ISSN 1060-992X, Optical Memory and Neural Networks, 2019, Vol. 28, No. 4, pp. 283–295. © Allerton Press, Inc., 2019.

Object Detection with Deep Neural Networks for Reinforcement Learning in the Task of Autonomous Vehicles Path Planning at the Intersection

D. A. Yudin[a], *, A. Skrynnik[b], A. Krishtopik[a], I. Belkin[a], and A. I. Panov[a], [b], **

[a]*Moscow Institute of Physics and Technology (National Research University), Moscow, 141701 Russia*

[b]*Artificial Intelligence Research Institute, Federal Research Center Computer Science and Control, Russian Academy of Sciences, Moscow, 119333 Russia*

**e-mail: yudin.da@mipt.ru*

***e-mail: panov.ai@mipt.ru*

Received August 29, 2019; revised September 25, 2019; accepted October 2, 2019

Abstract—Among a number of problems in the behavior planning of an unmanned vehicle the central one is movement in difficult areas. In particular, such areas are intersections at which direct interaction with other road agents takes place. In our work, we offer a new approach to train of the intelligent agent that simulates the behavior of an unmanned vehicle, based on the integration of reinforcement learning and computer vision. Using full visual information about the road intersection obtained from aerial photographs, it is studied automatic detection the relative positions of all road agents with various architectures of deep neural networks (YOLOv3, Faster R-CNN, RetinaNet, Cascade R-CNN, Mask R-CNN, Cascade Mask R-CNN). The possibilities of estimation of the vehicle orientation angle based on a convolutional neural network are also investigated. Obtained additional features are used in the modern effective reinforcement learning methods of Soft Actor Critic and Rainbow, which allows to accelerate the convergence of its learning process. To demonstrate the operation of the developed system, an intersection simulator was developed, at which a number of model experiments were carried out.

Keywords: deep neural network, intersection simulator, computer vision, vehicle recognition, reinforcement learning, behavior control

DOI: 10.3103/S1060992X19040118

1. INTRODUCTION

Unmanned transport is an integral part of the new digital infrastructure of the city, the so-called "smart city". In this context, modeling the behavior of a car when it moves on city roads should be considered as modeling the behavior of an autonomous agent, which has access to additional information about the environment besides data from its own sensors.

Consider a model example for unmanned vehicle driving through an intersection. An intersection is a place of increased danger, therefore, in the "smart city" additional tools are being created for intersections to increase awareness of road agents and improve the predictability of taken actions. As such tools, additional video cameras can be used for intersection monitoring. They may be located on the road infrastructure support elements or on the so-called tethered aircraft. The information received from such cameras is transmitted to all road agents in the area of this intersection, playing the role of global data supplementing the own local information of particular unmanned vehicle.

The scientific community is actively researching the task of autonomous vehicle motion planning on the roads. For this, approaches based on deep reinforcement learning [1] and analytical dynamic models of vehicles [2] are actively used.

Most of the papers are devoted to the application of various reinforcement learning approaches for operation with simplified intersection models [3] in various particular conditions [4]. For example, the article [5] considers the model task of learning negotiating behavior between cars in intersections using deep Q-learning. There are also implementations of simulators and platforms for studying safe autonomous vehicle movement at the intersection [6], which however do not have photorealism.

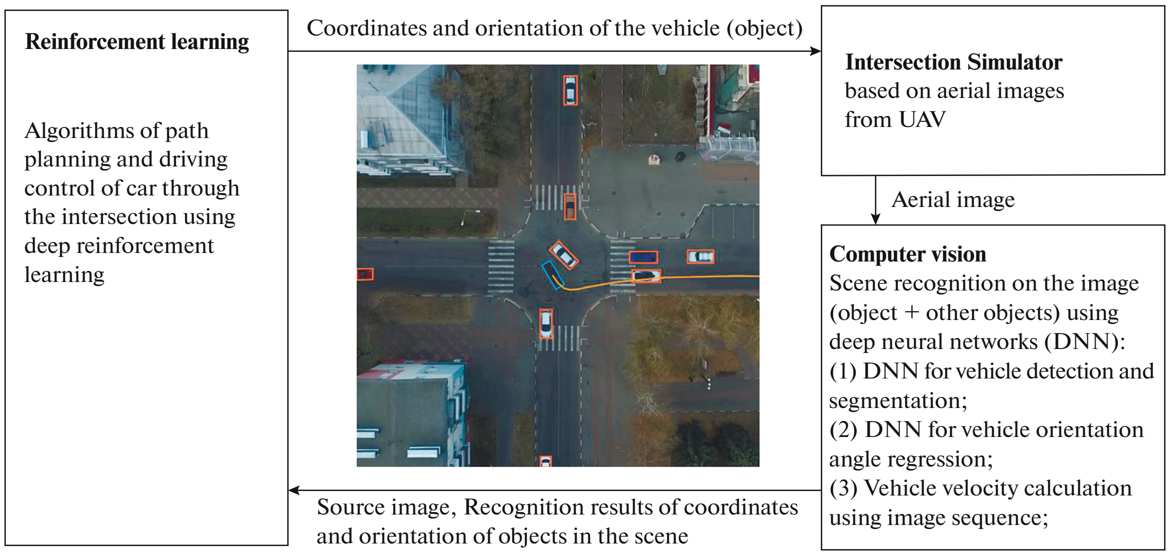

Reinforcement learning
Algorithms of path planning and driving control of car through the intersection using deep reinforcement learning
Coordinates and orientation of the vehicle (object)
Intersection Simulator based on aerial images from UAV
Aerial image
Computer vision
Scene recognition on the image (object + other objects) using deep neural networks (DNN):
(1) DNN for vehicle detection and segmentation;
(2) DNN for vehicle orientation angle regression;
(3) Vehicle velocity calculation using image sequence;
Source image, Recognition results of coordinates and orientation of objects in the scene

Fig. 1. Scheme of the task.

To improve the planning of the agent's trajectory at intersections in the presence of obstacles, additional approaches can be applied. For example, in paper [7] a modification of the potential field method was studied.

A number of modern researchers consider a modular approach to training intelligent agents, which demonstrates high efficiency due to the use of additional information from image recognition modules. For example, research described in [8] shows effective indoor navigation of an intelligent agent with modular approach. Paper [9] considers the usage of deep reinforcement learning for simulation of a vehicle group movement at an intersection using synthetically generated results of road segmentation, obstacles and the desired path.

It should be noted that the results obtained in the course of ongoing research are still far from practical application and are still at the experimental stage.

