped).

Complete search of all combinations for 72 points will amount to more than one million variants, which will significantly reduce the speed of the entire algorithm. Therefore, the genetic algorithm (GA) is used in this work to optimize the search. In terms of this approach, each point will be a gene, and the combination of four points will be an individual (a chromosome). Thus, the initial task is to find an individual with the best fitness function, which will be revealed during the evolutionary process.

The fitness function should be a simple algebraic function, because it will be addressed frequently at each step of the calculation. In this case, it has the meaning of the accuracy factor and is implemented in the form of criteria combined by the Chebyshev measure [22]:

$$PF = \min < q_1, ..., q_n >,$$
$$q_i = \frac{|\text{max-val}|}{\text{max-min}} \xrightarrow{\min = 0} q_i = 1 - \frac{|\text{val}|}{\text{max}}, \qquad (14)$$

where PF is the fitness function, and q_i is the normalized criterion.

The second formula denotes the normalization of a criterion of the form "the less, the better" [23], where max, min are the maximum and minimum values of the criterion; val is the current value of the selected criterion.

A priori information about the geometric dimensions of the runway is used as the reference. We denote the reference perimeter, threshold lengths, and area as P_R, W_R, and S_R, respectively. The fitness function is made up of four criteria:

1) $q_1 = 1 - \frac{2|P_S - P_i|}{P_S}$ is the similarity of perimeters, where P_i is the perimeter of the i-th solution;

2) $q_2 = 1 - \frac{2|S_S - S_i|}{S_S}$ is the similarity of areas, where S_i is the area of the i-th solution;

3) $q_3 = 1 - \frac{|2W_S - W_{Ni} - W_{Fi}|}{W_S}$ is the parallelism of the lateral boundaries, where W_{Ni}, W_{Fi} are the lengths of the near and far thresholds of the i-th solution;

4) $q_4 = 1 - \frac{|0.5 - \Psi_i|}{0.5}$ is the similarity of directions, where Ψ_i is the direction of the i-th solution, defined as $\Psi_i = \arccos(\frac{-a_Y}{\sqrt{a_X^2 + a_Y^2}}) \cdot 57.3$; (a_X, a_Y) is the centerline vector of the i-th solution.

The canonical evolutionary process implemented in the algorithm under development consists of the following stages [24] (the terms used in the GA description are indicated in parentheses):

1) formation of a sample (population);

2) selection of the best candidates (selection of parents);

3) crossing (crossing-over);

4) random changes (mutation);

5) breeding;

6) algorithm termination criterion.

The behavior of the algorithm strongly depends on the population size set at the first stage. To select the optimal size in terms of the speed-to-accuracy ratio, an experiment was conducted: for the same set of points, 100 iterations of solving the problem of finding the fitness function maximum with different population sizes (from 2 to 20 individuals) were performed. The average result of the experiment is shown in Fig. 9.

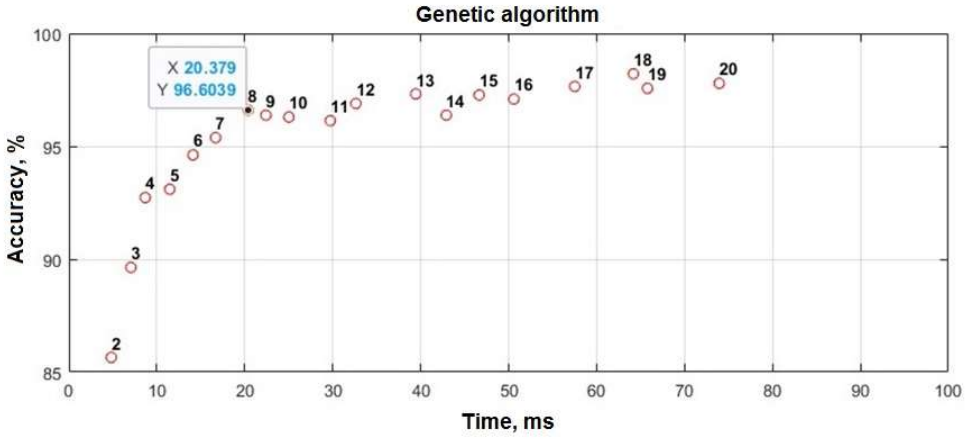

Fig. 9. Experiment to determine the optimal population size

It was found from the experiment that the increase in accuracy slows down when the number of individuals in the population grows above 8, and is only 1% for 20 individuals. Therefore, a population of 8 individuals is further created.

