Received October 17, 2020, accepted October 24, 2020, date of publication October 28, 2020, date of current version November 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034524

Real-Time Object Navigation With Deep Neural Networks and Hierarchical Reinforcement Learning

ALEKSEY STAROVEROV[1], DMITRY A. YUDIN[1], ILYA BELKIN[1], VASILY ADESHKIN[1], YAROSLAV K. SOLOMENTSEV[1], AND ALEKSANDR I. PANOV[1,2]

[1]Moscow Institute of Physics and Technology, 141700 Dolgoprudny, Russia
[2]Federal Research Center "Computer Science and Control," Russian Academy of Sciences, 119333 Moscow, Russia

Corresponding author: Aleksandr I. Panov (panov.ai@mipt.ru)

This work was supported in part by the Russian Science Foundation [Theoretical Investigation and Methodology (Sections I–V)] under Grant 20-71-10116, and in part by the Government of the Russian Federation [Experimental Evaluation (Section VI)] under Grant 075-02-2019-967.

ABSTRACT In the last years, deep learning and reinforcement learning methods have significantly improved mobile robots in such fields as perception, navigation, and planning. But there are still gaps in applying these methods to real robots due to the low computational efficiency of recent neural network architectures and their poor adaptability to robotic experiments' realities. In this article, we consider an important task in mobile robotics - navigation to an object using an RGB-D camera. We develop a new neural network framework for robot control that is fast and resistant to possible noise in sensors and actuators. We propose an original integration of semantic segmentation, mapping, localization, and reinforcement learning methods to improve the effectiveness of exploring the environment, finding the desired object, and quickly navigating to it. We created a new HISNav dataset based on the Habitat virtual environment, which allowed us to use simulation experiments to pre-train the model and then upload it to a real robot. Our architecture is adapted to work in a real-time environment and fully implements modern trends in this area.

INDEX TERMS Indoor navigation, cognitive robotics, object segmentation, neural networks, intelligent agents, real-time systems, robot learning.

I. INTRODUCTION

Real-time navigation, path planning, localization and object avoidance are the major challenges of mobile robots [1]. Often such problems are solved by methods that do not use machine learning [2]–[4]. Such approaches often face a low level of adaptability to the various conditions in which the robotic platform finds itself.

On the one hand, this provides a relatively high speed of solutions but requires knowledge of dynamic models of the robot and environmental objects movement, noise models in sensor data, and other a priori data about the environment. In real scenarios of the robot's operating, such a priori data may not be available. Therefore, it is necessary that robot had the opportunity to learn in the course of performing any actions in the environment. Thus, it can be considered as an intelligent agent [5] whose behavior is synthesized by the control architecture, which includes the learnable subsystems.

Recently, deep learning and reinforcement learning (RL) methods have brought significant improvements for mobile robots. They provide new neural network methods in such fields as perception, navigation, and planning [6], [7]. However, there are a number of difficulties in applying these methods due to the low computational efficiency of the most recent neural network architectures and their low adaptability to the features of robotic experiments.

In this work, we consider a particularly important vision-based task in mobile robotics - indoor navigation to an object using an RGB-D images. The solution to this problem requires the study of three subtasks:

1) instance segmentation of the target object,
2) ego-motion estimation and localization on the map,
3) automatic exploration and path planning.

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar.

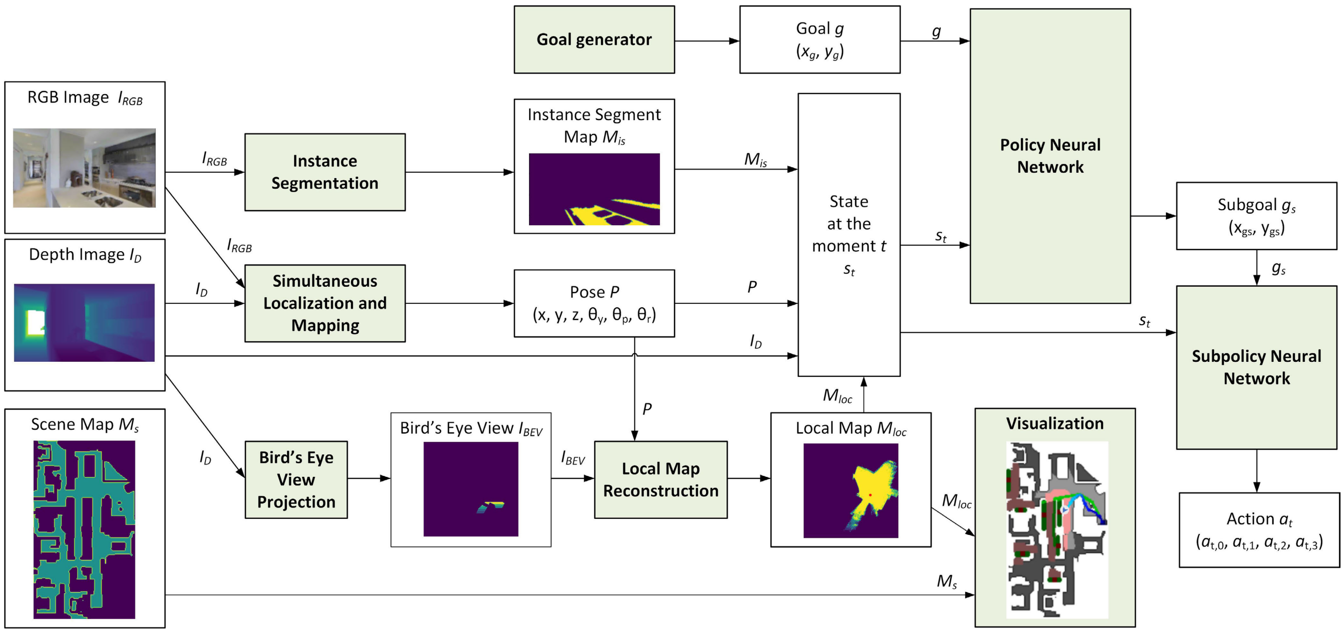


FIGURE 1. **Structure of the proposed Habitat-based Instance Segmentation, SLAM and Navigation (HISNav) framework.**

For each of the tasks presented above, there are a number of classical and neural network methods for solving problems that are of sufficient quality but not always computationally efficient in order to be used in energy-efficient embedded control systems of mobile robots. In this article, we propose real-time methods for solving these problems that are combined into a single learnable behavior control framework for a mobile robotic platform that automatically explores the environment and navigates to the specified class of objects (see Figure 1). Our solution is based on the well-known *sim2real* paradigm when we pre-train compact neural networks using a simulator, and then transfer the resulting models to a real robot. It is well known that the quality of the resulting solution directly depends on the quality of the simulator, its randomization capabilities, and the generalizing ability of the neural networks used. We created unique datasets based on the accessible Habitat simulation environment [8], which allowed us to build efficient compact models for solving the problems of instance segmentation and localization on the map. Using the Habitat simulator, we were able to create complementary agent's policies for map exploration and trajectory planning.

Many well-known methods used for solving the object navigation problem are weakly resistant to noise. Particularly clear results were shown by the recent Habitat Challenge 2020 competition [9], in which almost no one managed to cope with the noise of both cameras and actuators at the same time. Thus the task of creating unified neural network architectures for robot control in real-time to solve the problem of object navigation, resistant to various kinds of noise, is unsolved and relevant from a practical point of view. The architecture we are developing allows us to move forward in solving this global problem.

Detail analysis of these topics is carried out in the following article sections: related work, methodology, and experimental results. The main contributions of the authors of the article are:

- new neural network architecture for robot control that is fast and resistant to possible noise in sensors and actuators,
- original integration of semantic segmentation, mapping, localization, and reinforcement learning methods to improve the effectiveness of exploring the environment, finding the desired object, and quickly navigating to it,
- a new approach to control data integration in simultaneous localization and mapping algorithm to increase its quality metrics,
- a new HISNav dataset based on the Habitat virtual environment and included RGB-D images, their instance segmentation labels, robot camera poses and control actions for different indoor scenes.

Proposed HISNav Framework and Dataset are publicly available at https://github.com/cds-mipt/HISNav under MIT Licence.

II. RELATED WORK

When solving the problem of automatic navigation to the specified class of objects using the RGB-D camera, we solved subtasks for which there are already a number of high-quality neural network solutions. When creating our architecture, we relied on a number of the methods that we have significantly improved and adapted for real-time work on a mobile robot.

A. AUTONOMOUS AGENT INDOOR NAVIGATION

Autonomous navigation has a long history in robotics and various fields of artificial intelligence. For a human, navigation is a complex task, including simultaneous localization and mapping (SLAM), planning a route on it, setting intermediate goals, controlling one's actions, and interacting with the environment. In the classical approach, such a complex system is implemented through various modules, each responsible for its own task. These subsystems include SLAM, object segmentation, planning, exploration modules, etc. Recently, neural network learning methods as a part of those modules have also begun to appear. Integrating such subsystems into a unified neural network architecture allows us to hope that the general problem can be solved with training in an end-to-end manner.