In this paper, we examined the task of training an intelligent agent, which is a realistic model of an unmanned vehicle, to drive an intersection taking into account other participants and in compliance with traffic rules. When approaching the intersection, the agent connects to the global intersection observation system and, at every moment of its stay in its zone, receives aerial photos of the intersection. In the proposed agent behavior control system we use the integration of computer vision and reinforcement learning methods to form agent movement actions more efficiently. The control system consists of two modules. In the first module, the aerial image is processed, the road agents are selected, in particular other cars, their parameters are evaluated (position, speed, direction of movement). The resulting feature set is transmitted to the second module and used as additional data to the complete information transmitted to the reinforcement learning algorithm. For training, we use one of the modern effective methods that allows us to work with continuous actions Soft Actor Critic and with discrete action space—Rainbow. We have shown that features obtained using computer vision module can accelerate the agent's learning process.

The article also describes the modeling environment that we used to conduct experiments with the developed control system. In the experiments, we presented several levels of integration of computer vision and learning methods with reinforcement, depending on the set of additional features obtained by selecting objects in the image.

2. PROBLEM DEFINITION

The problem to be solved in this work is a complete cycle of movement planning for an intelligent agent—an autonomous vehicle (Fig. 1):

• from obtaining and recognizing a photorealistic aerial photograph of a road scene from a computer intersection simulator using deep neural networks that are most effective in computer vision applications (Computer vision module);

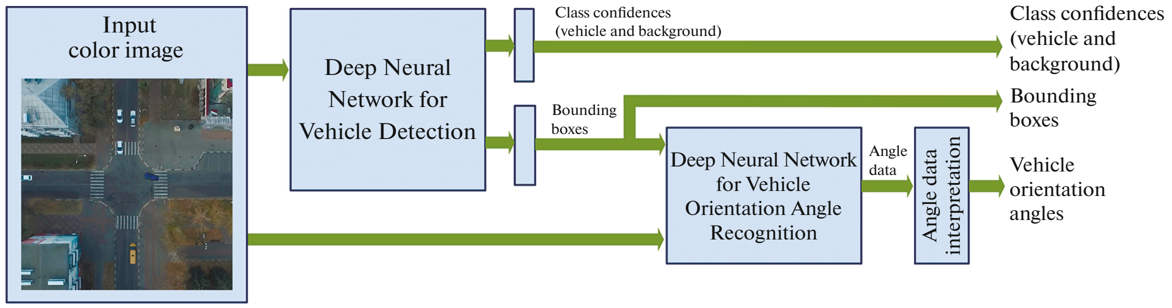

Input color image
Deep Neural Network for Vehicle Detection
Class confidences (vehicle and background)
Bounding boxes
Deep Neural Network for Vehicle Orientation Angle Recognition
Angle data
Angle data interpretation
Class confidences (vehicle and background)
Bounding boxes
Vehicle orientation angles

Fig. 2. Structure of computer vision module.

• to the path planning of the intelligent agent using deep reinforcement learning methods (Reinforcement learning module) and transferring updated coordinates and orientation of the vehicle back to the environment of a computer simulator (Intersection simulator module).

The parameters that the computer vision module should form at the output are

—bounding boxes of vehicles, their detection confidence and their orientation angles, these parameters were decided to be found using the approach based on deep neural networks, shown in Fig. 2;

—speed of vehicles, which must be determined by the relative displacement of the found bounding boxes on the adjacent two frames.

Agent training for the reinforcement learning module is conducted in the standard formulation of the reinforcement learning task. The Markov decision-making process $\langle S, A, T, r\rangle$ with a finite event horizon is considered, where

• S—set of states;

• A—continuous set of actions;

• $T : S \times S \times A \to [0, \infty)$—unknown transition function defining the probability density for the next state $s_{t+1} \in S$ with current state $s_t \in S$ and taken action $a_t \in A$;

• $r : S \times A \to [r_{\min}, r_{\max}]$—reward issued by the environment at every step.

We will denote by $\rho_\pi(s_t)$ and $\rho_\pi(s_t, a_t)$ particular probability distributions of the action sequence of the agent (trajectories), generated strategies $\pi(a_t \mid s_t)$ for state and state-action.

The agent's task will be to generate such trajectories that lead to the maximum total reward based on incoming states and rewards. In our work, we determine the state $s_t = \hat{s}_t \cup \bar{x}$ as a union of global environmental information $\hat{s}_t$ (image of the entire intersection received from the camera) with a vector of additional features $\bar{x}$, which is the result of processing of $\hat{s}_t$ using special computer vision methods to detect and determine the parameters of significant objects and other agents.

3. INTERSECTION SIMULATOR BASED ON AERIAL PHOTOS

The developed intersection simulator is built on the basis of real aerial photographs obtained from the drone's video camera, which were subjected to further processing. Photorealism was one of the important tasks in creating this simulator. The process of preparing data to display them in the simulator is shown in Fig. 3.

At the first stage, we had labeled vehicles with polygons indicating the relative location (left or right), as well as points corresponding to the dividing lines of the lanes. The labeling was carried out using the OpenCV CVAT software tool [10]. In addition, with this tool we marked with polylines the possible trajectories of vehicles at the intersection. This allows us to conveniently change the configuration of intersections and simplifies the preparation of the simulator for reconfiguration to a new scene.

In the process of vehicles cutting, we use algorithms for constructing minimal bounding rectangles [11] and ellipses [12] using vehicle polygonal masks. For the type of scenes under consideration, a more stable result was obtained for bounding ellipses (Fig. 4) and they are used to prepare "zero" position of cropped image with vehicle.