5. LEARNING METHODOLOGY

In this work, the open-access flight simulator FlightGear is used for collecting the learning data and testing the trained NN [25]. Special software has been developed, where landing is programmed in the specified flight simulator at Sheremetyevo airport on the runway with the code 06R/24L. The runway characteristics are as follows: length 3700 m, width 60 m, heading 75° [26]. During each landing, the screen and all current navigation characteristics were recorded. The output were videos and navigation log files linked to frames.

At the next stage, the collected materials need to be processed, labeled, and prepared for NN learning. It takes too long to do this manually, so it became necessary to automate the process. For this purpose, additional software has been developed; it splits the video into frames with a set frequency and performs labeling. It is based on the calculation of the screen coordinates of runway corners using homography:

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = H \cdot \begin{bmatrix} P_X - X \\ P_Y - Y \\ P_Z - Z \end{bmatrix}; \; H = D_{\mathrm{S}}^{\mathrm{C}} D_{\mathrm{C}}^{\mathrm{fix}} D_{\mathrm{fix}}^{\mathrm{Run}}, \qquad (15)$$

where p_x, p_y are the screen coordinates of corner point; P_X, P_Y, P_Z are the coordinates of corner point in the runway CF; X, Y, Z are the coordinates of the survey point in the runway CF; $D_{\mathrm{fix}}^{\mathrm{Run}} = D_{\mathrm{fix}}^{\mathrm{NECF}}(\gamma, \psi, \mathrm{J}) \cdot D_{\mathrm{NECF}}^{\mathrm{Run}}(\psi_{\mathrm{Run}})$ is the matrix of rotation from the runway CF to the fixed CF (NECF is the Normal Earth Coordinate Frame, $\psi_{\mathrm{Run}} = 75^{\circ}$).

Orientations of the mentioned coordinate frames are shown in Fig. 10.

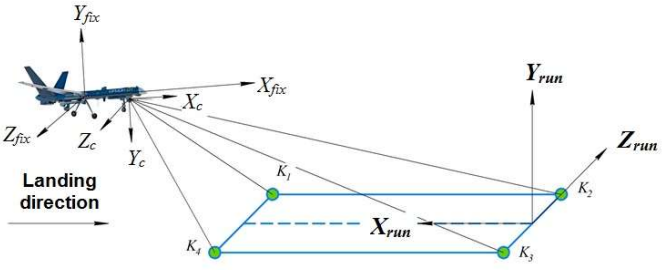

Fig. 10. Orientations of the CF used.

The calculated screen coordinates of the corner points are converted to text for labeling in YOLO format [27]. Thus, the developed software for automatic labeling made it possible to create a dataset of about 3000 images of runway at different distances and at different times of the day (morning, afternoon, evening). The NN learning was performed in the Google Colab cloud environment [28] using the Ultralytics framework [29]. Since the algorithms being developed are supposed to be used in UAV *on-board equipment*, which limits the recognition time, the YOLOv8n architecture with a small number of parameters was used for learning. It was trained to solve the three tasks described above. Some of the resulting parameters and metrics are presented in Table 1.

Table 1. YOLOv8n learning results

Task	Resolution	mAP 50	mAP 50–95
Detection	288×288	0.67	0.6
Segmentation	384×384	0.48	0.35
Pose estimation	416×416	0.71	0.64

The mAP (mean Average Precision) metric is a generally accepted metric for assessing the quality of NN learning [30]. It is measured in the range from 0 to 1 and shows how accurate the NN is on average on a certain dataset at different operating thresholds, which means the degree of overlap between the found and reference objects in the image, expressed as a percentage (mAP 50 – for the threshold of 50%, mAP 50-95 – averaging for the threshold range of 50-95%). A detailed mAP calculation is described in [31]. As can be seen in Table 1, segmentation is the worst task to be learned (each task was trained on the same dataset for an equal number of epochs).

