Intelligent Algorithms for Planning and Performing Flight Missions of an Unmanned Aerial Vehicles Based on Neural Networks

Yu.Yu. Gromov [1], I.N. Ishchuk [1] and B.K. Telnykh [1]

[1] *Tambov State Technical University, 106/5 Sovetskaya Street, Tambov, 392000, Russia*

Abstract
Unmanned aerial vehicles are widely used in various industries, such as aerospace, geodesy, cartography, remote environmental monitoring, and for military purposes. However, currently there are a number of factors that have a negative impact on the successful performance of special tasks of unmanned aerial vehicles due to the fact that currently remote-controlled means based on the «natural intellect» of the operator prevail in unmanned aerial vehicles and the lack of transition to the most autonomous actions, such as route planning, bypassing flight blocking zones, forecasting navigation situation, as well as search, recognition and classification of remote monitoring objects. To effectively perform tasks, it is necessary to develop intelligent algorithms to automate the process of planning and executing mission tasks. The article discusses the basic principles of neural networks and their application in the tasks of route planning and flight control for remote monitoring of an unmanned aircraft.

Keywords
Remote monitoring, reinforcement learning, neural network, unmanned aerial vehicle.

1. Introduction

The experience of using complexes with unmanned aerial vehicles (UAVs) has shown that one of the main reasons for the malfunction of the UAV and its loss is that at present, UAVs are dominated by remotely controlled means based on the "natural intelligence" of the operator and the lack of transition to highly autonomous actions, such as route planning, bypassing prohibited flight zones, as well as searching, recognizing and classifying objects of remote monitoring (ORM). Modern technologies and computing power have made it possible to apply non-standard ways to solving navigation problems. It has become possible to take into account factors and conditions that increase the accuracy of planning the route of the UAV flight. In addition, automated systems for processing remote monitoring data do not always ensure accuracy and efficiency. To solve these tasks, it is necessary to use artificial neural networks, which have a number of advantages, such as working with large amounts of data, high speed and autonomy. Neural networks can effectively solve the problems of constructing flight routes taking into account prohibited zones, recognizing objects in the visible and infrared (IR) wavelength ranges and ensuring high adaptability and self-learning.

2. Statement of the problem

UAV flies from the airfield to a remote monitoring area equipped with a multispectral optoelectronic system in a simulation environment. Creating an environment for training an unmanned aerial vehicle is a complex task that includes aspects of both computer vision and flight control. To create such an environment, we use a standardized environment to create and train OpenAI Gym intelligent agents, which will allow us to create a learning environment for planning the flight route of the UAV.

GraphiCon 2023: 33rd International Conference on Computer Graphics and Vision, September 19-21, 2023
V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russia
EMAIL: gromovtambov@yandex.ru (Yu.Yu. Gromov); boerby76@mail.ru (I.N. Ishchuk); bogdanfm0508@yandex.ru (B.K. Telnykh)
ORCID: 0000-0003-3313-2731 (Y.Y.Gromov); 0000-0002-8140-3746 (I.N. Ishchuk); 0009-0001-1870-3738 (B.K. Telnykh)

Let there be a certain simulation environment with a set of states S_n and an intelligent Deep Q-Network agent for the UAV capable of performing certain actions a_n n this environment and receiving a particular reward $r(S,a)$ for the action performed. The agent chooses the direction of movement based on a strategy or policy:

$$a = \pi(S), \tag{1}$$

where $\pi(S)$ – the function that selects the optimal action for the current state.

To optimize the direction selection, a discount factor γ is introduced, which reduces the importance of rewards for previous actions if the agent makes many actions. Thus, we create a penalty system that encourages the agent to receive rewards in the current step. Now the task of finding the optimal policy can be expressed by the following equation:

$$\pi = \arg\max \sum_{n \ge 0} \gamma^n r_n, \quad \gamma \in (0;1], \tag{2}$$

where r_n– the next discounted reward.