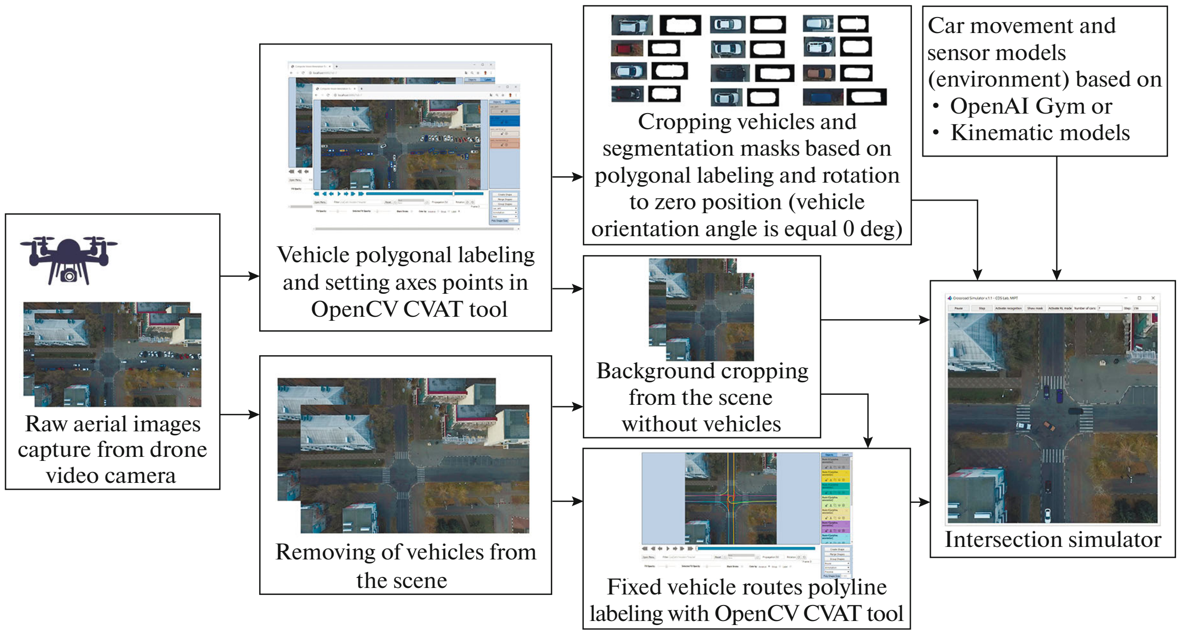

Raw aerial images capture from drone video camera
Vehicle polygonal labeling and setting axes points in OpenCV CVAT tool
Removing of vehicles from the scene
Cropping vehicles and segmentation masks based on polygonal labeling and rotation to zero position (vehicle orientation angle is equal 0 deg)
Background cropping from the scene without vehicles
Fixed vehicle routes polyline labeling with OpenCV CVAT tool
Car movement and sensor models (environment) based on
• OpenAI Gym or
• Kinematic models
Intersection simulator

Fig. 3. Structure of data preparation for proposed intersection simulator.

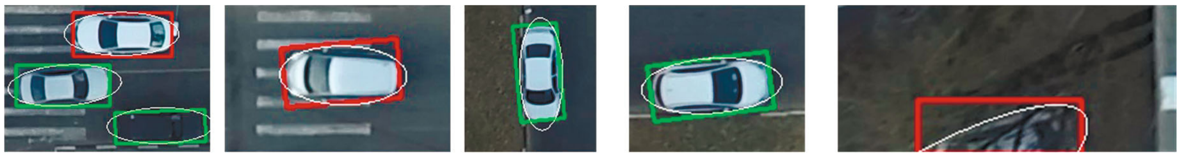


Fig. 4. Examples of min rectangles and ellipses building using polygonal masks of vehicles.

The process of removing vehicles from the image in this work was performed manually in a graphical editor, while the automation of this process using generative adversarial neural networks (GANs) is promising [13, 14]. Studying the capabilities of GAN for such a task is the subject of a separate study.

The limitation of current implementation of intersection simulator is the requirement of road scene with cloudy weather in the absence of shadows from objects. Otherwise, photorealism will be substantially disrupted.

In the simulator, user can choose one of two types of vehicle motion models and virtual sensors: simple kinematic models that do not take into account the dynamics of objects and the OpenAI Gym environment [15] in which the dynamics of cars are taken into account.

The simulator allows us to display information about the current iteration of the calculations, the maximum number of vehicles at the intersection (up to 4), as well as the current reward during testing the reinforcement learning algorithm. There are also buttons to start/pause/stop the simulation, as well as start/stop the process of image recognition.

The simulator is implemented in the Python 3.7 programming language and its source code is publicly available [16]. It uses NVidia CUDA technology to work effectively with various investigated neural network architectures based on Keras, Tensorflow or PyTorch platforms.

The ability to research various deep neural networks to solve computer vision problems and vehicle path planning using a photorealistic model of the intersection was the main goal of creating the simulator in question.

4. VEHICLE RECOGNITION WITH DEEP NEURAL NETWORKS

4.1. Detection Approach Based on Segmentation with Fully Convolutional Network and Clustering

The images used in this research have the specialty that the desired objects (cars) in the image do not overlap (Fig. 5). Therefore, it is promising to use a quick approach, which consists in segmenting the scene

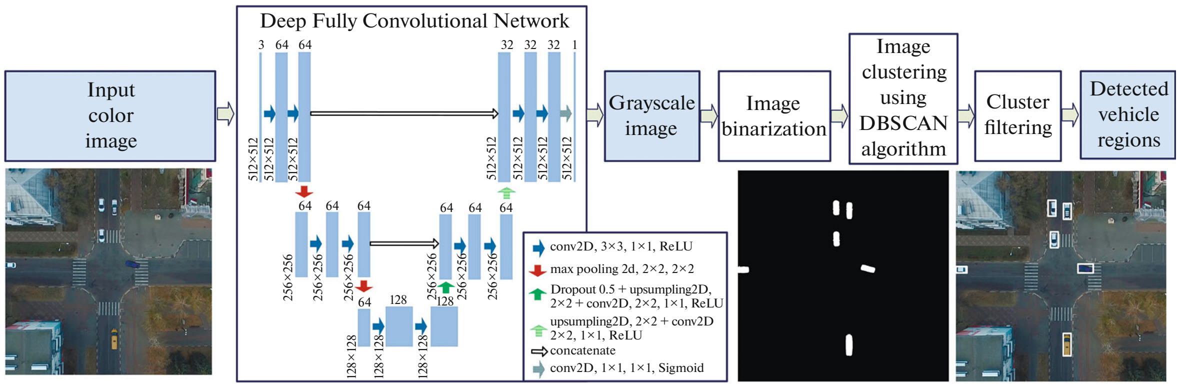


Fig. 5. Structure of vehicle detection approach based on segmentation with fully convolutional network and clustering.

by U-Net-like light fully convolutional network (FCN) and its subsequent clustering. This approach was developed by the authors for the task of traffic light detecting on the road scene [17].

As the input FCN takes one color image of the road scene, at the output of the neural network we get a single-channel grayscale map, the brightness of the its pixels varies from 0 to 255. The closer the pixel brightness of the map to 255, the more similar it is to the searched area—the vehicle.

At the binarization stage, the empirically chosen brightness threshold is applied, for our task its value is 128.

Next, clustering of the resulting binarized image into closed regions is performed. We use fast DBSCAN algorithm [18] to solve this subtask.

Then the detected regions are filtered. Too small clusters with an area less than 100 pixels are discarded. At the last stage we form bounding boxes around the remaining clusters, which are the result of the vehicle detection algorithm.

To assess the quality of the segmentation algorithm, the *Dice* measure is used.

$$Dice = \frac{2(Y_{GT}, Y_P) + \varepsilon}{(Y_{GT}, Y_{GT}) + (Y_P, Y_P) + \varepsilon}.$$

The main cause of this selection is small number of pixels in vehicle masks in relation to the number of pixels in the background. In the formula, the notation (Y_{GT}, Y_P) is the scalar product of two vectors: the Y_{GT} vector containing the "true" segmentation of the image represented as a one-dimensional array, and the Y_P vector containing the segmentation result using the applied algorithm, represented as a one-dimensional array. Each element of the Y_{GT} and Y_P arrays takes values in the range [0, 1], where 0 corresponds to a pixel not related to the desired area (traffic light), 1 corresponds to a pixel falling into the image region with a traffic light. Thus, the value (Y_{GT}, Y_P) is equivalent to the area of intersection of two regions, (Y_{GT}, Y_{GT})—equivalent to the area of true regions containing traffic lights, (Y_P, Y_P)—equivalent to the area of regions with traffic lights, found using the algorithm, ε—magnitude, equal to 1 (one pixel), necessary to take into account the case when the image has neither true nor found regions with traffic lights.