In the work [10], the authors investigated the task of navigation to specific coordinates on a previously unknown map. The agent had a camera, depth sensor, and agent position as input. In the course of the article, both classical approaches with SLAM (ORB-SLAM2) [11] and planning (D * Lite), and the neural network approach (Direct Future Prediction) [12] were analyzed in different noise conditions with different types of sensors. Even though they failed in surpassing humans, the authors concluded that with RGB-D data, the classical approach provides better results, especially in a challenging environment. But compared to RL, it suffers a disaster in noise conditioned or RGB input only. The classical method's advantage is the absence of training and difficulties with transferring to the real world, its interpretability, and stability. In conclusion, the authors say that the RL algorithm they are testing requires further improvements and does not indicate all end-to-end approaches, further demonstrated by the DDPPO [13] approach.

Active Neural SLAM (ANS) [14] is an excellent implementation of a combination of the classical approach and the neural network approach. The authors were able to surpass both classical methods and completely end-to-end algorithms. The task that the authors chose was a little more complicated; they were required not to reach a point with specific coordinates but in a finite number of steps to explore as large an area as possible in a previously unseen to the agent room. The modular structure remained from the classical approach, which consisted of the Global Policy, SLAM, Local Policy blocks. SLAM was carried out using neural networks in a supervised manner. Based on this reconstructed map and the agent's current position, Global Policy sets a goal on the map in the form of coordinates, following to which the agent maximizes the explored area. Global Policy is implemented through the classic RL algorithm Proximal Policy Optimization (PPO) [15] and learns an unsupervised manner. Local Policy takes coordinates from Global Policy, the agent's current state, and output an action that brings the agent closer to these coordinates. Even though Local Policy is based on PPO, it is trained in a supervised manner on the planner (Fast Marching Method), which gives ground truth action in learning mode. According to the authors, this approach provides more results than using a planner directly. Also, for only RGB input, the authors implemented neural network depth restoration and showed its effectiveness as insignificant in the results in comparison with RGB-D.

Goal-Oriented Semantic Exploration (SemExp) [16] is a continuation of ANS for another problem statement, navigation to the object. The map also remains unknown to the agent. The type of object is assigned to the agent as a target, and stopping at a distance from it is a termination condition. For object recognition, the authors used Mask R-CNN. The main difference was in the Global Policy, which already maximized not the explored area, but the distance traveled to the nearest target type object. The target coordinates are used only during agent learning to determine the reward. During the test, the agent does not have access to these coordinates. At the same time, Global Policy, in addition to maps of the studied area and obstacle maps, accepted semantic maps for each of the target types. As soon as the semantic module has seen the target object on the agent's camera, non-zero points of belonging to the target are taken as the target. Local Policy defines the action to achieve the goal from Global Policy using a deterministic scheduler (FMM), which showed similar performance to the learnable option under these conditions.

Decentralized Distributed Proximal Policy Optimization (DDPPO) [13] is an algorithm that is based on PPO and expanded its idea to the efficient parallelization with the linear character up to 256 GPUs. The authors tested their performance on the point navigation problem. The agent had an RGB-D sensor and a GPS + Compass for determining the agent's position as data. Using 64 GPUs and three days of training, the agent made 2.5 billion steps in the environment and reached a success rate weighted by path length (SPL) equal to 0.997. For the RGB version with GPS + Compass, SPL 0.92 was obtained. Using this end-to-end approach without any mapping or planning modules, the authors surpassed all previous methods and showed human-like performance. Long short-term memory (LSTM) [17] layers in policy were used for the agent memory effect. The only remark was that in the absence of GPS + Compass, SPL drops to 0.15. The authors call the problem with only RGB input the next step for further research.

In our work, we rely on the end-to-end approach based on PPO, adapting it to real-time conditions and extending it to the object navigation task.

B. REAL-TIME OBJECT SEGMENTATION

The problem of locating the target object to be reached during navigation is usually formulated as an instance segmentation task. This allows us to get more accurate information about the image's pixels belonging to the object and more correctly estimate its three-dimensional coordinates, which is challenging to do with a simple bounding box.

For this, various approaches based on deep learning are also distinguished, e.g., the well-known model Mask R-CNN [18] based on the Faster R-CNN [19] architecture with an additional fully convolutional head for object segmentation in the found bounding box. To increase such models' speed, basic feature extraction networks (so-called backbones) with fewer layers or mobile architectures are used, for example, based on MobileNet V1-V3 [20], but this negatively affects the quality of detection and segmentation. The modern development of two-stage segmentation is the relatively fast model YOLACT++ [21].

Another approach to improving the quality of segmentation of found objects involves deformation of the found contour with a special neural network, for example, based on the polar representation of the contour in PolarMask [22], the concept of the circular convolution in Deep Snake [23], or deep polygon transformer in PolyTransorfm [24].

Several modern models do not use intermediate bounding box detection but perform segmentation directly, albeit in several stages. These methods include modern architectures BlendMask [25] and CenterMask [26].

Recently, the concept of a one-stage SOLO method has appeared, which, in addition to the segmentation of dynamic objects, can also perform semantic segmentation of static objects [27]. Its modification SOLOv2 [28] uses the concept of a feature pyramid network, attention, and a matrix method of non-maximum suppression, which allows it to surpass other methods of instance segmentation in the quality of object detection. The problem with SOLO-based approaches is still low performance. At the same time, they have significant improvement potential thanks to the use of fast backbones based on ResNet34 or ResNet18 models [29], Deformable Convolution Networks (DCN) [30], Deep Layer Aggregation (DLA) [31].

The recently appeared cascade models Cascade Mask R-CNN [32], implement a step-by-step multi-stage refinement of the found areas of the location of objects, which leads to a significant increase in the accuracy and completeness of detection. At the same time, the speed of image processing slows down even more, which makes such cascade methods not applicable for real-time tasks.

In this article, we explore the capabilities of modern deep architectures with fast backbones for instance segmentation of indoor scenes that can provide real-time across a variety of hardware platforms.

C. VISUAL-BASED ROBOT LOCALIZATION

SLAM methods are used for robot pose and velocity estimation by data from RGB-D sensors. These methods can be divided into two categories: direct and indirect.

Dense direct methods are considered to be the most precise, but also the slowest. These methods minimize photometric error and apply geometric prior to dense or semi-dense scene geometry estimation directly on the image. Sparse direct methods work with pixel intensity values, but at the same time, have a sparse cloud of 3D points. Typical examples of direct SLAM methods are DSO [33], Direct Sparse Mapping [34] (last two methods are monocular), and StereoDSO (for stereo cameras). For different scenes with different sensors, these methods are difficult to transfer because sensors should be photometrically recalibrated, and also correct uncertainty map formation for matching points is required. Modern enhancements of these approaches are neural network methods that train in a self-supervised manner – D3VO [35], DeepMatchVO [36], DF-VO [37]. All of them allow generating pose estimation of two neighbor frames a monocular camera and depth map. D3VO can also generate an uncertainity map, DF-VO – optical flow. The effectiveness of neural network methods is limited by training samples cause the quality of neural network SLAM falls dramatically in an unfamiliar environment.

Indirect SLAM methods deal with transformed images or their features, computed for keypoints, segments, or objects, for example, special ArUco marks [38]. These methods allow reliably rejecting outliers among matched points. The sparse map requires less memory; localization is faster than in direct methods. But feature extraction is sensitive to blur, noise, angle of view, and lightning variation.

The most wide-spread and popular methods are that retrieve keypoints via classical computer vision methods ORB-SLAM2 [11], RTAB-Map [39], or their modern modification ORB-SLAM3 [40] or OpenVSLAM [41]. This group of approaches is most stable on unfamiliar scenes, but at the same time has significant problems with obtaining odometry during sharp turns of the robot.

There are promising SLAM methods that use neural networks to predict reliable descriptors both for keypoints and for images as a whole: hierarchical localization method [42], Hybrid WNN-CNN [43], HF-Net [44], GCNv2 [45], AD-VO [46], as well as the recently released DXSLAM [47]. On the one hand, they make it possible to generate more reliable descriptors for keypoints and images on the scenes for which they were trained, and thereby perform better point matching and loop detection. However, their applicability to new types of scenes requires additional research.

One of the modern trends is also the joint solution of SLAM, exploration and path planing problems, as suggested by Adaptive Computation SLAM (ACSLAM) [48] when using lidar data, as well as in Active Neural SLAM [14].

In this article, we propose our own approach for modifying fast SLAM methods, both classical and based on neural networks. To do this, we will explore the possibility of integrating control data from the navigation framework into them.

III. PROBLEM STATEMENT AND PROPOSED NETWORK

In this study, we set ourselves the task of developing an object navigation framework for an indoor environment (see Figure 1).

The source of data for this is RGB-D images, which contain both information about the color of pixels I_{RGB}, and a dense depth map of the observed scene I_D. This information can be obtained from modern high-precision RGB-D cameras commonly used in mobile robotics.