To estimate the future state model, an estimation function is introduced that estimates the total discounted reward for the state-action pair $Q(S,a)$. However, to determine the values of future values Q' t is necessary to use the Bellman optimality principle, which states that the maximum future reward from an action a is the current reward plus the maximum future on the next step from performing the next action a' [1]. Based on this, the function $Q(S,a)$ can be represented by the following expression:

$$Q(S,a) = M[r + \gamma \max_{a'} Q'(S',a')], \tag{3}$$

where M – the mathematical expectation of the sum of the current reward and the discounted accumulated reward.

In this way, the task of reinforcement learning becomes finding the optimal values of the weights of the function Q, so as to maximize the future total reward, as well as selecting the optimal action based on the current policy.

To solve the problem of searching and recognizing ORM from aerial photography taken in the visible and IR wavelength range in an automatic mode, we will use supervised machine learning methods.

Let us have a set of ORM on IR images with different shooting times X, each of them is given by a vector of features $\vec{x}$. There is a training set $X_{train} \subset X$ of L objects with a given target feature y. It is necessary to determine the quality functional $L(a, \vec{x}, y)$, which characterizes how well the approximation algorithm a predicts the values of the target X. Next, the parametric model $a(\vec{x}, \vec{\theta})$ is selected. The parameters of the model $\vec{\theta}$ must be optimized to achieve a minimum L.

The machine learning task is to extrapolate data from X_{train} to X, in order to improve the generalization capability of the recognition algorithm.

As a feature for ORM classes, a classifier for typical ORM is proposed based on thermophysical properties of materials, including the thermal activity of the material (thermal inertia) I [2].

Objects (backgrounds) are assigned an evaluation of the thermophysical parameter value or coefficient of thermal activity (thermal inertia I), and, based on a normalization scale, the corresponding type of material is determined, indicating the possible types of material [3].

3. The algorithm for planning the route of an unmanned aerial vehicle based on deep reinforcement learning

In the traditional approach to reinforcement learning, the Q-function is represented as a table with the number of states of the environment and the number of actions that the agent can take. However, in deep reinforcement learning (DQN), a neural network is used to approximate the Q-function. This is because in tabular form, the Q function would occupy enormous amounts of memory, and it would be impossible to use such an approach for more complex problems.

The deep reinforcement learning algorithm for planning an unmanned aerial vehicle's route consists of the following steps.

Step 1. Creation of a simulation area of the UAV as a tactical situation map 20x20 with an environment and divided into several computational areas represented by equal cells, each corresponding to one agent move. An intelligent agent, acting as the UAV, can freely move across the entire environment in four directions. Prohibited flight areas are highlighted on the map.

Step 2. The definition of hyperparameters of training. At this stage, values are assigned to training parameters that directly affect the modeling process itself, such as the learning rate α (takes values in the range from 0 to 1), the reward discount rate γ, the number of simulation episodes, and the maximum number of iterations in one episode. The parameter γ is necessary so that the UAV uses knowledge obtained in previous episodes to choose the most reliable actions [4]. The limitation on the number of steps in one episode is necessary to prevent the agent from looping in a cycle, although the episode may end earlier. The values of hyperparameters are given in table 1.

Table 1: Training Settings

Training coefficient	**Value**
Learning rate, α	0,1
Discount rate, γ	0,95
Number of training episodes	20000
Maximum number of steps per episode	10000

Step 3: Creation of a neural network model to approximate the optimal *Q*-function. To build the network architecture, TensorFlow and Keras, an object-oriented programming language Python, were used The task of approximating the Q-function can be formulated as a supervised learning problem. Let's Q_{target} target value, which, based on the expression (3), in DQN is presented in the following form:

$$Q_{aim} = r + \gamma \max_{a'} Q_W(S', a') \tag{4}$$

where Q_W – parametric function, which predicts the quality of the pair $Q(s,a)$ and depends on the parameters of the approximator W.

It is necessary to update the parameters Wso, that function $Q_W$approaches to target values Q_{target}We will formulate the loss function *E*, which in DQN will be presented as follows:

$$E = (Q_{target} - Q_W(S,a))^2 \tag{5}$$

Using gradient descent, we minimize the loss function by updating the weights of the approximator, taking into account the parameter α - the learning rate.