The original resolution of the training images was 1920×1080. Then they were compressed to the resolutions indicated in Table 1. The compression resolution was chosen so that the total operating time of the algorithms, including post-processing, was no more than 100 ms, which is the period of the SNS data update (it is assumed that the navigation algorithm based on runway recognition should have a comparable speed to replace that system). With increased resolution, the recognition accuracy is higher, while the inference time is slower. The performance of the developed algorithms is discussed in the next section.

6. SIMULATION AND DISCUSSION OF RESULTS

We have studied three runway recognition algorithms using a NN. Figure 11 shows a flowchart of their operation, including the selection of a specific algorithm.

The fitness function (14) is calculated for segmentation and pose estimation, while for detection it will be obtained at the stage of GA completion. If the fitness function reaches a value of ≥ 0.9, the found corner points are tracked in subsequent frames using the Lucas–Kanade optical flow estimation method [32]. This helps to remove the random noise of the recognized contour, as well as extrapolate the exact solution to subsequent frames. If a solution with a high *PF* has been found again, then the tracked points are updated.

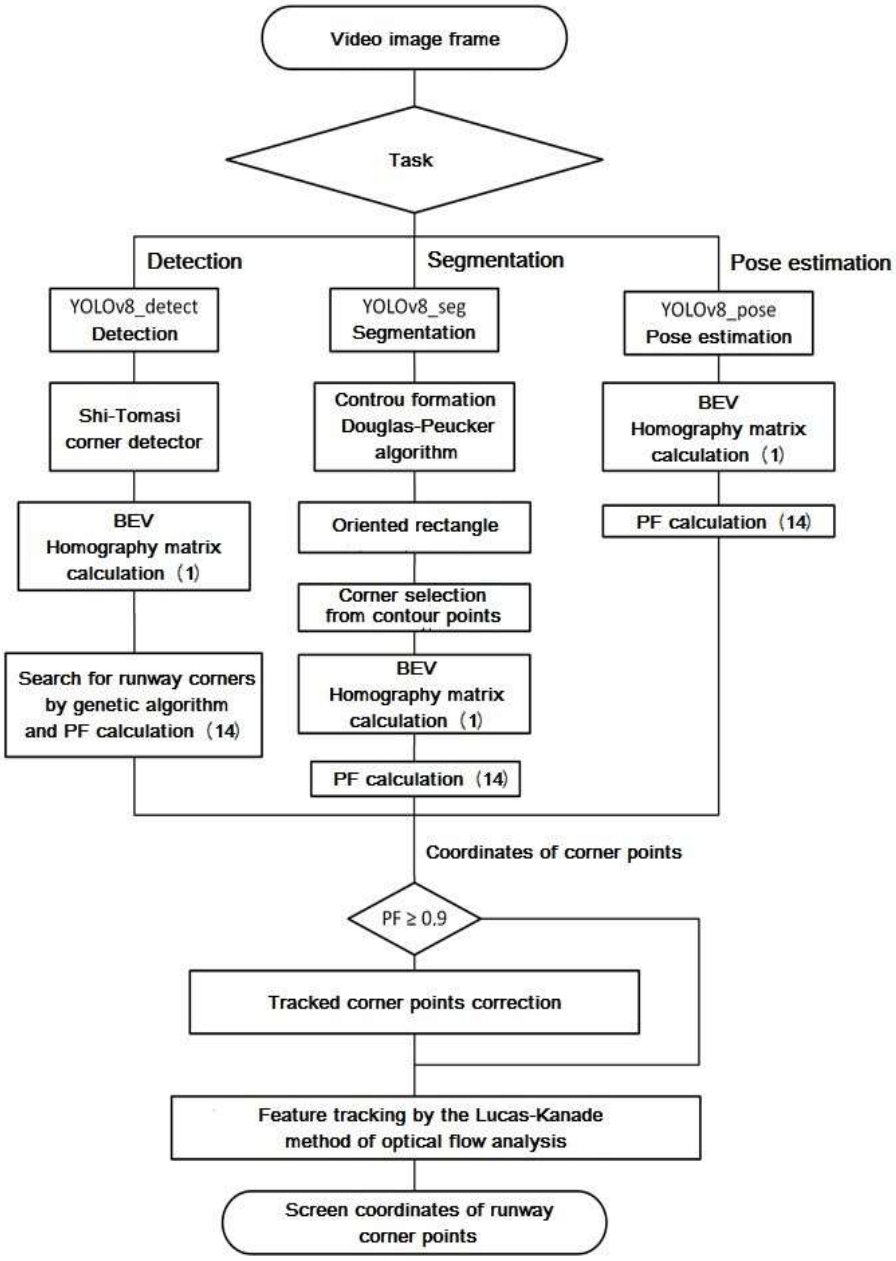

Fig. 11. Flowchart of runway recognition algorithms.

All three runway recognition algorithms were assessed using previously recorded video materials in the simulations with an embedded Jetson Nano single-board computer [32]. This device supports the graphics cores which accelerate the work of NN due to parallel computations. Thanks to its small size, Jetson Nano can be used on board UAVs.