To train and test a convolutional neural network, a loss function based on the *Dice* measure is applied:

$$Loss = 1 - Dice.$$

Thus, during the training process, the intersection area of the found and true regions containing vehicle is maximized.

The training process for a convolutional neural network was carried out using the Adam optimizer with a learning rate coefficient of 0.0001.

4.2. Vehicle Detection and Instance Segmentation Based on Different DNN Architectures

On the base of prepared dataset of aerial images we research different modern architectures as for detection (YOLOv3 [19], Faster R-CNN [20], RetinaNet [21] and Cascade R-CNN [22]) as for instance segmentation (Mask R-CNN [23], Cascade Mask R-CNN [24]).

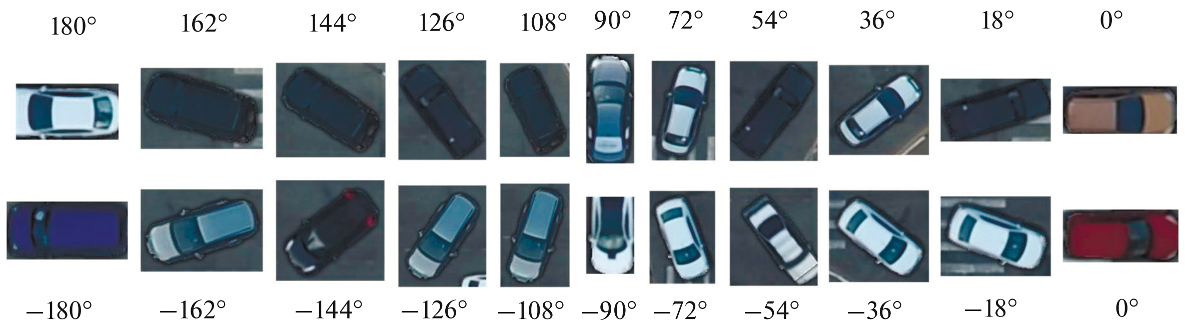


Fig. 6. Dataset fragment for vehicle orientation angle regression.

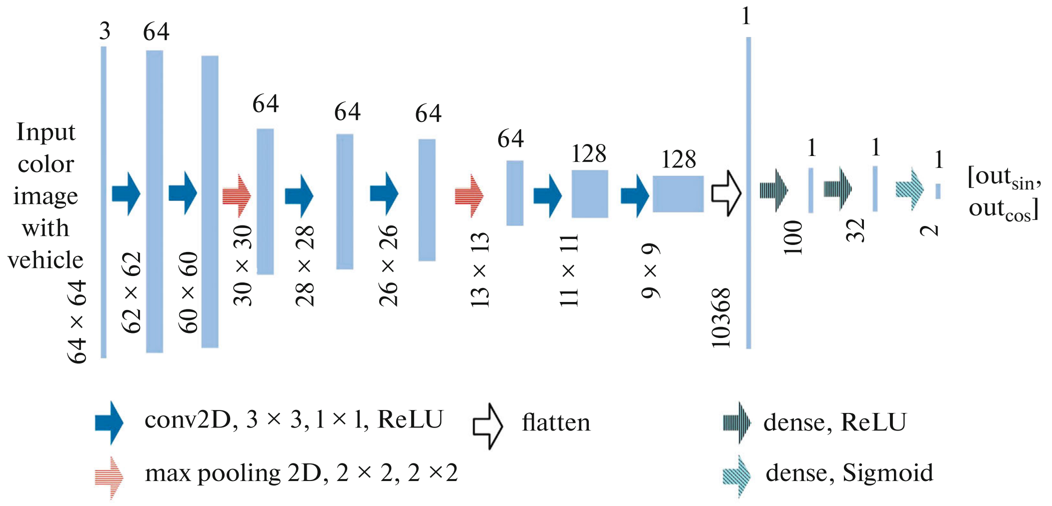


Fig. 7. Architecture of vehicle orientation angle regression deep neural network.

As backbone for such detectors as Faster R-CNN, RetinaNet and Cascade R-CNN we have used Res-Net-50 unit and Feature Pyramid Network (FPN). This choice is due to their relatively high speed and efficient operation at different image scales.

In addition, we had investigated the capabilities of the HRNet (hrnetv2p W48) [25] and ResNeXt (x101 32x4d) [24] backbones for instance segmentation with Mask R-CNN architectures which are provided high intersection over union (IoU) values for detected bounding boxes and ground truth.

4.3. Vehicle Orientation Angle Estimation Using Convolutional Neural Networks

For the formation of control actions for an intelligent agent, it is important to know the movement direction of other road agents. In the present work, it was decided to analyze the possibility of determining the orientation angle of a vehicle from its static image obtained as a result of the detection algorithm. The orientation angle is counted as shown in Figure 6, while the image of the vehicle may be partially cropped.

The input color image of the vehicle has a small size of 64 × 64 pixels, therefore, a relatively simple deep convolutional neural network is used for its orientation recognition (see Fig. 7).

Since cropped vehicle bounding boxes have different aspect ratios, in the course of this study we compared the recognition quality depending on one of the two types of reduction to a square image (Fig. 8).

The output of deep neural network is vector of two values $[out_{sin}, out_{cos}]$ in the range [0, 1] corresponding to the sin and cos values of vehicle orientation angle. We cannot take the absolute value of the vehicle orientation angle to train the neural network directly, because Mean squared error (MSE) is used as a loss function. For two close angles with different signs located in the vicinity of 180° (for example, 178° and −175°), the quadratic error will be large, which will interfere with the learning process.

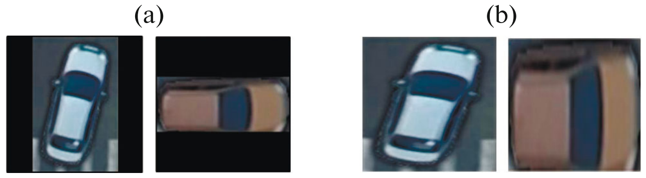


Fig. 8. Two types of squaring cut-out images with vehicles: (a) filling an empty area of the image with zeros (padded images), (b) stretching the image (stretched images).

This output data we can interpret as sin and cos values of vehicle orientation angle with formula:

$$sin_{pred} = out_{sin} \times 2 - 1\ ,$$

$$cos_{pred} = out_{cos} \times 2 - 1\ ,$$

$$angle = degrees\left(arctan2\left(sin_{pred}, cos_{pred}\right)\right).$$

angle value should be in the range [–180°, 180°]

Their squared sum can be used as confidence of angle prediction:

$$conf = \sqrt{sin_{pred}^2 + sin_{pred}^2}.$$

If *conf* is close to 1 than angle prediction is correct and otherwise if *conf* is less of some threshold value than angle prediction is wrong.