Also in the common statement, we assume that the general scene map M_s is not known to the agent, but can be used to

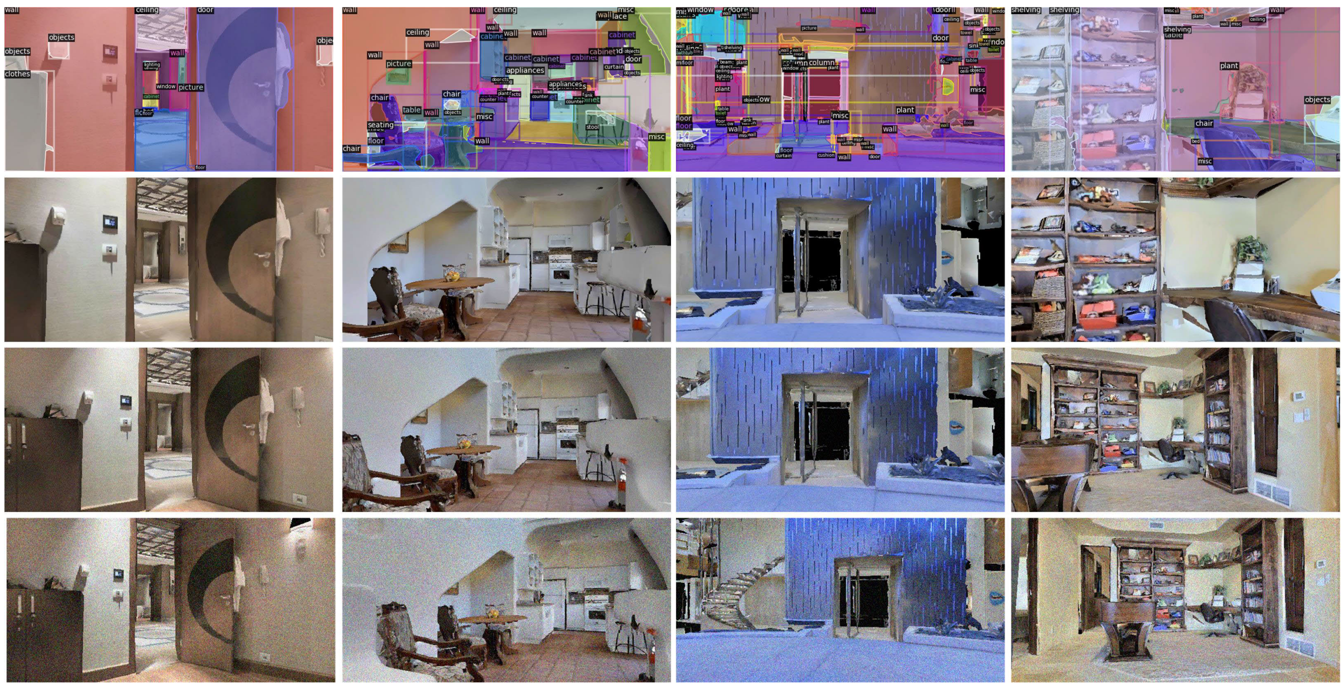


FIGURE 2. **Examples of images from HISNav Dataset with three levels of noise: the first row contains visualization of ground truth segmentation, second row demonstrates images without noise, the third row includes images with light Gaussian noise (*mean* $= 0$, $\sigma = 1$, *intensity* $= 0.05$), the bottom row contains images with strong Gaussian noise (*mean* $= 0$, $\sigma = 1$, *intensity* $= 0.1$).**

visualize the results. It is not available for the formation of robot actions.

At the moment of time t the robot itself can choose an action a_t belonging to the set $A = \{a_{t,0}, a_{t,1}, a_{t,2}, a_{t,3}\}$, where $a_{t,0}$ is stay in place, $a_{t,1}$ is turn left to the corner Δ_α, $a_{t,2}$ is turn to the right by an angle Δ_α, $a_{t,3}$ is move directly to the distance Δ_d. A similar type of movement is implemented in robots that have the ability to turn in place, in particular robotic vacuum cleaners, robotic couriers, etc.

In the framework being developed, navigation to the target is carried out by specifying a certain class of objects (for example, a chair, table, etc.). The goal generator spawns the coordinates of the target $g = (x_g, y_g)$. The target object is supposed to be detected by segmentation on a color image to obtain an Instance Segment Map M_{is}.

The robot's own pose P in the environment must be determined both the position of its center x, y, z and the orientation $\theta_y, \theta_p, \theta_r$ (the yaw, pitch, and roll angles). Pose estimation is possible using odometry data obtained at the output of the SLAM method, which uses RGB-D images as input (I_{RGB}, I_D pairs).

The navigation framework also involves the reconstruction of the two-dimensional local map M_{loc}, which was visited by the robot. It is formed on the basis of Bird's eye view image I_{BEV}, which is obtained as a projection onto the horizontal plane of the depth map I_D.

One of the key features of our framework is the automatic training of a hierarchical neural network, which allows, based on interaction with the environment, taking into account the current state of s_t (combining the segmentation map M_{is}, the current pose P, the depth image I_D, local map M_{loc}) to learn how to form actions a_t, bringing the robot closer to the given goal g. Automatic selection of subgoals allows the agent to significantly reduce the time required to explore the map and find the desired object.

The photo-realistic indoor simulator Habitat with Matterport3D-based scenes was chosen as the environment in which the robot operates. The structure of such a Habitat-based Instance segmentation, SLAM and Navigation Framework (shortly named as HISNav Framework) proposed by the authors is shown in Figure 1.

To study the qualitative indicators of the most important elements of the framework: the segmentation instance and SLAM, a special diverse dataset was prepared using the simulator. It will be discussed in detail in the next section.

IV. DATASET PREPARATION

In this article, we introduce a new Dataset - the HISNav Dataset, which consists of various robot movements tracks, recorded in virtual environment Habitat. Tracks were built on 49 unique scenes from Matterport3D [49] that present rooms with different styles. Each scene has no more than 5 trajectories with 3 different levels of noise in camera images and in actions.

We pursue the goal to research the steadiness of the developed framework to the noise. We use three levels of noise in images: without noise, light Gaussian noise (*mean* $= 0$, $\sigma = 1$, *intensity* $= 0.05$), strong Gaussian noise (*mean* $= 0$, $\sigma = 1$, *intensity* $= 0.1$). The examples of images from the dataset are shown in Figure 2.

Besides noise in images, noise to actions was added: without action noise, light action noise ($mean = 0, \sigma = 1, intensity = 0.2$), strong action noise ($mean = 0, \sigma = 1, intensity = 0.5$).

Each RGB image has a resolution 640×320, and the depth map has the same resolution. Each pixel contains a distance value in meters (from 0 to 100m). Ground truth instance labels of 40 classes (wall, floor, chair, door, table, sofa, etc.) correspond to each image.

For the convenience of visual SLAM algorithms research, the Dataset HISNav also contains ground truth of camera poses in TUM format [50] and information about control commands in the form of three rows of control matrice C_i, $1 \leq i \leq 4$, which were written string by string for each agent action along with the taking action moment.

All the dataset includes 135962 images and is split ted into three parts: train, val and test. Information about splitted samples can be found in Table 1. While splitting into samples a goal of diversity and balance between training, validation and test samples was pursued.

V. METHODOLOGY

A. REAL-TIME INSTANCE SEGMENTATION OF INDOOR SCENES

The HISNav Dataset described in the previous section has a number of peculiarities - objects can be both small and occupy a significant part of the frame. In this case, we need to segment both movable interior elements (for example, chair, table, towel, etc.) and stationary, background objects (wall, floor, window, stairs, etc.). In such a situation, two-step segmentation techniques that first detect bounding boxes and then segment them should not work very well. To test this, we will consider various architectures of the common model Mask R-CNN [18] with light backbones, as well as their modern counterpart YOLACT++ [21].

Consideration of models such as DeepSnake [23], PolarMask [22], PolyTransform [24] for this task is impractical since they are intended to refine the convex contours of segments without cavities. But in the prepared dataset, segments of objects may contain holes.

The most promising method seems to be the light versions of modern one-step fully convolutional approaches to the segmentation of objects Blendmask [25] and SOLOv2 [28]. They will also be explored in this article.

To compare the quality, it is advisable to analyze the resulting metrics mean Average Precision (mAP) [51] with different thresholds of Intersection over Union (IoU) measure. Inference time on different hardware platforms is also a topic of study.

B. INDOOR ROBOT LOCALIZATION USING VISUAL SLAM

To solve the task of robot localization in the environment using RGB-D images, we have chosen two modern open source methods as basic: OpenVSLAM and DXSLAM. Both of them refer to indirect sparse approaches that estimate the pose of the camera (robot) by the found keypoints and their descriptors: in the first case, by the ORB method, in the second, by the neural network-based method HF-Net [44]. They are improved versions of the popular ORB-SLAM2 [11] visual odometry method.

In these methods the current camera frame pose estimation is performed in two stages: at first, an approximate pose is estimated based on the pose of the previous frame or previous keyframe, then this pose is improved using local map (local bundle adjustment), which is built from multiple last keyframes (covisibility keyframes) and have common 3D points with the current frame. When the current frame becomes a keyframe, its pose is also refined by global bundle adjustment.

The current frame pose estimation includes keypoints matching, followed by bundle adjustment. Keypoints matching is performed based on its descriptors and accelerated with Bag Of Words. For further acceleration, modern approaches adopt the motion model for keypoints matching.

TABLE 1. Details of the proposed HISNav Dataset.

