IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 May 2022, accepted 4 June 2022, date of publication 13 June 2022, date of current version 8 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3182803

Hierarchical Landmark Policy Optimization for Visual Indoor Navigation

ALEKSEI STAROVEROV[1,2,3] **AND ALEKSANDR I. PANOV**[1,3]
[1]Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 119333 Moscow, Russia
[2]Moscow Institute of Physics and Technology, 141701 Dolgoprudny, Russia
[3]AIRI, 111402 Moscow, Russia

Corresponding author: Aleksei Staroverov (staroverov.av@phystech.edu)

This work was supported in part by the Russian Science Foundation under Grant 20-71-10116.

ABSTRACT In this paper, we study the problem of visual indoor navigation to an object that is defined by its semantic category. Recent works have shown significant achievements in the end-to-end reinforcement learning approach and modular systems. However, both approaches need a big step forward to be robust and practically applicable. To solve the problem of insufficient exploration of the scenes and make exploration more semantically meaningful, we extend standard task formulation and give the agent easily accessible landmarks in the form of the room locations and those types. The availability of landmarks allows the agent to build a hierarchical policy structure and achieve a success rate of 63% on validation scenes in a photo-realistic Habitat simulator. In a hierarchy, a low level consists of separately trained RL skills and a high level deterministic policy, which decides which skill is needed at the moment. Also, in this paper, we show the possibility of transferring a trained policy to a real robot. After a bit of training on the reconstructed real scene, the robot shows up to 79% SPL when solving the task of navigating to an arbitrary object.

INDEX TERMS Navigation, reinforcement learning, robotics, neural networks, complex indoor environments.

I. INTRODUCTION

Autonomous navigation in a semantically extensive environment is one of the significant components in building intelligent robotic systems.

This article focuses our direction on indoor robots and their ability to fulfill the user's requests by moving in the previously unseen environment to target objects. Classically, navigation problems are solved with simultaneous localization and mapping (SLAM) methods [1]. As a result, these methods generate an obstacle map, and the planners [2], [3] build upon it the collision-free path to the goal.

These classical approaches show convincing performance when provided with a multimodal input [4] in an environments where signal is not blocked [5]. On the other side, with the appearance of fast performing simulators [6]–[8], large photo-realistic 3D datasets [9]–[11] and with the latest progress of learned methods, training virtual robots in simulation has gets a lot of interest in recent years. The learning approaches, like Reinforcement Learning (RL) [12], could adapt to almost any task in a simulator. The downside of RL is that it needs to be specifically trained to each task setup from scratch, and these algorithms are hard to deploy in a real navigation scenario. [13]

This paper aims to overcome this issue and show that RL can be practically usable in real-world conditions with sensor noise present in a challenging navigation task of finding objects.

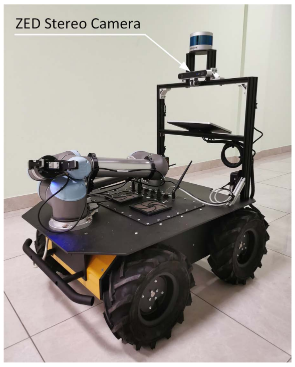


FIGURE 1. Ground robot platform based on the clearpath husky chassis with a ZED camera. We used it to evaluate our results in real-world scenarios.

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

To achieve this, we first train our agent in a high framerate simulator Habitat [8] and then smooth transfer policy on a real robot (Fig. 1). Habitat is an open-source simulator that consume a photo-realistic Matterport [10] dataset (and many others) and build an environment upon it.

Habitat authors also claim that mobile operation in three-dimensional environments is a primary topic of study in Artificial Intelligence [14] and start a Habitat Challenge to make progress in this field.

We choose to take ObjectNav task formulation from the Habitat Challenge track as a test platform. ObjectNav is defined as the task of navigating to a semantic type of object in an unseen environment [15]. During the challenge in 2021, the end-to-end Reinforcement Learning (RL) algorithm became the state-of-the-art solution and showed a success rate of 23% on a test-standard phase [16]. This indicates that the ObjectNav task is far from being solved or stagnated. Further increasing metrics is impossible due to unsolved episodes being extensive areas that are hard to explore without prior information of its semantical structure. We assume that such information should be easy to annotate by humans and remains unchanged in the areas with the often moving objects. Therefore, the agent should have a hierarchical structure to use spatial understanding efficiently.