To estimate the orientation angle, we can also use the result of object segmentation, but this approach gives an answer with a period of 180° (an angle of –45° can be defined as an angle of 135°), which is unacceptable. Besides that, the angle can be approximately found on the basis of the relative displacement of the bounding box of the vehicle on two or more adjacent frames, however, in this work, an attempt was made to use just one static frame. For real-world applications, it is advisable to combine all three approaches in order to provide higher accuracy indicators of recognition of the vehicle orientation angle.

5. VEHICLE BEHAVIOR CONTROL USING DEEP REINFORCEMENT LEARNING

5.1. Environment Parameters

Developed environment is a intersection of two-way roads [26]. The goal of an agent to reach a red rectangular at the end of the road. During the movement the agent should avoid driving on walk side, wrong way or colliding with another cars. Environment includes the agent (a red car) and several bots (blue cars). The number of bots in the environment can be regulated by user and varied from 1 to 10. Each car including agent can start at different position among the four section of the roads (see Fig. 3). Each bot has predefined trajectory. It moves along three possible paths. Each path assigned to a bot as it spawned. It can be chosen by a user or otherwise assigned randomly. Bot car moves in the way which allows it to avoid any collisions with other cars. It stops if another car is directly in front of it. To model smooth behavior of bots on crossing section each of them follows designed pattern. Each of them has priority based on time a car-bot reach the crossing section. A car which drove to the section earlier has a priority. If car-bots' trajectories do not cross then each of them continue moving. Otherwise car with lower priority is not moving and wait until a car with higher priority leaves crossing area.

As soon as a car-bot reaches the end of its path a new bot-car generated at the beginning of random section with new trajectory. Thus there is always the same number of car-bots defined by user.

Information of the state of the environment is provided as a vector

$$st = \left(x, y, v_x, v_y, a_i, b_i, v_a^i, v_b^i, z_x, z_y\right)^T,$$

where:

—x, y: positions of the centre of the agent in the environment;

—v_x, v_y: x and y component of the agent's velocity vector;

—a_i, b_i: x and y positions of the centre of the i-th bot car in the environment;

—v_a^i, v_b^i : x and y component of the i-th bot car velocity vector;

—z_x, z_y: positions of the target.

To control the agent we pass a vector with three real numbers $a_t = (st_t, g_t, b_t)^T$, where:

—$st_t \in [-1, 1]$: steer, controls direction of movements, from left to right;

—$g_t \in [0, 1]$: gas, from no acceleration to full;

—$b_t \in [0, 1]$: break, where 0 is full stop.

In addition to continuous action space our environment supports a discrete action space of size 5:

—0: correspond to absence of actions at = (0, 0, 0)T in continuous action space; movements are following rules of the world without actions from the agent;

—1: left turn $a_t = (-1, 0, 0)^T$ in continuous action space;

—2: right turn $a_t = (1, 0, 0)^T$ in continuous action space;

—3: gas $a_t = (0, 1, 0)^T$ in continuous action space; agent is provided with short impulse to move;

—1: break $a_t = (0, 0, 1)^T$ in continuous action space; all movements of agent is stopped entirely.

5.2. Basic Control Algorithm Based on Reinforcement Learning

As the basic learning algorithm, we used Soft Actor Critic [27] (SAC) for continuous action space and Rainbow [28] for discrete action space. In this SAC approach, instead of maximizing the expected sum of rewards $\sum_t E_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t)]$, we will maximize the entropy objective function [29]:

$$\sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha H(\pi(\cdot \mid s_t))]$$

This function allows by maximizing the entropy term $H(\pi(\cdot \mid s_t))$ to support effective environmental exploration by maintaining an appropriate variety of actions and their sequences (trajectories). The temperature parameter α can be omitted by scaling the reward.

The Ballman operator B^π of updating the state-action estimate $Q(s_t,a_t)$when determining the current strategy π estimation for our problem is as follows:

$$B^\pi Q(s_t,a_t) = r(s_t,a_t) + \gamma E_{s_{t+1}\sim T, a_{t+1}\sim\pi}\left[Q(s_{t+1},a_{t+1}) - \log\pi(a_{t+1}|s_{t+1})\right].$$

Updating the strategy on the basis of the obtained estimation consists in choosing from the predetermined family of possible strategies Π (for example, Gaussian) the closest in the sense of the information distance Kulbleck-Leibler to the soft estimations given by the previous version of the π_{old} strategy $Q^{\pi_{old}}(s_t,a_t)$:

$$\pi_{new} = \arg\min_{\pi'\in\Pi} D_{KL}\left(\pi'(\cdot|s_t)\frac{\exp\left(Q^{\pi_{old}}(s_t,\cdot)\right)}{Z^{\pi_{old}}(s_t)}\right).$$

Here $Z^{\pi_{old}}(s_t)$ is a certain normalizing factor, which, when approximated, does not affect the final result of updating the strategy.

In the case of environments with a continuous set of actions, we turn to the algorithms of the actor-critic family and use the approximation of the parameterized estimations of the state-action $Q_\theta(s_t,a_t)$ and the strategy $\pi_\phi(a_t|s_t)$, respectively, with the parameters ψ and ϕ defined by deep neural networks. The objective function $J_Q(\theta)$ for updating the state-action estimation is the difference between the Bellman update of the estimations $\hat{Q}_\theta(s_t,a_t)$ and their current value $Q_\theta(s_t,a_t)$:

$$J_Q(\theta) = E_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - B^\pi Q(s_t,a_t)\right)^2\right],$$

ywhere D is the current state of the agent's memory, which is a distribution over previously observed states and completed actions. This function is optimized due to stochastic gradient descent in the parameter space θ. Similarly, the parameters ϕ of the strategy π are determined by minimizing the objective function defined by the KL distance:

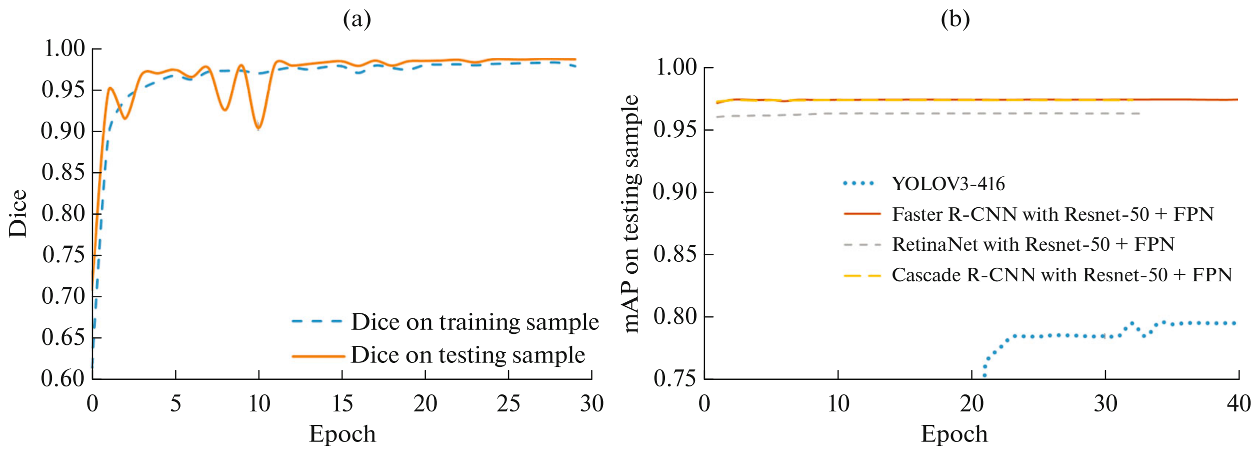


Fig. 9. Training process: (a) for vehicle segmentation using FCN, (b) for deep neural networks for object detection.

$$J_\pi(\phi) = E_{s_t \sim D}\left[D_{KL}\left(\pi_\phi(\cdot|s_t)\frac{\exp(Q_\theta(s_t,\cdot))}{Z_\theta(s_t)}\right)\right].$$

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1. Workstation Configuration

Computational experiments were done using server with GPU NVIDIA Tesla V100 32GB, CPU Intel Xeon Gold 6154 16 cores 3GHz and 128 GB RAM. Server were supplied by the Center for Science and Technology of Artificial Intelligence (CSTI) of MIPT. Operation system is Ubuntu 18.04 with configured NVidia CUDA GPU Toolkit 10.0. Programming language is Python 3.7 with Keras, Tensorflow and PyTorch deep learning frameworks.

6.2. Vehicle Detection on Aerial Photos

Training sample for object detection and segmentation includes 1600 labeled aerial images with different intersection scenes. Test sample contains 500 labeled images. Figure 9 shows the learning process for various image recognition approaches. Figure 9a demonstrates training process for vehicle segmentation using FCN. We can see high value of Dice Metric both on training and testing samples (>0.97) that reflects good quality of vehicle segmentation on aerial photos.

Figure 9b shows training process for deep neural networks for object detection for YOLOv3-416, Faster R-CNN, RetinaNet and Cascade R-CNN with Resnet 50 and FPN backbones. We can see that last three architectures provide fast training process and mAP of vehicle detection on testing sample higher than 96%.

Training process for instance segmentation of vehicles was done for several Cascade Mask R-CNN. It is shown in Fig. 10 and demonstrates high values of mean average precision (mAP) metric for vehicle detection with IoU in range of 0.5...0.95.

Table 1 shows the comparison of vehicle detection approaches based on deep neural networks on Testing sample.

The results obtained indicate high recognition quality and acceptable performance, which allows the use of the Faster R-CNN (mAP = 97.43, 17.6 fps) and RetinaNet (mAP = 96.32, 20.2 fps) architectures for reliable relatively fast recognition of vehicles in aerial photographs. Mask R-CNN-based architectures do not give sufficient improvement of quality but they are much slower. The main loss of detection quality in this case is due to the presence of vehicle overlaps or partial exit of objects beyond the frame.

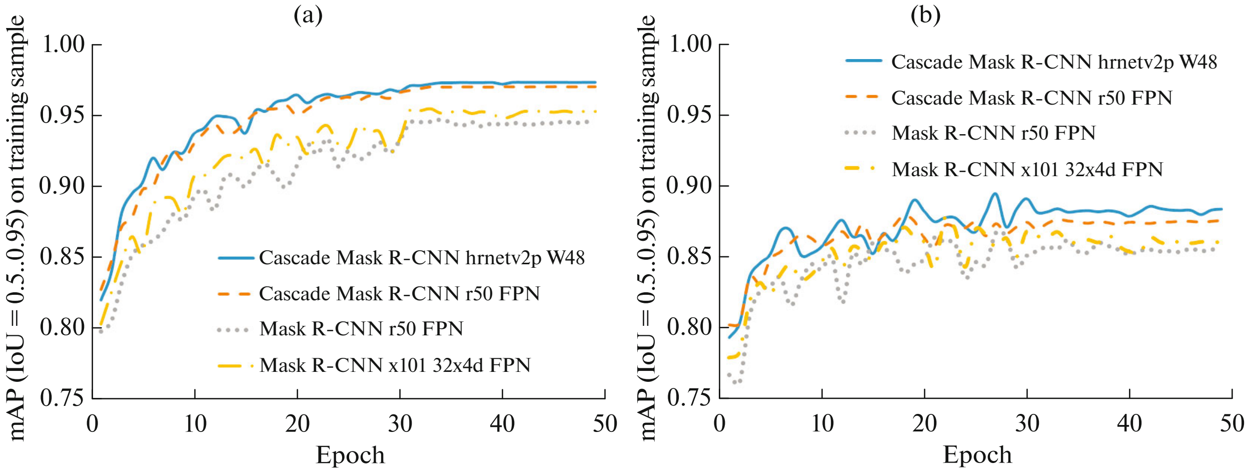


Fig. 10. Training process for deep neural networks for instance segmentation: (a) mAP (IoU = 0.5...0.95) on training sample, (b) mAP (IoU = 0.5...0.95) on testing sample.

6.3. Vehicle Orientation Angle Estimation

Training process of deep neural network for recognition of vehicle orientation angle with two types of input images is shown on Fig. 11. Training sample contains 1088 images and testing sample includes 100 images.

The better result was obtained for the case of padded input images. Mean squared error (MSE) is 0.0234 for test sample and mean absolute error (MAE) is 0.075. It is corresponds to the average angle recognition error of 4.5°. Average time of angle orientation recognition for one image is 8.2 ms. This result indicates that the approach to estimation of the vehicle orientation angle can be applied as part of both a simulator and a system for vehicle monitoring on the road based on aerial photographs.

6.4. Reinforcement Learning for Vehicle Behavior Control

Implementation of reinforcement learning approach was done using PyTorch deep learning library. In the first stage of the project we choose the Rainbow agent with discrete action space and run the learning process on the intersection environment and with goals to make the only one turn. Figure 12 shows preliminary results obtained by the agent with simple action discretization. In the learning process we have achieved that the agent avoids the main obstacles and tries to reach the goal—move to a given part of the intersection. Preprocessing of the global information containing in the image supplied to the input of the algorithm can significantly speed up the learning process.

Figure 13 shows illustration of Autonomous Vehicles Path Planning in the Intersection simulator with Object Detection based on Deep Reinforcement Learning.