	HISNav-train	HISNav-val	HISNav-test	total
Number of images	72626	27952	35384	135962
Ratio, %	53,4	20,6	26,0	100
Number of unique scenes	49	35	43	
Number of tracks	88	35	43	
Number of instances per class (40 classes)				
wall	420782	171545	231628	823955
floor	181355	73785	94042	349182
chair	153759	54164	59711	267634
door	228804	83998	103098	415900
table	76787	33727	37339	147853
picture	80842	31947	39932	152721
cabinet	37649	18275	21674	77598
cushion	47449	25205	23442	96096
window	110639	35525	54558	200722
sofa	24254	10277	10999	45530
bed	19651	8117	10491	38259
curtain	45786	14236	16187	76209
chest of drawers	16311	5817	8023	30151
plant	36286	19104	27240	82630
sink	7002	3014	3275	13291
stairs	19337	6359	11485	37181
ceiling	139829	55721	81929	277479
toilet	1948	1219	1444	4611
stool	14721	6572	9056	30349
towel	6082	3201	4642	13925
mirror	14857	5987	6692	27536
tv monitor	13440	5357	8471	27268
shower	5987	1166	1726	8879
column	31210	11130	10720	53060
bathtub	3854	1465	1571	6890
counter	15917	6626	9077	31620
fireplace	7218	2337	1960	11515
lighting	33907	9153	15949	59009
beam	5514	1490	3183	10187
railing	23927	9405	12778	46110
shelving	22927	11515	14981	49423
blinds	2150	782	1061	3993
gym equipment	868	372	495	1735
seating	19521	4747	7178	31446
board panel	1498	67	398	1963
furniture	3140	967	1277	5384
appliances	7839	2691	3745	14275
clothes	586	561	634	1781
objects	113931	45902	59614	219447
misc	223810	87047	126577	437434

In the motion model robot displacement is estimated from previous motion. In details, a velocity matrix V is estimated and the pose of next frame P_{cw}^{i+1} is calculated from the pose of previous frame P_{cw}^{i} as follows: $P_{cw}^{i+1} = VP_{cw}^{i}$. When P_{cw}^{i+1} is known, keypoints of the previous frame are projected into the current frame and point correspondence is found based on euclidian distance on image plane.

When agent motion is continuous and smooth motion model assumptions are satisfied and keypoints matching based on euclidian distance on the image plane is performed correctly. However, when motions are sharp, these assumptions become incorrect and its use leads to rapid loss of keypoints and quality decrease. But when robot displacement in the environment results from performing certain known action and expected displacement is known, it can be used instead of the standard motion model.

In the Habitat environment an agent (robot) is allowed to perform four actions: go forward for d meters, rotate for $\pm\alpha$, finish episode. For these actions, we can determine robot displacement in the environment using matrices of control C_i, $1 \leq i \leq 4$, which have the next form:

$$C(\alpha, d) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{1}$$

This matrix determine displacement in the direction of z axis for d and rotation around y axis for α. In these notation, forward displacement is given by $C_1 = C(0, d)$, rotation - $C_{2,3} = C(\pm\alpha, 0)$, episode finish $C_0 = C(0, 0)$. Then the velocity matrix for motion model is constructed $V = C_i^{-1}$. It is worth noting that only only expected displacement is known. In fact, displacement is determined by a random matrix $\hat{C} = C(\alpha+\xi_\alpha, d+\xi_d)$, where ξ_α ξ_d - random variables, noises in rotation angle and forward step. This means that information about actions is insufficient to localize on the map without the use of visual odometry.

The proposed approach with the displacement of the standard motion model with the control model allows us to integrate prior knowledge about robot motion in the SLAM method and, as shown in experiments, improve robot localization from camera images and depth maps.

For our study, the following SLAM metrics were taken:

1) Relative translation (T_{KITTI}, %) and rotation (R_{KITTI}, deg/m) errors which are introduced in the KITTI Odometry Benchmark [52], [53]. Because of short indoor tracks in the HISNav Dataset, we use distance subsequences of length (0.25, 0.5, 1, 2, 4, 8, 16, 20) meters instead of conventional (100, 200, ..., 800) distances.

2) Also, we use absolute pose error for translation (APE_T, m) and rotation (APE_R, deg) introduced in the tool [54].

Absolute pose error for translation APE_T is computed as median of the errors array E_T. Each element of the array E_T is calculated as follows:

$$E_T^i = ||P_{pred}^i - P_{GT}^i||_2, \tag{2}$$

where P_{GT}^i is the ground truth translation for i-th frame $(x_{GT}^i, y_{GT}^i, z_{GT}^i)$, P_{pred}^i is the predicted translation for i-th frame $(x_{pred}^i, y_{pred}^i, z_{pred}^i)$, $||.||_2$ is $L2$ norm corresponding to absolute error between two points.

Absolute pose error for rotation APE_R is computed as median of the errors array E_R.

For each frame i error pose P_R^i is calculated as follows:

$$P_E^i = (P_{pred}^i)^{-1} \cdot P_{GT}^i, \tag{3}$$

where P_{pred}^i is the predited pose (4×4) on frame i, P_{GT}^i is the ground truth pose (4×4) on frame i.

Then, from each error pose (4×4) P_E^i rotation matrix (3×3) is extracted. Each rotation matrix is transformed into a 3 dimensional vector which is co-directional to the Euler axis of rotation and whose norm gives the angle of rotation (in radians). After that from each rotation vector, norm is obtained. Norm of the received rotation vector is transformed from radians to degrees and used as angle error E_R^i itself.

3) Lost (tracks), %. This metric shows in percentage how often SLAM method fails in tracking at the beginning or at the end of a track.

SLAM method is considered to fail tracking if one of the two conditions is true:

- if the first successfully predicted pose stands further than 0.2 sec from the first GT pose.
- if the last predicted pose stands further than 0.2 sec from the last GT pose.

C. REINFORCEMENT LEARNING FOR INDOOR OBJECT NAVIGATION

After solving the subtasks of instance segmentation and agent localization on the map, the agent needs to solve the key subtask - fast navigation to the desired object. We have chosen a modular hierarchical approach that allows us to separate the tasks of navigation in a short horizon and route planning over long distances. Another advantage of our approach is the absence of a complex reward function during learning process, which is usually programmed manually and cannot convey all the features of the environment.

The main purpose of our method is to learn k-level policy Π_{k-1}. We took k as 2. Each level of policy learns $\pi_i : S_i \times G_i \rightarrow A_i$. To learn these policies π_0,π_1 we use the set of Universal Markov Decision Process (UMDP) U_0,U_1, in which $U_k = (S, G, A, R, \gamma)$ where γ is a discount factor. The state space S to this policies is identical and consist of depth, semantic sensor observation and localization information. Both layers goal space G consist of coordinates of the goal. First higher layer action space A is the coordinates of subgoal that needed to achieve the task goal. Also, it should be mentioned that the first layer output subgoal measures relative to the current agent position and restricted to some area *Targ* around to be achievable in any situation. Second lower layer action space is the action that agent needed to perform to achieve the subgoal that was given by the first layer. After the subgoal is set by the first layer of policy, the second layer of policy need to output the sequence of N^a_{max} actions to achieve the given subgoal. If this subgoal was achieved, agent recieve reward R of 0, else -1. The first layer has maximum of N^t_{max} attempts to achieve the final goal.

The RL agent have three actions: forward, turn left, and turn right. Evaluation occurs when the goal object appears in the agent camera and is closer than 0.5 m. As a metric, SPL (Success weighted by Path Length) is used (4).

$$SPL = \frac{1}{N}\sum_{i=1}^{N}\frac{l_i}{max\,(p_i, l_i)}, \tag{4}$$

where l_i is the length of shortest path between start point and goal object that was closest to the point where agent trajectory is end p_i is the length of path taken by agent in an episode.

The lower level of a policy due to the discrete action space is based on DDDQN [55], the higher policy has a continuous action space (coordinates of the subgoal) and based on a Twin Delayed DDPG (TD3) [56] algorithm. Sructure of the neural net is shown here (Figure 3). As a goal state, TD3 receiving a semantic image because we filter it to 1 if object belongs to the goal class, else 0.

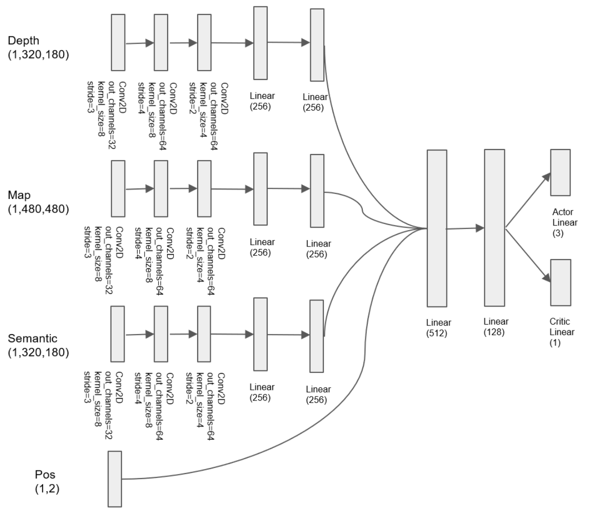


FIGURE 3. **Structure of the policy neural network.**

The biggest problem to learn these policies is that if all of the policies are to be trained in parallel, any level's actions cannot be evaluated with respect to the policy below that level. This lower level hierarchy will continue to change as long as these lower-level policies both learn from experience and explore. Changes in the lower level will cause non-stationary transitions, and rewards functions become difficult to learn at higher levels. To overcome this, we used the Hierarchical Actor-Critic [57] main idea. Instead of evaluating the actions with respect to the lower level of policies, evaluate the actions concerning where the lower level is headed — an optimal lower level hierarchy. The optimal lower level hierarchy, which consists of optimal versions of all lower-level policies, does not vary over time. As a result, the states and rewards for any action will be stable, allowing the hierarchical agent to learn its multiple levels of policies in parallel.