Step 4. *Q*-function training. To facilitate observation, initialize variables that accumulate data on trajectory length and cumulative episode reward. To select a direction vector, we use an *ε*–greedy policy. This approach maintains a balance between exploration of the simulation environment and selecting a random action and a direction chosen according to the current policy.

Let the target vector $\overrightarrow{Q(S)}$ be equal to the original values for all actions except the action a_i The task of the algorithm is to train the neural network on this target vector so that it changes its weights so that only the value for the updated, and for the remaining actions not changed. Implementation of such a strategy is carried out by minimizing the error.

According to the algorithm, one minimization step requires making one gradient descent step through the built-in batch gradient descent function (batch size), which implements the ability to perform an evaluation of subsets of the training sample of a fixed size [5]. In the considered algorithm, a subset of the training sample is received at the input with a length of one element. The progress of training the neural network is presented in figures 1 and 2.

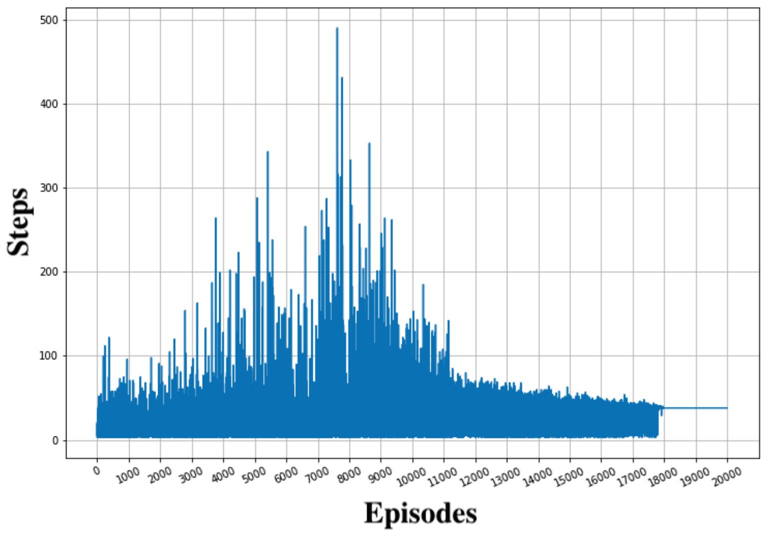


Figure 1: The plot of the flight trajectory lengths of the UAV over episodes during the training of the DQN neural network

The graph of flight trajectory lengths for the UAV shows that the agent explored the environment in early episodes, hit the forbidden zones, exceeded the allowed number of steps per iteration of learning. Subsequently, with a linear decrease in the ε parameter, relying on the updated weights of the target vector $\overrightarrow{Q(S)}$, it began to reach the goal in a fixed number of steps.