Table 1. Comparison of vehicle detection approaches based on deep neural networks

Method	mAP for IoU ≥ 50%	Image recognition speed, fps
FCN-512 with DBSCAN clustering	54.30	28.2
YOLOv3-416	79.55	**36.3**
Faster R-CNN with Resnet-50 + FPN	97.43	17.6
RetinaNet with Resnet-50 + FPN	96.32	20.2
Cascade R-CNN with Resnet-50 + FPN	97.40	14.4
Cascade Mask R-CNN hrnetv2p W48	**97.56**	5.2
Cascade Mask R-CNN r50 FPN	97.49	6.5
Mask R-CNN r50 FPN	96.50	7
Mask R-CNN x101 32x4d FPN	97.43	6.4

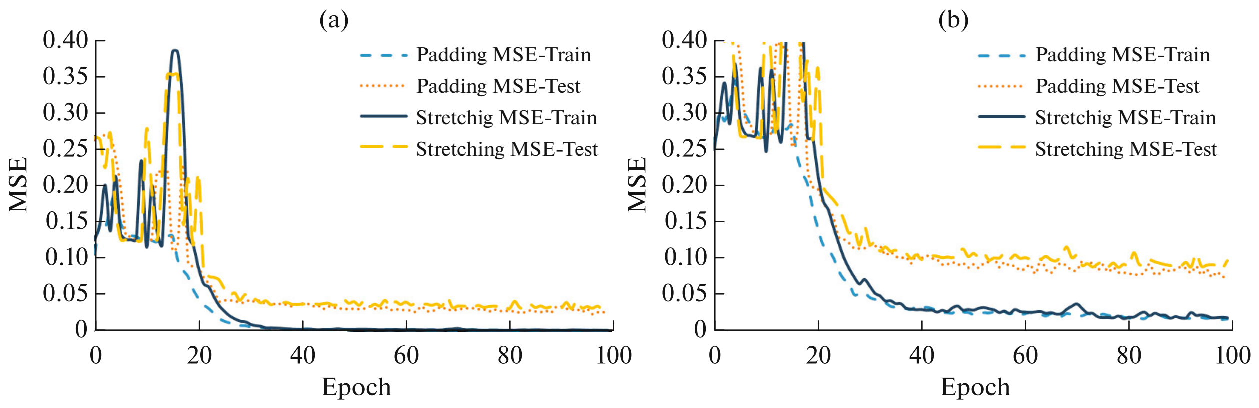


Fig. 11. Training process of deep neural network for vehicle orientation angle recognition: (a) MSE changing during training, (b) MAE changing during training.

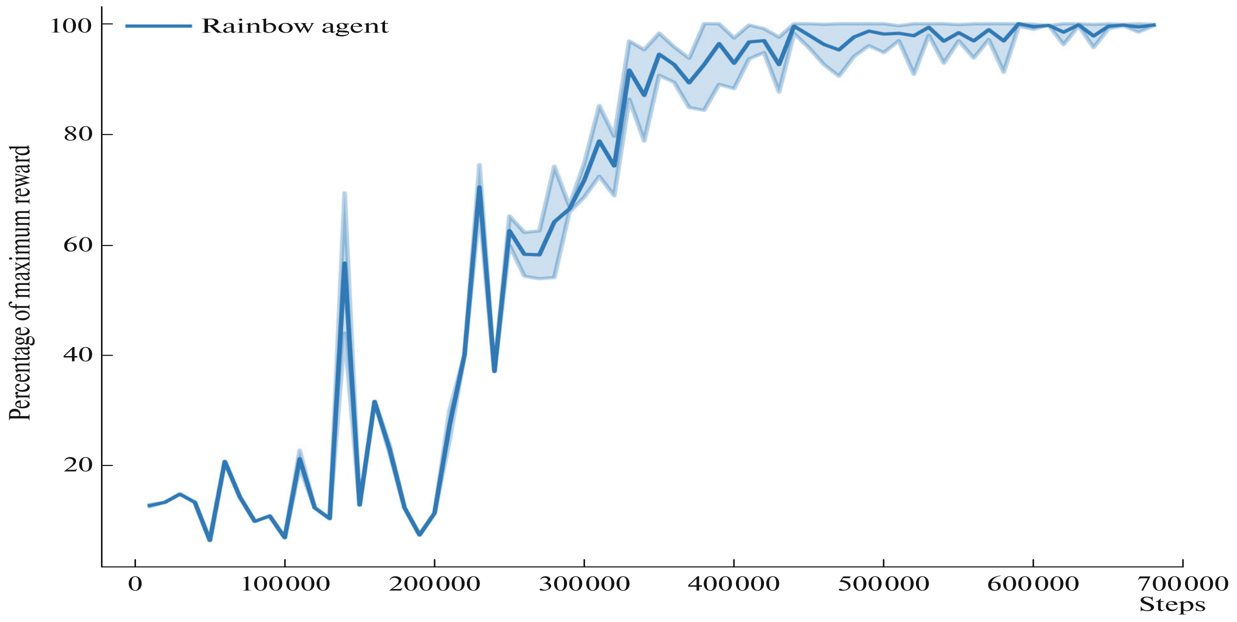


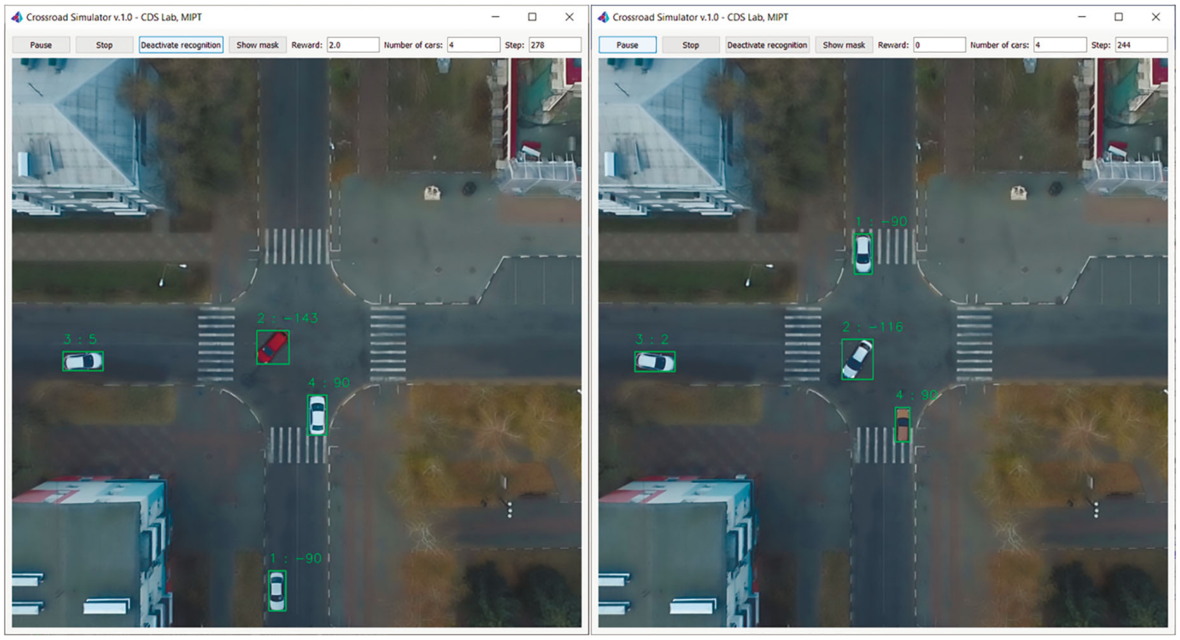
Fig. 12. The percentage of the total path that Rainbow agent traverses depending on the training step.