Optimal low lever hierarchy is achieved by two types of transitions: Hindsight Action Transitions and Hindsight Goal Transitions [57].

Hindsight action transitions serve to replace the proposed action with the action that was actually executed in hindsight. It means that for non-base levels when a policy presents a subgoal state, but the level below misses that subgoal state and ends in another state after N^t_{mid} attempts, the hindsight action transition will use that ended state as the original subgoal action. With this substitution, the next state and action components in the transition will be the same, and hindsight action transition is simulating how an optimal lower-level policy would act. For the reward component of the transition, the reward will only depend on the reached state and the goal state. The reward will not take into consideration the exact path taken to the state reached after N^t_{mid} attempts because the goal is to create transitions that simulate an optimal lower-level policy hierarchy, and it is not known what path an optimal lower-level policy hierarchy would take. The reward will also be sparse and binary to avoid the issues that arise when reward functions are manually engineered.

Hindsight action transitions reduce the non-stationary transitions, and the reward function problem is not presented. But dealing with sparse reward function can be very complicated due to the complexness of getting a positive reward. Hindsight Goal Transitions could overcome that. This type of transition extends the idea of Hindsight Experience Replay (HER). It is done by copying of original (s_t, a_t, r_t, s'_t, G) transition and replacing the achieved goal with a state that the agent stopped in. The reward is also recalculated according to the new conditions. For the non-base level, hindsight goal transitions being made from copies of the hindsight action transitions.

VI. EXPERIMENTAL RESULTS

A. HARDWARE PLATFORM

To test performance of proposed approaches we use different hardware platforms:

1) Server platform with NVidia Tesla V100 32Gb GPU, Intel Xeon Gold 6154 (16 cores 3GHz), 128Gb RAM,
2) Ground robot platform based on Clearpath Husky chassis with ZED camera 1200×600 and onboard computer with GTX1050Ti 4Gb, Intel Core i5-4570TE (2 cores 3.3GHz), 8Gb RAM (see Figure 4).

B. REAL-TIME INSTANCE SEGMENTATION OF INDOOR SCENES

In the course of computational experiments on the training set proposed by the HISNav Dataset, we train several modern models with fast backbones:

- SOLOv2 with ResNet34 backbone,
- Blendmask with DLA34 backbone,
- YOLOACT++ with ResNet50 and FPN backbone,

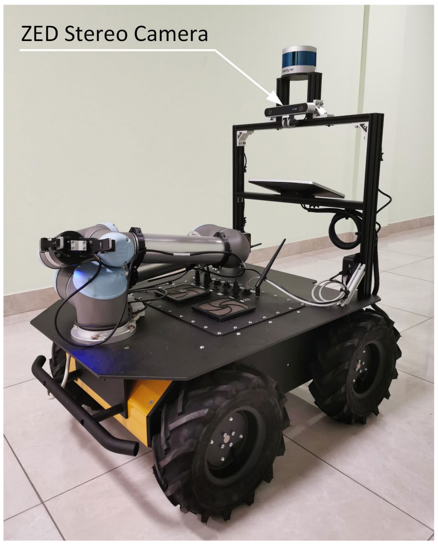


FIGURE 4. Ground robot platform based on Clearpath Husky chassis with ZED camera. It was used for experiments with proposed algorithms.

- Mask R-CNN implementation in Detectron2 [58] with Resnet50 and FPN,
- Mask R-CNN implementation in MMdetection [59] with the same backbone.

During training, batch size equal to 8 was set for all models. The quality assessment of the segmentation instance during training on the validation set (HISNav-val) is shown in Figure 6. Using mAP value of instance segmentation for all 40 object classes we have selected the best models of each type.

Using the selected models, Table 2 was built to detail the segmentation quality at different threshold IoU values on the HISNav-test dataset.

On its basis, we can conclude that the highest quality indicators are for the network with the Blendmask architecture in terms of mAP with IoU = 0.50:0.95. But the SOLOv2 model is better in more common metric mAP with IoU = 0.5. These two models can be considered as basic for integration into the proposed HISNav Framework.

TABLE 2. Quality of instance segmentation neural networks on HISNav-test data.

Model	mAP (IoU=0.50:0.95)	mAP (IoU=0.5)
No noise in images		
BlendMask	**0,2408**	0,3648
SOLOv2	0,2391	**0,3778**
YOLACT++	0,1691	0,2739
Mask R-CNN$_{det2}$	0,2198	0,34834
Mask R-CNN$_{mmdet}$	0,2267	0,3593
Light noise in images		
BlendMask	**0,2317**	0,3564
SOLOv2	0,2311	**0,3703**
YOLACT++	0,1631	0,2655
Mask R-CNN$_{det2}$	0,2097	0,3377
Mask R-CNN$_{mmdet}$	0,2174	0,3501
Strong noise in images		
BlendMask	**0,2175**	0,3427
SOLOv2	0,2154	**0,3527**
YOLACT++	0,1513	0,2526
Mask R-CNN$_{det2}$	0,1953	0,3204
Mask R-CNN$_{mmdet}$	0,2026	0,3363

Visualization of image instance segmentation for the all trained neural networks is shown on Figure 5. It demonstrates comparison of the different models with ground truth segmentation. SOLOv2 model seems more accurate than others which confirms its high quality metrics.

C. INDOOR ROBOT LOCALIZATION USING VISUAL SLAM

Table 3 shows that the proposed control-based DXSLAM method (named CDXSLAM) demonstrates better performance in terms of relative metrics T_{KITTI} and R_{KITTI}. We can also conclude that the integration of control data both in neural network-based DXSLAM and OpenVSLAM shows significant increase of relative and absolute quality metrics. A visualization of one of the tracks from the HISNav dataset is shown in the Figure 8.

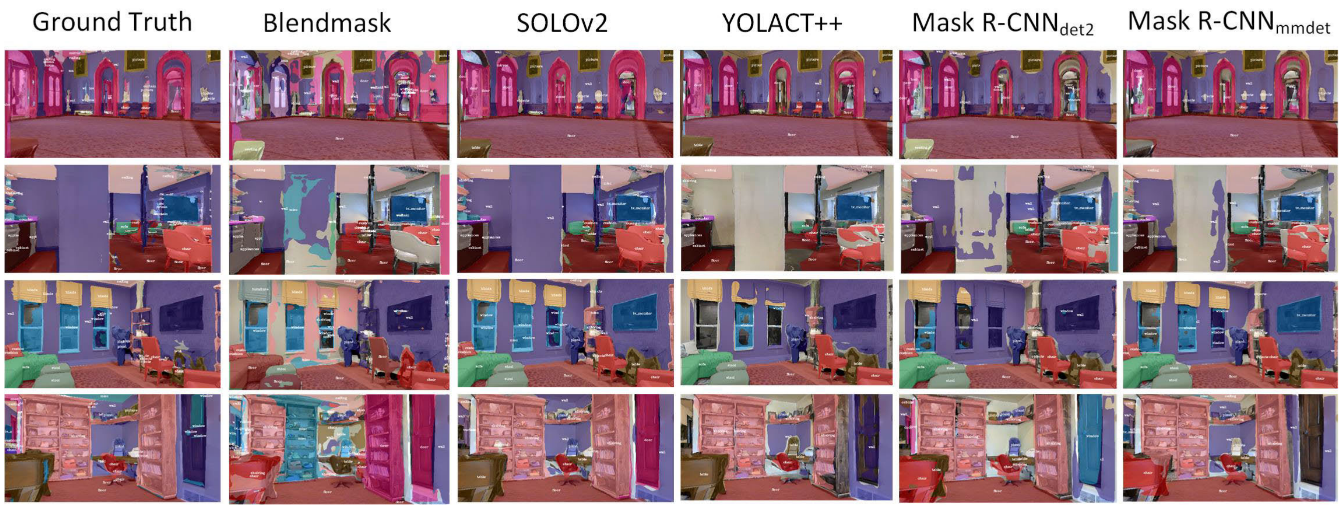


FIGURE 5. Examples of instance segmentation using different neural network models on images from HISNav-test dataset.

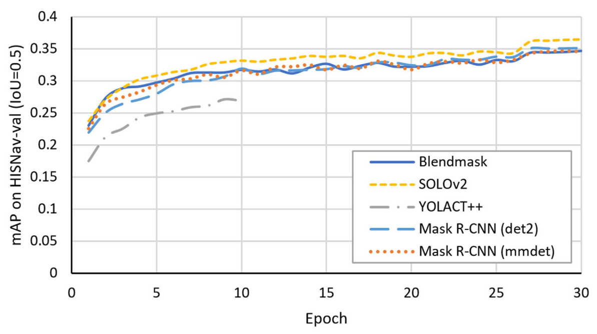


FIGURE 6. Results of instance segmentation models training.

TABLE 3. SLAM methods validation results on data with different motion mode.