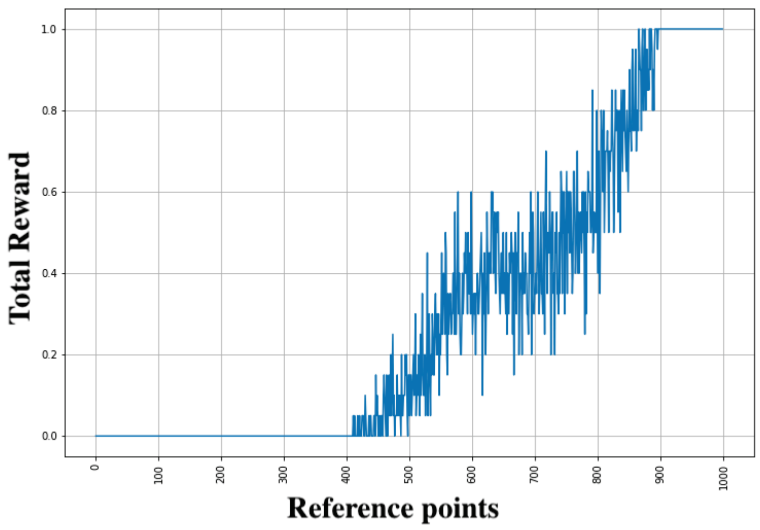
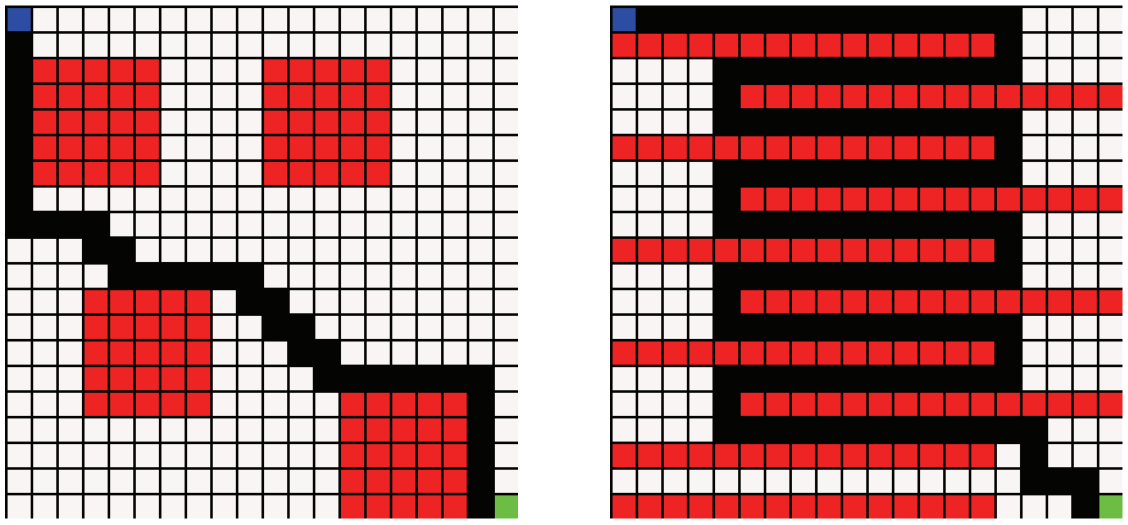


Figure 2: The plot of the total average rewards of the agent when choosing a direction according to the DQN policy

The graph of total average rewards shows a gradual increase in the maximum reward received by the agent. This is due to the chosen strategy and the sequential updating of *Q*-function weights.

As tools for training neural algorithms and conducting numerical simulations, we used the Google Colab interactive cloud platform with an integrated Nvidia Tesla T4 GPU with a memory capacity of 16 GB. This was due to the fact that training neural models is a resource-intensive process that requires significant computing resources.

The size of the control situation map is specified as a 20 by 20 matrix (start point – blue color, target point – green color). The map contains forbidden flight zones for UAVs (red color). A total of 250 control episodes were generated. The probability of not hitting the forbidden zones is 0.97. The training time on the NVIDIA Tesla T4 GPU was 52 minutes. The simulation results are presented in figure 3.



a) The optimal route for the UAV calculated using the DQN neural network; б) The corridor route of the UAV was calculated by the neural network DQN;

Figure 3: Possible routes for the UAV flying generated by the DQN algorithm

It is also important to note that in modeling a complex tactical scenario where the UAV flight trajectory can only follow a corridor, DQN algorithms address this challenge.

The algorithm presented here improves the survivability of UAVs and thus increases their effectiveness in addressing remote monitoring objectives.

4. The algorithm for recognizing objects of remote monitoring based on infrared images using a three-layered Rosenblatt perceptron

For creating an algorithm for detecting ODMs using time-separated infrared images with a UAV, a model of an artificial neural network (ANN), which is based on a three-layer Rosenblatt perceptron, a simplified version of a biological neural network that consists of four layers, has been used: input, two hidden layers and output. The structure of the three-layered Rosenblatt Perceptron-based ANN model used as the model is presented in figure 4.