To summarize, our contribution includes:

- **Task formulation with landmarks**. As humans solve indoor exploration tasks to find the object, they strongly rely on a room's understanding of a concrete scene and can predict the type of objects inside. To allow the agent to learn this concept, we gave the agent the landmarks ($\mathcal{G}$) in the form of all rooms center coordinates and their kind (Fig. 2). This information could be easily annotated by a human as opposed to the map. Note that the obstacle map or objects inside any rooms are still unknown to the agent.
- **Dividing policy into a set of skills**. As an agent's policy, we distinguished three basic skills: navigation to the point, exploration of the nearby area, and reaching the seen object. The agent was separately trained in all of these skills using the policy optimization approach.
- **Hierarchical structure**. To efficiently use the given list of landmarks, we have built the skill selector module that at each time navigates the agent to the most promising landmark zone by selecting one of the presented skills.
- **Smooth policy transfer to new real-world scenes**. To make policy transfer possible and predictable, we first 3D reconstructed our scene into a simulator. After a small stage of neural networks adaptation to it and being convinced that the agent navigates safely in it, we conducted a real test and got an agent behavior similar to the one in the simulator. Since our robot relies only on an RGB image, the depth and semantic were received by neural net methods.

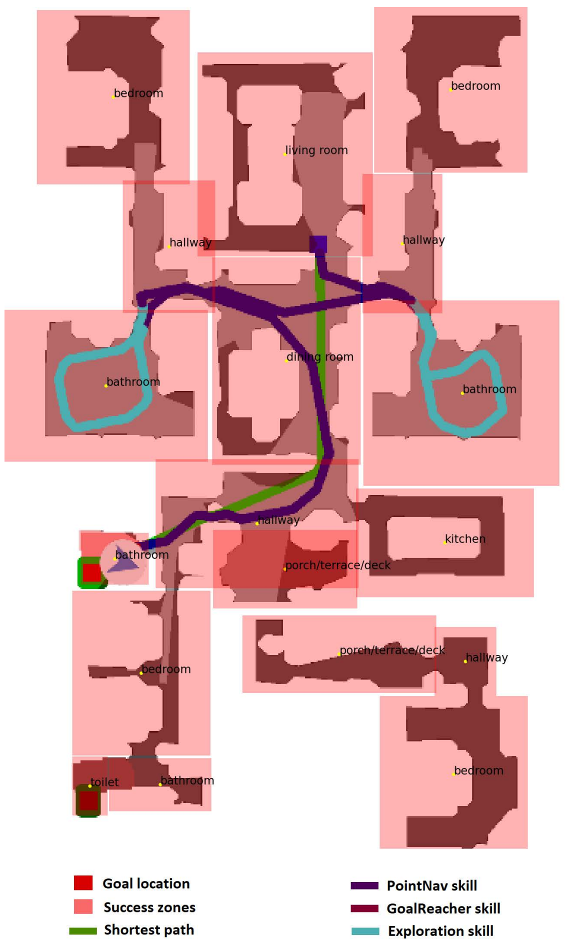


FIGURE 2. Example of the resulted agent's trajectory with the visualization of landmarks.

II. RELATED WORK

As the most navigational tasks, the ObjectNav task could be solved by a SLAM and a deterministic planner methods. As a results, the agent builds an occupancy map and a collision-free path to the goal. Despite the fact that the goal object's coordinates are unknown, methods such as a Frontier-based exploration (FBE) [17] are often used. A frontier is defined as the boundary between the explored free space and the unexplored space. Frontier-based exploration essentially samples points on this frontier as goals to explore the space. if the agent during this exploration sees a goal type of object it navigates to it directly. As one of our baselines (ExploreTillSeen), we followed this strategy and builded a two RL-based agents, one that explore the area and the other one that navigates to a goal object when the agent sees it.

A big breakthrough of the learning approaches in navigation tasks was the DDPPO method [18], which as its core used Proximal Policy Optimization [19]. Without any mapping or planning modules, DDPPO at the PointNav task was able to perform 2.5 billion steps in the environment and solve the task at human-level performance. Authors achieve that by giving the agent inputs directly from an RGB-D camera and a GPS+Compass sensor. To optimize the training process, the authors investigated the most efficient neural network architecture [20], which we also use in our approach.

Previous works have shown that at ObjectNav task, the pure end-to-end RL algorithms that use vanilla visual and recurrent modules perform poorly due to overfitting and sample inefficiency. The authors of Auxiliary task RL method [16] partialy solved that by adding auxiliary learning tasks and an exploration reward during training phase.