To verify the developed algorithms, three critical parameters were estimated: speed, recognition accuracy, and recognition distance. The NN inference is the most time-consuming operation in the created

algorithms. In this paper, it is implemented using ONNX Runtime [34] for the inference on the central processing unit (CPU), and also using NVIDIA TensorRT [35] for the inference on the graphics processing unit (GPU). The accuracy was estimated using the IoU (Intersection over Union) metric [36]. It reflects the degree of overlap of the recognized object relative to the reference one and is expressed as a percentage. The distance was measured relative to the near threshold and was fixed at the moment of stable recognition start ($PF \geq 0.9$). The graph of the IoU change depending on the distance to the threshold is shown in Fig. 12.

The average results of all indicators obtained during the experiment are summarized in Table 2.

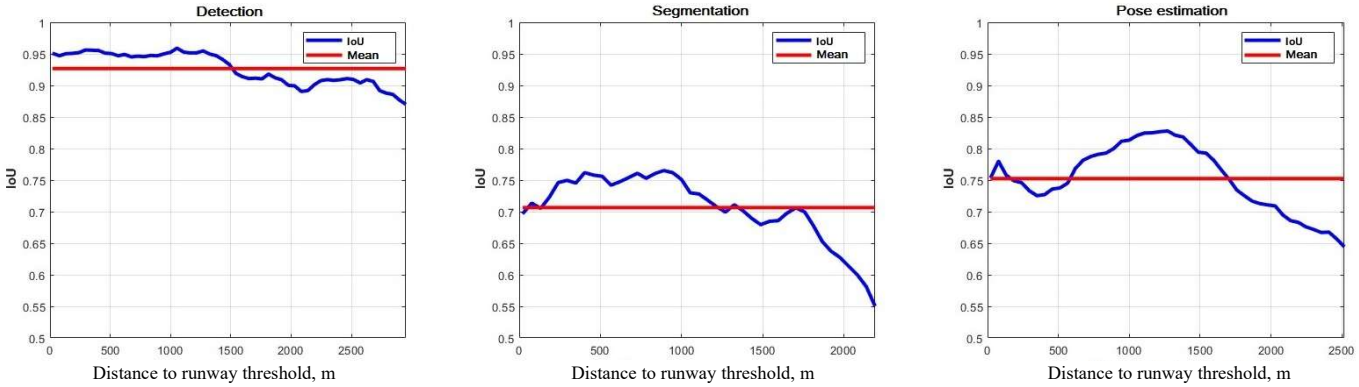

Fig. 12. Dependence of IoU on the distance to runway threshold

Table 2. Comparison of algorithms

Task	Resolution	Inference, ms		Total time, ms		IoU	Distance, m
		CPU	GPU	CPU	GPU		
Detection	288	144	29	177	79	0.92	2950
Segmentation	384	402	78	408	81	0.71	2150
Pose estimation	416	324	70	324	70	0.75	2500

7. CONCLUSIONS

The paper is devoted to the choice of algorithms for recognizing the runways in a video image. A review of publications in this field has been conducted and it has been shown that there are no solutions with a recognition distance significantly exceeding D_{min} = 1430 m (calculated based on the minimum height of establishing visual contact with the runway). Three variants of the YOLOv8 NN trained for the following operating modes have been analyzed:

1) detection;
2) segmentation;
3) pose estimation.