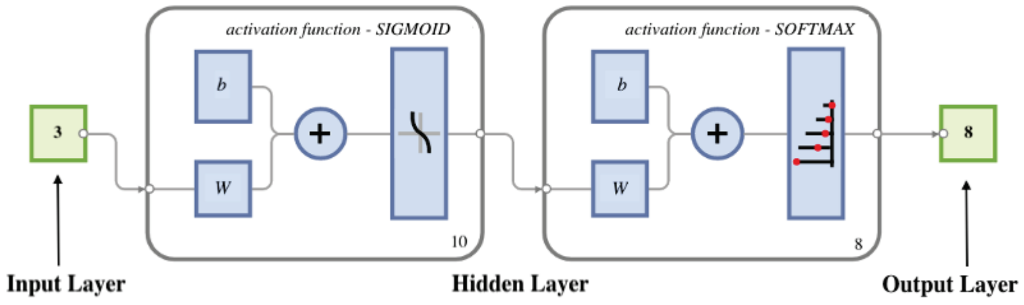


Figure 4: The structure of a three-layer Rosenblatt perceptron

The algorithm for recognizing objects of remote monitoring based on infrared image data using a three-layer perceptron Rosenblatt consists of the following steps:

Step 1. Form a training dataset. The input data for the ANN is time-separated near-infrared and visible images obtained with the UAV during field experiments during the winter period. An image for the period of 12:00 was excluded from processing due to errors associated with the presence of a shadow from the Sun.

Next, a cuboid of successive daytime images of the near-infrared and visible wavelength range, which forms a matrix of input parameters of the ANN, was formed.

The next step in forming the training dataset is to mark reference objects of the same size for 8 classes. Each class was assigned its own color to improve the visualization of the algorithm's operation.

Step 2. Train the neural network. To minimize the overfitting of the ANN algorithm, it is necessary to divide the training dataset in the following proportions:

X_{train} – training dataset – 70%;

X_{val} – validation dataset – 15%;

X_{test} – testing dataset – 15%.

The training will stop after a certain number of epochs or when a specified error threshold is reached on the validation dataset [6].

To assess the quality of the algorithm training, the values of the cross-entropy $H(p,q) \to \min$ function are highlighted. The plot of the change in the quality functional during the algorithm training process is presented in figure 5.

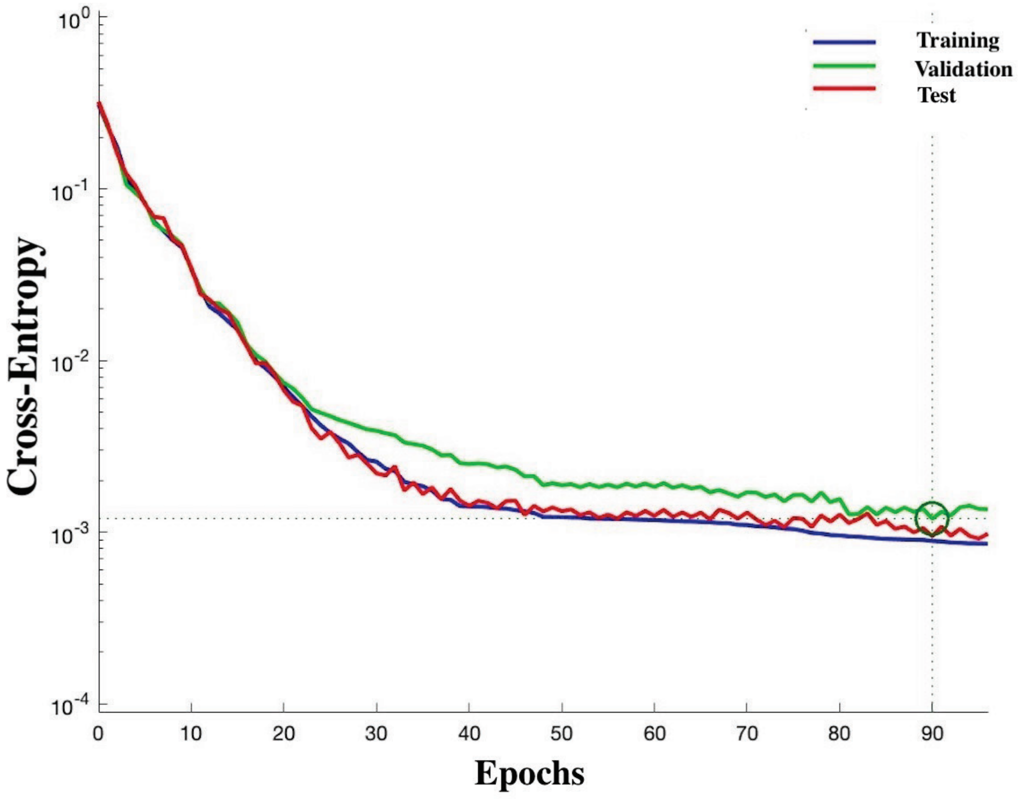
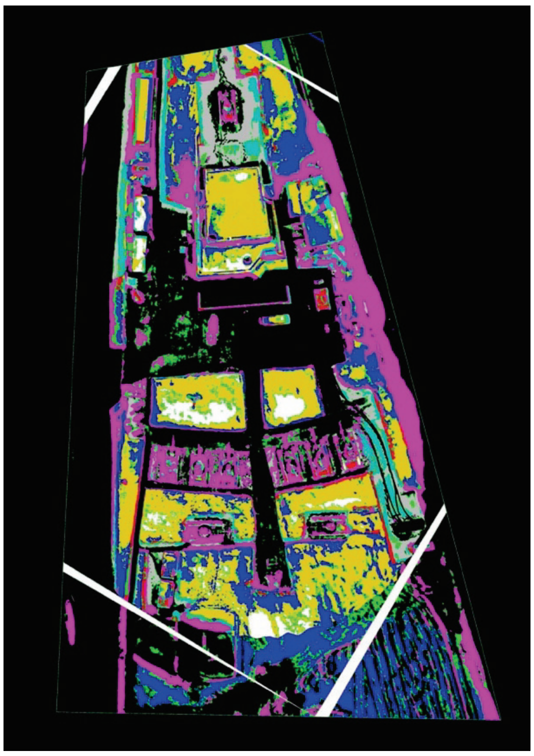