Metrics	OpenVSLAM			CDXSLAM		
	motion	bow	control	motion	bow	control
Experiment results with fast motion ($\Delta_\alpha = 10^\circ$ and $\Delta_d = 25cm$)						
T_{KITTI}, %	14.59	12.78	12.55	23.21	13.91	**12.40**
R_{KITTI}, °/m	5.06	4.73	4.54	7.53	4.45	**3.77**
APE_T, m	0.79	0.72	**0.67**	1.13	0.79	0.77
APE_R, °	11.35	10.42	**9.71**	12.24	10.40	10.45
Lost, %	48.1	30.2	**27.1**	82.2	65.9	46.5
Experiment results with slow motion ($\Delta_\alpha = 5^\circ$ and $\Delta_d = 10cm$)						
T_{KITTI}, %	15.99	13.08	13.28	34.63	13.66	**11.48**
R_{KITTI}, °/m	4.80	4.49	4.43	16.54	4.18	**3.41**
APE_T, m	0.77	0.77	**0.70**	1.42	0.78	0.78
APE_R, °	10.84	11.36	10.77	27.84	10.79	**9.53**
Lost, %	38.8	**24.8**	27.1	89.9	45.7	28.9

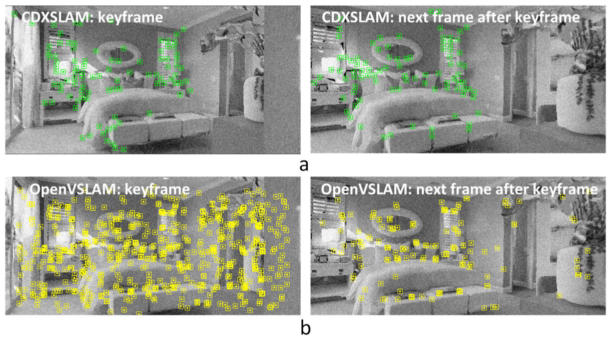

CDXSLAM: keyframe
CDXSLAM: next frame after keyframe
a
OpenVSLAM: keyframe
OpenVSLAM: next frame after keyframe
b

FIGURE 7. Keypoint detection and matching details in noisy images: a - for CDXSLAM method, b - for OpenVSLAM method.

Figure 7 demonstrates the efficiency of keypoints detection in noisy images by the considered SLAM methods. As we can see, there are no noise points on the key and subsequent frames of the neural network CDXSLAM. OpenVSLAM adds a lot of noise points to the map, which at the same time do not match the next one.

Table 4 also shows that the neural network-based method CDXSLAM often loses tracking on noisy data. A possible reason is that keypoints generated by neural network are less universal and their descriptors are less reliable than those generated by the ORB method. The basic neural network in CDXSLAM was trained on OpenLORIS [60] indoor dataset with real scenes, which did not contain type of noise, the effect of which we are analyzing in this article.

TABLE 4. SLAM methods validation results on data with different noise.

Metrics	OpenVSLAM			CDXSLAM		
	motion	bow	control	motion	bow	control
Experiment results on data without noise						
T_{KITTI}, %	15.16	13.89	14.15	23.12	15.20	**11.90**
R_{KITTI}, °/m	4.94	4.63	4.59	8.65	4.54	**3.57**
APE_T, m	0.74	0.72	**0.68**	0.89	0.74	0.78
APE_R, °	11.37	10.52	**9.95**	14.18	10.51	9.96
Lost, %	28.2	17.6	15.3	59.3	27.9	**15.2**
Experiment results on data with light Gaussian noise (mean = 0, $\sigma = 1, intensity = 0.05$)						
T_{KITTI}, %	16.14	13.15	**11.29**	175.05	12.10	–
R_{KITTI}, °/m	4.96	4.75	4.13	85.42	**3.97**	–
APE_T, m	0.86	0.76	**0.68**	8.09	0.82	–
APE_R, °	11.29	10.98	**9.44**	147.33	10.57	–
Lost, %	37.2	23.2	**17.4**	98.8	55.8	100.0
Experiment results on data with strong Gaussian noise (mean = 0, $\sigma = 1, intensity = 0, 1$)						
T_{KITTI}, %	14.34	**11.33**	14.52	–	11.88	–
R_{KITTI}, °/m	4.80	4.39	4.94	–	**3.98**	–
APE_T, m	0.70	0.75	**0.66**	–	0.89	–
APE_R, °	**10.12**	11.34	12.23	–	11.36	–
Lost, %	64.0	**40.7**	50.0	100.0	83.7	100.0

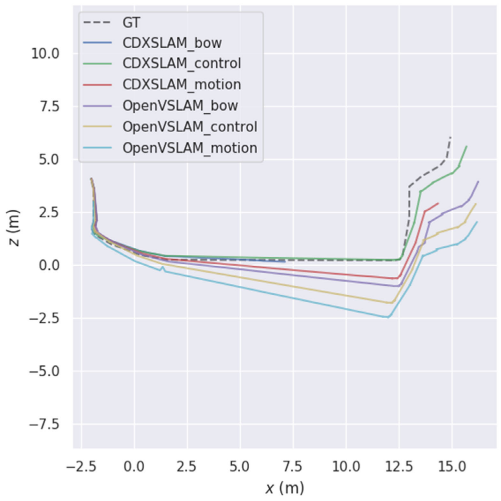


FIGURE 8. Results of studied CDXSLAM and OpenVSLAM methods on some tracks of HISNav Dataset. Integration in motion model of control data increase quality of both methods.

The experimental results of the CDXSLAM and OpenVSLAM on the ground robot platform moved in the university campus are shown in the Figure 9. We can see the main advantage of CDXSLAM: its trajectory is more smooth (see Figure 9b). It is important for some applications, for example, map reconstruction.

D. REINFORCEMENT LEARNING FOR INDOOR OBJECT NAVIGATION

Deep reinforcement learning methods lack the exploration bonus to speed up the learning process in the long-distance scenes. The most current promising approach is Random Network Distillation [61], which we choose to compare with our method. In the general case, the object navigation task comes down to a point navigation task with the addition of an exploration. We have been able to solve point navigation

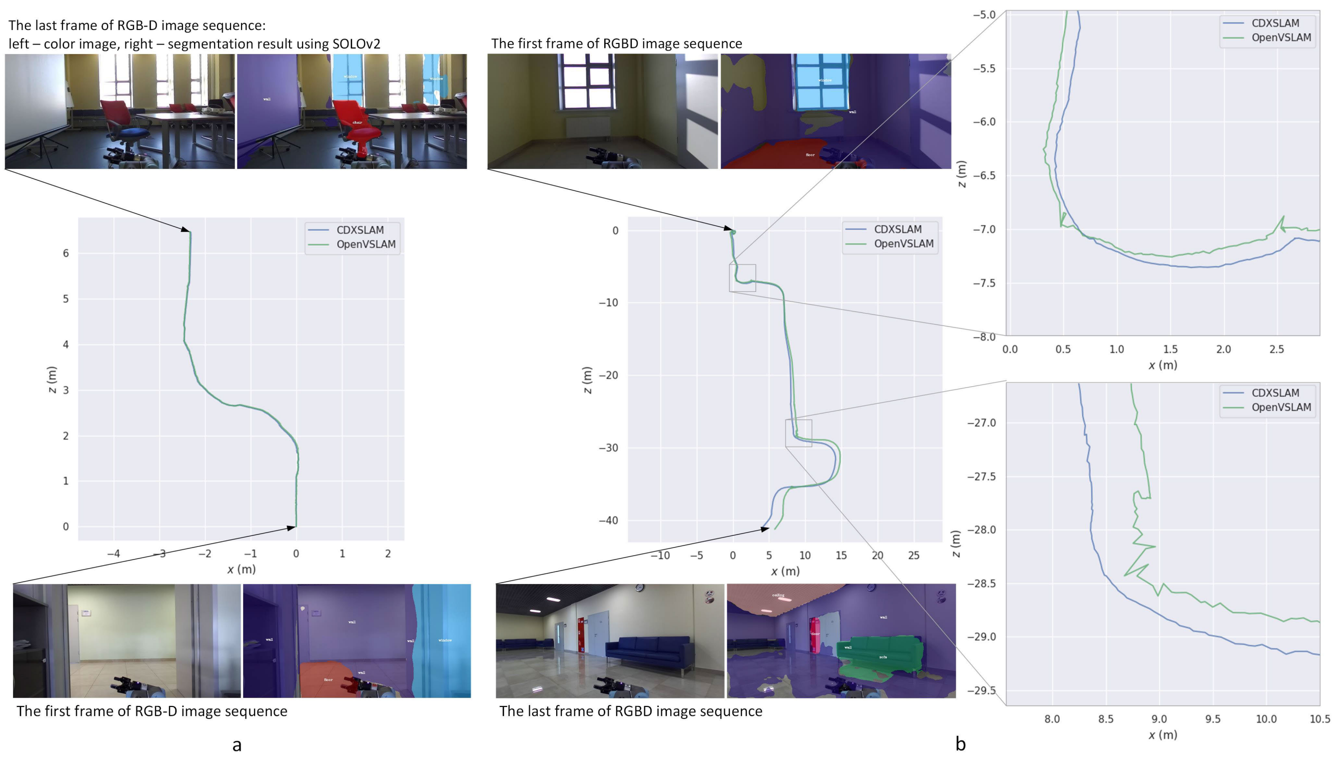


FIGURE 9. Experiment results with Clearpath Husky chassis in MIPT campus.

task with our framework, and as the exploration task solution, we took as a baseline the SemExp [16] global policy module. To speed up the training, for all experiments below we took one scene from our dataset with multiple episodes with different start and goal points. The results on object navigation task are present here (Figure 10). The result we demonstrate is sufficient to quickly find the object, but it is still low in our opinion. We have future plans to develop a better solution to this problem, and the HISnav framework is able to do it.