To collect learning data and generate a dataset, software has been developed that allows for automatic labeling of video frames, taking into account the task being solved by the NN. To verify the developed algorithms, software-based simulation was performed using the FlightGear flight simulator and the Jetson Nano computing module. The following results were obtained during the simulation:

1) the segmentation-based algorithm showed the lowest accuracy and distance of recognition;

2) the algorithm based on pose estimation showed a medium distance of recognition, but was comparable to algorithm 1 in terms of accuracy;

3) the detection-based algorithm showed the highest accuracy and distance of recognition, as well as acceptable performance speed.

Thus, according to the totality of values of the estimated parameters, the detection-based recognition algorithm showed the best result. The recognition distance of all developed algorithms exceeded D_{min} by more than 50% and amounted to at least 2100 m. This indicates the expediency of the NN use in the task of runway recognition instead of classical computer vision methods. The achieved performance speed is comparable to the performance of standard SNS receivers (100 ms (10 Hz)), which indicates that the proposed algorithms can be used in embedded systems in real conditions.

FUNDING

This work was supported by ongoing institutional funding. No additional grants to carry out or direct this particular research were obtained.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. Mallick, S., Head pose estimation using OpenCV and Dlib. URL: https://learnopencv.com/head-pose-estimation-using-opencv-and-dlib/. Cited: October 12, 2023.
2. Ali, B., Sadekov, R.N., and Tsodokova, V.V., A review of navigation algorithms for unmanned aerial vehicles based on computer vision systems, *Gyroscopy and Navigation*, 2022, vol. 13, no. 4, pp. 241–252. https://doi.org/10.1134/S2075108722040022
3. Vovk, V.I., Lipin, A.V., and Saraiskii, Yu.N., *Zonal'naya navigatsiya* (Zonal Navigation), Textbook, 2nd ed., St, Petersburg: Tsentr avtomatizirovannogo obucheniya, 2004.
4. Krasnopevtsev, B.V., *Fotogrammetriya* (Photogrammetry), Moscow: UPP "Reprografiya" MIIGAiK, 2008.
5. Jingnan Shi, Perspective-n-Point: P3P. URL: https://jingnanshi.com/blog/pnp_minimal.html. Cited: October 12, 2023.
6. Bondarev, V.G., Lopatkin, D.V., and Smirnov, D.A., Automatic landing of aircraft, *Vestnik VGU, System Analysis and Information Technologies Series*, 2018, no. 2, pp. 44–51.
7. Burns, W.R., A vision-based algorithm for UAV state estimation during vehicle recovery, *Master Thesis*, University of Kansas, 2011.
8. Hiba, A., Gáti, A., and Manecy, A., Optical navigation sensor for runway relative positioning of aircraft during final approach, *Sensors*, 2021, vol. 21, p. 2203, https://doi.org/10.3390/s21062203.
9. Singh, S., How computer vision based ATTOL system helps aircrafts in landing & takeoff. URL: https://www.labellerr.com/blog/how-computer-vision-based-attol-system-helps-air-crafts-in-landing-takeoff/. Cited: May 22, 2023.
10. Airbus concludes ATTOL with fully autonomous flight tests. URL: https://www.airbus.com/en/newsroom/press-releases/2020-06-airbus-concludes-attol-with-fully-autonomous-flight-tests. Cited: May 22, 2023.
11. Scherer, S., Mishra, C., and Holzapfel, F., Extension of the capabilities of an automatic landing system with procedures motivated by visual-flight-rules, *Proc.33rd Congress of the International Council of the Aeronautical Sciences* (ICAS), Stockholm, Sweden, 4–9 September 2022.