Figure 5: Graph of the quality functional by training epochs

Step 3 Check the operability of the algorithm The operability of the algorithm was checked by uploading a real image, a cube of orthophotomap, in the infrared wavelength range. The algorithm's output is shown in figure 6.



Classes of materials:

1) Sidewalk (thick pavement)– ⬛
2) Sidewalk (thin coating)– ▨
3) Snow cover (untouched snow) – ⬜
4) Snow cover (hard snow on concrete) – 🟨
5) Snow cover (melted snow on concrete) – 🟦
6) Metal (thick layer of metal) – 🟥.
7) Metal (metal layer / internal heat source) – 🟪
8) Plastic box– 🟦
9) Unclassified object – 🟩

Figure 6: Orthophotomap of the RM area processed by the neural network algorithm for recognizing ORMs

The analysis of the results indicates that the algorithm provides an acceptable level of accuracy in classifying the given objects while identifying objects of particular interest.

According to the results of the functionality test of the algorithm, one can conclude that it successfully identifies objects in IR images and could be used for processing remote sensing data in real-time mode.

5. Conclusion

Developed intelligent algorithms for planning and executing flight tasks for UAVs based on neural networks allow for an increase in the efficiency of UAV-based complex systems for solving remote sensing problems through increased flight safety by preventing UAVs from entering prohibited flight zones and increasing the accuracy of processing information received from optical-electronic systems. Implementing algorithms as an onboard intelligent support system ensures the maximum autonomy of a UAV during flight tasks.

6. References

[1] R.S. Sutton, A.G. Barto Reinforcement Learning. MIT Press, Cambridge, MA, 1998. A Bradford Book.
[2] A.V. Lykov Theory of thermal conductivity. Moscow: Higher School, 1967.
[3] K.F. Fokin Building heat engineering of building envelopes. Moscow: Stroyizdat, 1973.
[4] Ch. Yan, X. Xiang, Ch. Wang: Towards Real-Time Path Planning through Deep Reinforcement Learning for a UAV in Dynamic Environments. Journal of Intelligent & Robotic Systems, 2019 URL: https://doi.org/10.1007/s10846-019-01073-3.
[5] Keras API docs: official site. URL: https://keras.io/api.
[6] Mathlab Documentation: MathWorks official site. URL: https://www.mathworks.com/help/matlab.