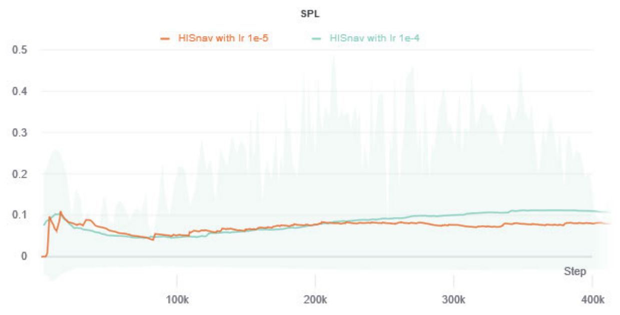


FIGURE 10. HISNav at object navigation task.

We also tested our framework on more simple point navigation task. It's showed identical behavior on the first steps, but then rapidly speed up, so we hope that on object navigation task it should behave the same. In our experiment, we compare vanilla PPO, RND, and HISNav framework at point navigation task with the presence of sensor Gaussian noise ($mean = 0, \sigma = 1, intensity = 0.05$) and action noise ($mean = 0, \sigma = 1, intensity = 0.2$).

As a result (Figure 11), RND showed slightly worse than vanilla PPO. RND's intrisic rewards showed no gain in the beginning of the training process and it decays to minimum at the end. Our method at the end converge to optimal SPL of 0.7 as PPO and RND, but achieve 0.6 as twice as fast, at 3M rather than 6M.

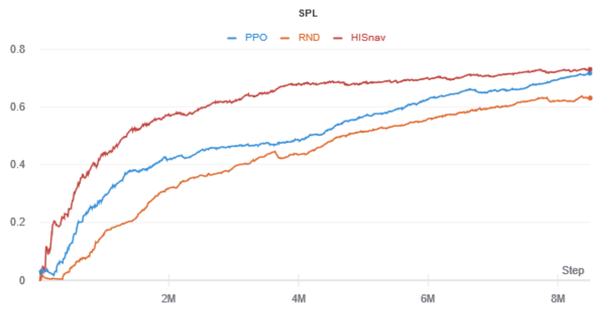


FIGURE 11. PPO vs RND vs HISNav at point navigation task.

E. REAL-TIME PERFORMANCE OF THE PROPOSED NAVIGATION FRAMEWORK

The performance of various modules developed by HISNav Framevork was tested on two hardware platforms and summarized in Table 5. Specification of the server and ground

TABLE 5. Mean inference time of the modules of HISNav Framework, s. For CDXSLAM we report total inference time as well as inference time of keypoint detector HFNet and time of tracking separately (in parenthesis).

Module	Server platform	Ground robot
Instance segmentation module		
Blendmask	0.020	0.068
SOLOv2	**0.018**	**0.066**
YOLACT++	0.026	0.121
Mask R-CNN$_{det2}$	0.031	0.119
Mask R-CNN$_{mmdet}$	0.034	0.127
SLAM module		
CDXSLAM (bow)	0.119 (0.011+0.108)	0.107 (0.078+0.029)
CDXSLAM (control)	**0.065** (0.011+0.054)	0.095 (0.078+0.017)
OpenVSLAM (bow)	0.108	0.061
OpenVSLAM (control)	0.091	**0.058**
RL module		
Policy NN	0.005	0.009

robot platforms are described in subsection VI-A. It shows that the proposed approach provides a fairly high performance on both server and mobile hardware platforms.

The best speed between instance segmentation models has SOLOv2: about 15 frames per second (fps) on the ground robot platform and 55 fps on server hardware. Considering this and the high segmentation quality the SOLOv2 model is the most suitable for use as part of the proposed HISNav Framework.

CDXSLAM performance includes the performance of HF-Net - neural network for extraction of global image descriptors and keypoints with their descriptors and camera pose tracking performance. HF-Net is faster on the server due to faster GPU. However, tracking is faster on the ground robot because its central processor has higher frequency. The latter also explains the higher speed of OpenVSLAM on the robot than on the server.

Results show that proposed control model increases the performance of SLAM methods. Keypoints matching can be performed faster with the motion model than with Bag Of Words. Moreover, the improvement is critical since it allows us to achieve real-time operation at a camera frequency of 10 Hz.

Policy neural network used in reinforcement learning module of HISNav Framework is fast both on server (200 fps) and on the ground robot platform (111 fps).

VII. CONCLUSION AND DISCUSSION

In this article, we have proposed a new HISNav framework for solving the problem of indoor navigation through an RGB-D camera in the presence of noise, the source of which can be the sensors and actuators of the robot. Our approach implements the sim2real paradigm, following which we first pre-train various modules of our framework using a simulation environment, and then load the resulting models onto a real robot. Our framework includes modern neural network methods of instance segmentation, localization, and mapping, which allow the robot to solve its subtasks in real-time. To speed up the exploration of the environment and the search for an object of interest, we have developed a new hierarchical method of reinforcement learning with automatic generation of subgoals, which allows us not to create a complex reward function. To pre-train all subsystems, we developed a new HISNav dataset using the photo-realistic Habitat environment on the basis of which we were able to train models of segmentation, localization, and agent strategies that are resistant to external noise.

Experimental studies have shown the promise of our framework. With the necessary refinement, we have shown that modern neural network architectures and reinforcement learning methods are suitable for solving important robotic problems in real-time. Nevertheless, it should be noted that further research is needed in this direction. Even though we have shown the advantages of our architecture over the existing learning solutions of the object navigation problem in terms of work and learning speed, the solution's quality should be significantly improved.

There are several methods that are less resistant to noise and less computationally efficient, but which, in some cases, can reduce the distance traveled when searching for objects. Increasing the SPL metric and the necessary refinement of our framework for this is a direction for future research. We believe that the critical point here should be deeper integration of the RL and SLAM methods, which we have already demonstrated by integrating the dynamics model into the localization algorithms.

REFERENCES

[1] M. B. Alatise and G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39830–39846, 2020.

[2] M. Ghaffari Jadidi, J. Valls Miro, and G. Dissanayake, "Gaussian processes autonomous mapping and exploration for range-sensing mobile robots," *Auto. Robots*, vol. 42, no. 2, pp. 273–290, Feb. 2018.

[3] B. Fang, J. Ding, and Z. Wang, "Autonomous robotic exploration based on frontier point optimization and multistep path planning," *IEEE Access*, vol. 7, pp. 46104–46113, 2019.

[4] E. I. Al Khatib, M. A. K. Jaradat, and M. F. Abdel-Hafez, "Low-cost reduced navigation system for mobile robot in indoor/outdoor environments," *IEEE Access*, vol. 8, pp. 25014–25026, 2020.

[5] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. New York, NY, USA: Pearson, 4th ed., 2020.

[6] X. Mu, B. He, X. Zhang, T. Yan, X. Chen, and R. Dong, "Visual navigation features selection algorithm based on instance segmentation in dynamic environment," *IEEE Access*, vol. 8, pp. 465–473, 2020.

[7] Y. Zhang, R. Ge, L. Lyu, J. Zhang, C. Lyu, and X. Yang, "A virtual end-to-end learning system for robot navigation based on temporal dependencies," *IEEE Access*, vol. 8, pp. 134111–134123, 2020.

[8] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A platform for embodied AI research," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9339–9347.

[9] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, "ObjectNav revisited: On evaluation of embodied agents navigating to objects," 2020, *arXiv:2006.13171*. [Online]. Available: http://arxiv.org/abs/2006.13171

[10] D. Mishkin, A. Dosovitskiy, and V. Koltun, "Benchmarking classic and learned navigation in complex 3D environments," 2019, *arXiv:1901.10915*. [Online]. Available: http://arxiv.org/abs/1901.10915

[11] R. Mur-Artal and J. D. Tardos, "Orb-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, p. 1255–1262, Oct. 2017.

[12] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," 2016, *arXiv:1611.01779*. [Online]. Available: http://arxiv.org/abs/1611.01779

[13] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "DD-PPO: Learning near-perfect PointGoal navigators from 2.5 billion frames," 2019, *arXiv:1911.00357*. [Online]. Available: http://arxiv.org/abs/1911.00357

[14] D. Singh Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural SLAM," 2020, *arXiv:2004.05155*. [Online]. Available: http://arxiv.org/abs/2004.05155

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347

[16] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," 2020, *arXiv:2007.00643*. [Online]. Available: http://arxiv.org/abs/2007.00643

[17] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Jun. 2017, pp. 2961–2969.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[20] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.

[21] D. Bolya, C. Zhou, F. Xiao, and Y. Jae Lee, "YOLACT++: Better real-time instance segmentation," 2019, *arXiv:1912.06218*. [Online]. Available: http://arxiv.org/abs/1912.06218

[22] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and P. Luo, "PolarMask: Single shot instance segmentation with polar representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12193–12202.

[23] S. Peng, W. Jiang, H. Pi, X. Li, H. Bao, and X. Zhou, "Deep snake for real-time instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8533–8542.

[24] J. Liang, N. Homayounfar, W.-C. Ma, Y. Xiong, R. Hu, and R. Urtasun, "PolyTransform: Deep polygon transformer for instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9131–9140.

