Developing an unmanned vehicle local trajectory using a reinforcement learning algorithm

T S Evdokimova\textsuperscript{1,3}, A A Sinodkin\textsuperscript{1}, M I Tiurikov\textsuperscript{1}, L O Fedosova\textsuperscript{1} and A V Kalyashina\textsuperscript{2}

\textsuperscript{1} Nizhny Novgorod State Technical University n.a. R.E. Alekseev, Nizhny Novgorod, Russian Federation
\textsuperscript{2} Department of Laser Technologies of KNRTU KAI

E-mail: \textsuperscript{3}evdokimovats97@gmail.com

Abstract. This article describes the algorithm development for constructing a local trajectory for an unmanned vehicle or for implementation in an ADAS system using the reinforcement learning method. A special part is dedicated to reinforcement learning. One of the methods that is best suitable for the task conditions will also be implemented. This method will allow bypassing obstacles and reaching the specified short target points.

1. Introduction
Currently, one of the promising areas in the robotics field is the unmanned robots development and in particular - vehicles. The purpose of developing unmanned transport systems is to prevent accidents. Road users’ protection from unforeseen situations and the human factor minimization is of particular importance. One of the main functional purposes of active driver assistance systems is to prevent and avoid an emergency, both with static and dynamic obstacles. At the moment, in addition to completely unmanned vehicles, driver assistance systems (ADAS) are being created. Such assistance systems are aimed at predicting possible unforeseen situations for the driver while driving a vehicle. Apart from notifying the driver of an existing danger, this system involves intervention in the vehicle control, applying engine control algorithms, braking system and steering.

One of the most important stages in the unmanned vehicle development is constructing a route. Path developing algorithms require up-to-date and timely environment data in which the unmanned vehicle will be located. The route is divided into global and local. A global route is a trajectory from a starting point to a target. The local Planner is a separate fragment of the global planner path that allows achieving short goals, taking into account the presence of both static and dynamic obstacles.

2. Hardware equipment
To implement the local trajectory development the vehicle was equipped with a certain set of information measuring devices. Depending on the conditions in which the solution of the task will take place and taking into consideration the hardware equipment, a set of information measuring devices was selected. At the same time, it is important to take into account the systems expediency of the systems used. Since the tasks of developing local trajectories for unmanned vehicles are considered, the most accurate data on the mobile platform movement is needed. Therefore, reliable
and accurate rangefinders are required to determine the map of obstacles, and to determine the movement in space, only those measuring devices that will determine the direction and location relative to themselves at different points in time should be considered, and thus it one should realise that odometric, inertial data are suitable.

The following equipment was used to develop the local trajectory:

- Ultrasonic sensor HC-SR04;
- NVIDIA Jetson Nano Developer Kit;
- Laser rangefinder URG-04LX-UG01;
- Controller NUCLEO-F031K6;
- Sensor with inertial system BNO055 Xplained Pro.

3. Implementation of a local trajectory development for an unmanned vehicle

To implement the local trajectories development it was necessary to configure the data exchange between the Unity simulation environment and the training environment and to configure the processing of various data coming from the simulation environment for the subsequent application of this data to learning through the Asynchronous Actor-Critical Agents algorithm (Figure 1).

![Figure 1. Algorithm for the development sequence for building a local trajectory](image)

3.1 Receiving and exchanging data with the Unity simulation environment

The general data exchange algorithm between the simulation environment and the learning environment (Figure 2). The algorithm has two components - the simulation environment and the training environment. They interact in such a way that data about the world in which the model exists (obstacle map, data on the position, direction and the simulated robotic platform speed) comes from the simulation environment. The learning environment, in turn, conveys data about the actions that the platform should take in in the simulated environment. Interaction with the simulation environment is done through the ZeroMQ messaging library. ZeroMQ is a high-performance asynchronous messaging library designed for use in distributed or parallel applications.
3.2 Data preparation required for training a neural network
To implement neural network training for developing an unmanned vehicle local trajectories, it is necessary to use more than one type of input data, but to take into account all the factors and parameters necessary for the neural network correct training.

Implementation requires the following data and parameters:
- Cost card;
- Position (x);
- Position (y);
- Movement speed (υ);
- Wheels rotation angle (φ);
- Current orientation relative to the global system;

To process the cost map, a convolutional neural network is used, the rest of the data represents the kinematic state of the vehicle (Figure 3) at the current moment and a multilayer perceptron is used for these parameters.

One of the main stages is the cost map creation (Figure 4), since it is one of the leading parameters designed to solve the local trajectory planning problem.
Initially, the type in which data from measuring devices will be sent for processing was determined. When using a laser rangefinder and the ultrasonic sensors, a two-dimensional obstacle map, subsequently processed, is obtained.