12. Kugler, M. E., Mumm, N. C., Holzapfel, F., Schwithal, A., and Angermann, M., Vision-augmented automatic landing of a general aviation fly-by-wire demonstrator, *Proc. AIAA Scitech 2019 Forum*, American Institute of Aeronautics and Astronautics, 2019. https://doi.org/10.2514/6.2019-1641
13. Hecker, P., Angermann, M., Bestmann, U., Dekiert, A., and Wolkow, S., Optical aircraft positioning for monitoring the integrated navigation system during landing approach, *Gyroscopy and Navigation*, 2019, vol. 10, no. 4, pp. 216–230.
14. Herold, H., *Air Law*, CAE Oxford Aviation Academy (UK), 2014.
15. Radio Navigation Aids, International Standard AN 10-1, ICAO, 2018.
16. Ultralytics YOLOv8. URL: https://docs.ultralytics.com/ru/models/yolov8/#key-features. Cited: August 16, 2024.
17. YOLOv8. URL: https://docs.ultralytics.com/ru/models/yolov8/. Cited: August 16, 2024.
18. De Koning, E., Douglas-Peucker. URL: https://psimpl.sourceforge.net/douglas-peucker.html. Cited: September 10, 2024.
19. Shi-Tomasi Corner Detector & Good Features to Track. URL: https://docs.opencv.org/4.x/d4/d8c/tutorial_py_shi_tomasi.html. Cited: October 01, 2024.
20. RidgeRun's Birds Eye View project research. URL: https://developer.ridgerun.com/wiki/index.php/Birds_Eye_View/Introduction/Research. Cited: October 01, 2024.
21. Basic concepts of the homography explained with code. URL: https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html. Cited: October 01, 2024.
22. Lanskii, A., The three basic metrics that will be useful in Data Science: Euklidian, L1 and Chebyshev distances. URL: https://tproger.ru/translations/3-basic-distances-in-data-science. Cited: October 01, 2024.
23. Semenov, S.S., Voronov, E.M., Poltavskii, A.V., and Kryanev, A.V., *Metody I modeli prinyatiya reshenii v zadachakh otsenki kachestva i tekhnicheskogo urovnya slozhnykh tekhnicheskikh sistem* (Methods and Models for Making Decisions in the Problems of Quality and Engineering Level Assessment of Complex Technical Systems), 2nd ed., Moscow: Lenand, 2019.
24. Panchenko, T.V., *Geneticheskie algoritmy* (Genetic Algorithms), Astrakhan University Publisher, 2007.
25. Introduction to FlightGear. URL: https://www.flightgear.org/about/. Cited: October 01, 2024.
26. Aeronautical Information Publications (In-Flight), Aeronautical Information Center. URL: http://www.caiga.ru/sborniks/. Cited: February 06, 2024.
27. Object Detection Datasets Overview. URL: https://docs.ultralytics.com/datasets/detect/. Cited: October 01, 2024.
28. Welcome To Colab. URL: https://colab.research.google.com/. Cited: October 01, 2024.
29. Ultralytics. URL: https://docs.ultralytics.com/. Cited: October 01, 2024.
30. Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A., The PASCAL visual object classes (VOC) challenge, *International Journal of Computer Vision*, 2010, no. 2, pp. 303–338.
31. Mean Average Precision (mAP) in object detection, URL: https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/. Cited: October 01, 2024.
32. Patel, D., Upadhyay, S., Optical flow measurement using Lucas Kanade method, *International Journal of*

Computer Applications, 2013, no. 10, pp. 6–10. https://doi.org/10.5120/9962-4611
33. NVIDIA Jetson Nano. URL: https://www.nvidia.com/ru-ru/autonomous-machines/embedded-systems/jetson-nano/. Cited: May 23, 2023.
34. ONNX Runtime for Inferencing. URL: https://onnxruntime.ai/inference. Cited: October 01, 2024.
35. NVIDIA TensorRT. URL: https://developer.nvidia.com/tensorrt. Cited: October 01, 2024.
36. Intersection over Union (IoU) for object detection. URL: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. Cited: October 01, 2024.