7. CONCLUSIONS AND DISCUSSION

A new approach to the training of an intelligent agent modeling an unmanned vehicle behavior based on the integration of reinforcement learning and computer vision methods is proposed. Using full visual information about the road intersection obtained from aerial photographs, it is studied automatic detection the relative positions of all road agents with various modern architectures of deep neural networks (YOLOv3, Faster R-CNN, RetinaNet, Cascade R-CNN, Mask R-CNN, Cascade Mask R-CNN). The mAP metric of object detection quality in this case for most architectures exceeds 0.96, the speed for a number of architectures is more than 20 frames per second, which is acceptable for further practical application.

The possibilities of estimation the vehicle orientation angle on the basis of a convolutional neural network were investigated. Developed approach had provided an average angle estimation error of less than 4.5°.

Preprocessed image is used by reinforcement learning method of Soft Actor critic and Rainbow, which allows to accelerate the convergence of its learning process. To demonstrate the operation of the proposed



Reward: 2.0 Number of cars: 4 Step: 278

Reward: 0 Number of cars: 4 Step: 244

Fig. 13. Illustration of Autonomous Vehicles Path Planning in the Intersection simulator with Object Detection based on Deep Neural Networks.

system, an intersection simulator was developed, at which a number of model experiments were carried out and had shown effectiveness and high quality of proposed approaches.

FUNDING

This work was partially supported by the Russian Science Foundation, project no. 18-71-00143 (chapters 5 and 6, behavior control with reinforcement learning and experimental results) and by Government of the Russian Federation, agreement no. 075-02-2019-967 (chapters 3 and 4, detection and recognition with computer vision).

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. Sakib, N., Yao, H., and Zhang, H., Reinforcing classical planning for adversary driving scenarios, arXiv: 1903.08606v1, 2019.
2. Xu, Z., Tang, C., and Tomizuka, M., Zero-shot deep reinforcement learning driving policy transfer for autonomous vehicles based on robust control, *IEEE International Conference on Intelligent Transportation Systems (ITSC),* 2019.
https://doi.org/10.1109/ITSC.2018.8569612
3. Kim, M., Path planning and following for autonomous vehicles and its application to intersection with moving obstacles, *MSc Thesis,* University of Florida, 2017. https://ufdc.ufl.edu/UFE0051086/00001.
4. Paxton, C., Raman, V., Hager, G.D., and Kobilarov, M., Combining neural networks and tree search for task and motion planning in challenging environments, arXiv:1703.07887v1, 2017.
5. Tram, T., Jansson, A., Gronberg, R., Sjoberg, J., and Ali, M., Learning negotiating behavior between cars in intersections using deep Q-learning, arXiv:1810.10469v1, 2018.
6. Jaeyoung, L., Balakrishnan, A., Gaurav, A., Czarnecki, K., and Sedwards, S., WiseMove: A framework for safe deep reinforcement learning for autonomous driving, arXiv:1902.04118, 2019.
7. Receveur, J.-B., Victor, S., and Melchior, P., Trajectory optimization for autonomous vehicles on crossroads with mobile obstacles, *IEEE Intelligent Vehicles Symposium (IV),* 2018.
https://doi.org/10.1109/IVS.2018.8500379

8. Chaplot, D.S., Gupta, S., Gupta, A., and Salakhutdinov, R., Modular visual navigation using active neural mapping, 2019. http://www.cs.cmu.edu/~dchaplot/papers/active_neural_mapping.pdf.
9. Bacchiani, G., Molinari, D., and Patander M., Microscopic traffic simulation by cooperative multi-agent deep reinforcement learning, arXiv:1903.01365v1, 2019.
10. Computer Vision Annotation Tool (CVAT). https://github.com/opencv/cvat.
11. Structural Analysis and Shape Descriptors. MinAreaRect. https://docs.opencv.org/3.1.0/d3/dc0/group__imgproc__shape.html.
12. Fitzgibbon, A.W. and Fisher, R.B., A buyer's guide to conic fitting, *Proc. 5th British Machine Vision Conference,* Birmingham, 1995, pp. 513–522.
13. Shetty, R., Fritz, M., and Schiele, B., Adversarial scene editing: Automatic object removal from weak supervision, arXiv:1806.01911, 2018.
14. Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A., and Catanzaro, B., Image inpainting for irregular holes using partial convolutions, arXiv:1804.07723, 2018.
15. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W., OpenAI Gym, arXiv:1606.01540, 2016.
16. Intersection simulator, MIPT, 2019. https://github.com/cds-mipt/raai-summer-school-2019.
17. Yudin, D. and Slavioglo, D., Usage of fully convolutional network with clustering for traffic light detection, *7th Mediterranean Conference on Embedded Computing, MECO'2018,* 2018, pp. 242–247.
18. Ester, M., Kriegel, H.P., Sander, J., and Xu, X., A density-based algorithm for discovering clusters in large spatial databases with noise, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining,* Portland, OR, 1996, pp. 226–231.
19. Redmon, J. and Farhadi, A., YOLOv3: An incremental improvement, arXiv:1804.027672018, 2018.
20. Ren, S., He, K., Girshick, R., and Sun, J., Faster R-CNN: *Towards real-time object detection with region proposal networks*, *NIPS,* 2015.
21. Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P., Focal loss for dense object detection, arXiv:1 708.02002v2, 2017.
22. Cai, Z. and Vasconcelos, N., Cascade R-CNN: *Delving into high quality object detection*, *CVPR,* 2018.
23. He, K., Gkioxari, G., Dollár, P., and Girshick, R., Mask R-CNN, arXiv:1703.06870, 2017.
24. Cai, Z., and Vasconcelos, N., Cascade R-CNN: High quality object detection and instance segmentation, arXiv: 1906.09756, 2019.
25. Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., and Xiao, B., Deep high-resolution representation learning for visual recognition, arXiv:1908.07919v1, 2019.
26. Shikunov, M., and Panov, A.I., *Hierarchical reinforcement learning approach for the road intersection task, Biologically Inspired Cognitive Architectures 2019, BICA 2019, Advances in Intelligent Systems and Computing,* 2020, pp. 495–506.
27. Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S., *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor,* 2018.
28. Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D., Rainbow: Combining improvements in deep reinforcement learning, arXiv:1710.02298, 2017.
29. Ziebart, B.D., *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy,* Carnegie Mellon University, 2010.