The next step will be inflation (increase) of obstacles on the input map $M$, this is necessary so that the unmanned vehicle can stop at some distance from the obstacles. This stage is implemented by means of the Open CV library [3] (computer vision library as well as image processing), since, in fact, an image as an input is received.

This map is then presented as a cost map. The first is to fill an empty array with heuristic values. These costs are formed taking into account the distance from the center to the specified segment (cell). This is necessary so that in the future the path cost for the goal can be calculated.

Next is the input map $M$ processing; it is necessary to apply a Gaussian filter (blur filter), which is most often used when working with convolution matrices. This step is also carried out using the Open CV library and is necessary to determine the cost for cells near obstacles. Then a summary map of costs consisting of a heuristic costs map and a map with determining the cost near obstacles is
calculated. And the final step in processing the value map is the final map calculation with the calculation of the maximum values.

And in the end, applying all the data, it is necessary to train the model applying one of the reinforcement learning methods - AACA.

To implement this algorithm, it is necessary to use two neural networks, but at the same time, each of them will use the functions and improvements that are in other reinforcement learning algorithms. AACA algorithm (Figure 5) combines the best parts of both methods, i.e. the algorithm predicts both the cost function \( V(s) \) and the optimal policy function \( \pi(s) \) (s-state). The training agent uses the Value (Critic) to update the optimal policy (Actor) function. It should be noted here that in this case the policy function means the probabilistic distribution of the action space. The training agent determines the conditional probability \( P(a \mid s) \) i.e. the parameterized probability that the agent will choose the action \( a \) (action) in the current state \( s \).

In AACA, there is a global network and several working agents, each with its own set of network parameters. Each of these agents interacts with its own copy of the environment at the same time as other agents interact with their environment. The reason of its performing better than neural networks with a single agent (besides making more work faster) is that each agent's experience is independent of the experience of others. Thus, the overall experience available for learning becomes more diverse.

Understanding the use of benefit estimates, not just discounted rewards (formula 1.1 where \( \gamma(r) \) is a discount factor indicating to the agent which of its actions were rewarded and which were penalized) allows agents to determine not only how good its actions were, but also how much better they were than expected (formula 1.2 where \( Q(s, a) \) is the value of action in state \( s \), \( V(s) \) shows how good the current state is without considering actions). Intuitively, this allows the algorithm to focus on where network predictions were missing.

Total reward: \( R = \gamma(r) \) (1.1)

Advantage: \( A = Q(s, a) - V(s) \) (1.2)

![Figure 5. Actor Critic method Algorithm](image)

The ResNet18 convolutional neural network is used to process the map and a multilayer perceptron is used to process the kinematic parameters. At the output, two vectors are obtained and then they are
combined into one. Subsequently, after the AACA algorithm two values are obtained: state estimation and action estimation.

**Figure 6.** Actor Critic algorithm diagram in terms of solving the trajectory planning problem

4. Conclusion
As a result of the work, a reinforcement learning algorithm was developed. It showed that for the most correct operation and the algorithm learnability, it is necessary to put the system in a large number of situations. The advantages are that these algorithms have the ability to learn and they will allow considering more contingencies in the future and solve them.

A number of problems remain that can be identified as further directions of the algorithm development. One of the problems is that this system cannot be fully implemented in any vehicle, but if to train the system on one model, the entire series of these models can be implemented with the developed system. There are also requirements for increasing the operation speed and learning of algorithms, as well as the use of more complex algorithms for developing local trajectories for the comparative function. The research this direction will be continued. Tests were carried out in a simulation environment; the next step will be to introduce this system into an unmanned vehicle prototype.

5 References
[1] Dalal N and Triggs B 2005 Histograms of oriented gradients for human detection *CVPR* pp 886–893
[2] Convolutional neural network URL: https://intellect.ml/svertochnayanejronnaya-set-convolutional-neural-network-cnn-6013 (application date 30.05.2018)
[3] Bradski G and Kaehler A 2008 *Learning OpenCV* O’Reilly Media p 580
[4] Rosten E and Drummond T 2006 Machine learning for high-speed corner detection *European Conference on Computer Vision* 1 pp 430–443
[5] Konda V R and Tsitsiklis J N 1999 Actor-Critic Algorithms *NIPS* 13 pp 1008-1014
[6] Bhatnagar S. et al. 2009 Natural actor-critic algorithms *Automatica* 45 11
[7] Shariq Iqbal and Fei Sha 2019 *Actor-Attention-Critic for Multi-Agent Reinforcement Learning*