[25] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan, "BlendMask: Top-down meets bottom-up for instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8573–8581.

[26] Y. Wang, Z. Xu, H. Shen, B. Cheng, and L. Yang, "CenterMask: Single shot instance segmentation with point representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9313–9321.

[27] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "SOLO: Segmenting objects by locations," 2019, *arXiv:1912.04488*. [Online]. Available: http://arxiv.org/abs/1912.04488

[28] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "SOLOv2: Dynamic and fast instance segmentation," 2020, *arXiv:2003.10152*. [Online]. Available: http://arxiv.org/abs/2003.10152

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[30] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 764–773.

[31] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2403–2412.

[32] Z. Cai and N. Vasconcelos, "Cascade R-CNN: High quality object detection and instance segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Nov. 28, 2019, doi: 10.1109/TPAMI.2019.2956516.

[33] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[34] J. Zubizarreta, I. Aguinaga, and J. M. M. Montiel, "Direct sparse mapping," *IEEE Trans. Robot.*, vol. 36, no. 4, p. 1363–1370, Aug. 2020.

[35] N. Yang, L. von Stumberg, R. Wang, and D. Cremers, "D3VO: Deep depth, deep pose and deep uncertainty for monocular visual odometry," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1281–1292.

[36] T. Shen, Z. Luo, L. Zhou, H. Deng, R. Zhang, T. Fang, and L. Quan, "Beyond photometric loss for self-supervised ego-motion estimation," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6359–6365.

[37] H. Zhan, C. Saroj Weerasekera, J. Bian, and I. Reid, "Visual odometry revisited: What should be learnt?" 2019, *arXiv:1909.09803*. [Online]. Available: http://arxiv.org/abs/1909.09803

[38] R. Muñoz-Salinas and R. Medina-Carnicer, "UcoSLAM: Simultaneous localization and mapping by fusion of keypoints and squared planar markers," *Pattern Recognit.*, vol. 101, May 2020, Art. no. 107193.

[39] M. Labbé and F. Michaud, "RTAB-Map as an open-source Lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *J. Field Robot.*, vol. 36, no. 2, pp. 416–446, 2019.

[40] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," 2020, *arXiv:2007.11898*. [Online]. Available: http://arxiv.org/abs/2007.11898

[41] S. Sumikura, M. Shibuya, and K. Sakurada, "OpenVSLAM: A versatile visual SLAM framework," in *Proc. 27th ACM Int. Conf. Multimedia (MM)*, New York, NY, USA, 2019, pp. 2292–2295.

[42] P.-E. Sarlin, F. Debraine, M. Dymczyk, R. Siegwart, and C. Cadena, "Leveraging deep visual descriptors for hierarchical efficient localization," 2018, *arXiv:1809.01019*. [Online]. Available: http://arxiv.org/abs/1809.01019

[43] A. Forechi, T. Oliveira-Santos, C. Badue, and A. F. D. Souza, "Visual global localization with a hybrid WNN-CNN approach," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–9.

[44] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12716–12725.

[45] J. Tang, L. Ericson, J. Folkesson, and P. Jensfelt, "GCNV2: Efficient correspondence prediction for real-time SLAM," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3505–3512, Jul. 2019.

[46] J. Lee, S. Hwang, K. Lee, W. J. Kim, J. Lee, T.-Y. Chung, and S. Lee, "AD-VO: Scale-resilient visual odometry using attentive disparity map," 2020, *arXiv:2001.02090*. [Online]. Available: http://arxiv.org/abs/2001.02090

[47] D. Li, X. Shi, Q. Long, S. Liu, W. Yang, F. Wang, Q. Wei, and F. Qiao, "DXSLAM: A robust and efficient visual SLAM system with deep features," 2020, *arXiv:2008.05416*. [Online]. Available: http://arxiv.org/abs/2008.05416

[48] J. Baltes, D.-W. Kung, W.-Y. Wang, and C.-C. Hsu, "Adaptive computational SLAM incorporating strategies of exploration and path planning," *Knowl. Eng. Rev.*, vol. 34, p. e23, 2019.

[49] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niebner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D data in indoor environments," in *Proc. Int. Conf. 3D Vis. (3DV)*, Oct. 2017, pp. 667–676.

[50] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 573–580.

[51] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[52] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3354–3361.

[53] H. Zhan, C. Saroj Weerasekera, J. Bian, and I. Reid, "Visual odometry revisited: What should be learnt?" 2019, *arXiv:1909.09803*. [Online]. Available: http://arxiv.org/abs/1909.09803

[54] M. Grupp. (2017). *EVO: Python Package for the Evaluation of Odometry and SLAM*. [Online]. Available: https://github.com/MichaelGrupp/evo

[55] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.

[56] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018, *arXiv:1802.09477*. [Online]. Available: http://arxiv.org/abs/1802.09477

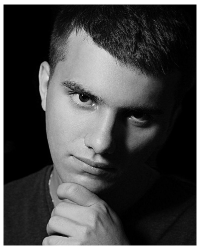
[57] A. Levy, G. Konidaris, R. Platt, and K. Saenko, "Learning multi-level hierarchies with hindsight," 2017, *arXiv:1712.00948*. [Online]. Available: http://arxiv.org/abs/1712.00948

[58] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. (2019). *Detectron2*. [Online]. Available: https://github.com/facebookresearch/detectron2

[59] K. Chen *et al.*, "MMDetection: Open MMLab detection toolbox and benchmark," 2019, *arXiv:1906.07155*. [Online]. Available: http://arxiv.org/abs/1906.07155

[60] X. Shi, D. Li, P. Zhao, Q. Tian, Y. Tian, Q. Long, C. Zhu, J. Song, F. Qiao, L. Song, Y. Guo, Z. Wang, Y. Zhang, B. Qin, W. Yang, F. Wang, R. H. M. Chan, and Q. She, "Are we ready for service robots? The OpenLORIS-scene datasets for lifelong SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 3139–3145.

[61] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," 2018, *arXiv:1810.12894*. [Online]. Available: https://arxiv.org/abs/1810.12894



ALEKSEY STAROVEROV was born in Russia. He received the M.S. degree from Bauman Moscow State Technical University, in 2019. He is currently pursuing the Ph.D. degree in computer science with the Moscow Institute of Physics and Technology. His research thesis is the methods and algorithms for the automatic determination of subgoals in a reinforcement learning problem for robotic systems. Since October 2019, he has been working as a Researcher with the Cognitive Dynamic System Laboratory, Moscow Institute of Physics and Technology. His research interests include reinforcement learning, deep learning, and robotic systems.

DMITRY A. YUDIN was born in Belgorod, Russia, in 1988. He received the Diploma degree in automation engineering of technological processes and production, in 2010, and the Ph.D. degree in computer science from Belgorod State Technological University named after V. G. Shukhov, Belgorod, in 2014.

From 2009 to 2019, he was a Researcher and an Assistant Professor with the Technical Cybernetics Department, Belgorod State Technological University named after V. G. Shukhov. Since 2019, he has been a Postdoctoral Researcher with the Cognitive Dynamic System Laboratory and the Head of the Intelligent Transport Laboratory, Moscow Institute of Physics and Technology, Moscow, Russia. He is the author of three books and more than 100 articles. His research interests include computer vision, deep learning, and robotics.

Dr. Yudin was a recipient of the IEEE Czechoslovakia Section Young Scientist Paper Award in 2017 and the Grant of the President of the Russian Federation for state support of young Russian scientists from 2017 to 2018.

ILYA BELKIN received the B.S. degree in computer science from the Moscow Institute of Physics and Technology, Moscow, Russia, in 2019, where he is currently pursuing the master's degree in methods and technologies of artificial intelligence.

From June 2018 to September 2018, he was a Data Science Intern with the Huawei Russian Research Center. From October 2018 to July 2019, he was a Computer Vision Intern with ABBYY Research and Development Russia. From June to the present, he is a Researcher and a Developer with the Intelligent Transport Laboratory, Moscow Institute of Physics and Technology. His research interests include computer vision, deep learning, and robotics.

VASILY ADESHKIN was born in Nizhnyaya Toya, Khakas Republic, Russia. He received the B.S. degree in applied mathematics and physics from the Moscow Institute of Physics and Technology, Moscow, Russia, in 2020, where he is currently pursuing the master's degree in methods and technologies of artificial intelligence.

Since November 2019, he has been an Intern Data Scientist at the Cognitive Dynamic System Laboratory, Moscow Institute of Physics and Technology. His research interests include computer vision and deep learning.

YAROSLAV K. SOLOMENTSEV received the M.S. degree in computer science from the Moscow Institute of Physics and Technology, Moscow, Russia, in 2020, where he is currently pursuing the Ph.D. degree in computer science.

His research interests include computer vision, and the methods of simultaneous localization and mapping.

ALEKSANDR I. PANOV received the M.S. degree in computer science from the Moscow Institute of Physics and Technology, Moscow, Russia, in 2011, and the Ph.D. degree in theoretical computer science from the Institute for Systems Analysis, Moscow, in 2015.

Since 2010, he has been a Research Fellow with the Federal Research Center "Computer Science and Control," Russian Academy of Sciences. Since 2018, he has been the Head of the Cognitive Dynamic System Laboratory, Moscow Institute of Physics and Technology. He is the author of three books and more than 90 articles. His research interests include behavior planning, reinforcement learning, semiotics, and robotics.