Another promising approach to solve the ObjectNav task is to mix analytic and learned components and operate on explicit spatial maps of the environment. Such a combination of classical and learned methods, SemExp [21] has shown the best result at ObjectNav during Habitat Challenge 2020. Authors use a deterministic map module and divide a policy into a global, that by planning on a map output a short-term subgoal, and a local policy, that pursues that subgoal. Inspiring this work, we also build a two-level policy, but our low level consists of several independent skills and high-level switches between them depending on what the agent needs to do at a given time.

The idea of using landmarks as a region of interest in navigation tasks is also described in [22], [23]. The first method, HIGL, also divided policy into high-level and low-level, where the high level generates a subgoal toward landmarks and the low level reaches it. The sample of landmarks was based on the "coverage" and "novelty" criteria from the previously visited states. In general, we found this idea suitable to us, but instead of sampling from previous trajectories (the agent must navigate in previously unseen scenes), we utilize the semantic structure of the indoor scenes and define landmarks as the coordinates of the room's center. The policy of sampling from landmarks is based on the distance to landmarks and statistics of objects relative to rooms type.

III. TASK SETUP

The indoor object navigation task is defined as the task of navigating to an object (specified by semantic label) in a previously unseen environment [24]. In practice, the agent is initialized at a random pose in an environment and aims to find an instance of an object category $I_{goal} \in \{c_1, c_2, \ldots, c_{20}\}$ (for example, a *couch*) by navigating to it.

This interaction is formally described by the Markov decision process (MDP), which is defined by sets of states S and actions A (*forward*, *turn left*, *turn right*, and *stop*), the distribution of the initial states $p(s_0)$, the reward function $r : S \times A \to \mathbb{R}$, the transition probabilities $p(s_{t+1} \mid s_t, a_t)$, the termination probabilities $T(s_t, a_t)$, and the discount factor $\gamma \in [0, 1]$. The agent receives a semantic mask of a goal-type of the object through the semantic segmentation module ($I_{sem} = \Phi_{semantic}(I_{RGB}, I_{goal})$).

During the evaluation process, the agent can only use the input from the RGB-D camera (I_{RGBD}), the GPS+Compass sensor (I_{GPS}), and a list of landmarks ($\mathcal{G}$) for navigation. A list of landmarks contains all center coordinates of rooms and their type with no information about the map, what objects are inside, or how to navigate to those rooms.

Evaluation occurs when the agent selects the *stop* action. As a metric, the Success rate weighted by Path Length (SPL) and the Success rate are used. SPL is computed to the object instance closest to the agent start location.

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

where l_i is the length of the shortest path between the goal and the target for an episode; p_i is the length of the path taken by the agent in an episode.

Thus, if an agent spawns very close to *chair1* but stops at a distant *chair2*, it will achieve 100% success (because it found a "chair") but a fairly low SPL (because the path to *chair2* is much longer than that to the *chair1*). More specifically, an episode is deemed successful if the agent is calling the *stop* action within 1.0m Euclidean distance from any instance of the target object category, and an oracle can view the object from that stopping position by turning the agent or looking up/down.

IV. METHODS

We propose a landmark-based modular framework (Fig. 3) for navigation to object goal (I_{goal}) – Hierarchical Landmark Policy Optimization (HLPO). The framework consists of three main modules: skill selector $\pi_{selector}$, functions for data preprocessing $\Phi_{semantic}$ and Φ_{depth}, and skill policies $\{\pi_{explore}, \pi_{reacher}, \pi_{pointnav}\}$.

The skill selector module is a deterministic strategy that sets a subtask at each step and selects the skill required to complete it.

The process of managing skills could be treated as a policy and integrated into the global policy module. Typically, the HRL methods can learn a two-level policy Π_1. Each level of policy learns $\pi_i : S_i, G_i \to A_i$ where G_i is the set of possible sub-goals. To learn these policies π_i, the set of MDPs U_0, U_1, in which $U_k = (S, G, A, T, R, \gamma)$, is used. However, learning multiple levels of policies in parallel is problematic because learning is inherently unstable. For the object goal task, the sequence of skills can be formulated explicitly.

During the training phase $\pi_{selector}$ analyzes all Matterport scenes to connect the object types to the type of rooms and makes a statistic out of it (Fig. 4). Thanks to it, we know that, for example, a sofa is located with a probability of 70% in the living room, and 10% each in the bedroom, living room, meeting room. During the execution of the episode, it works as follows. For each given landmark, at each step we calculate a visit importance coefficient equal to the probability of finding the object of the target type in the given room type, divided by the Euclidean distance to the center of the given room. $\pi_{selector}$ at each step chooses the Landmark with the highest importance factor and determines which skill should be used to complete the task. If the agent is outside the selected landmark, then $\pi_{selector}$ chooses $\pi_{pointnav}$ to reach it. Once

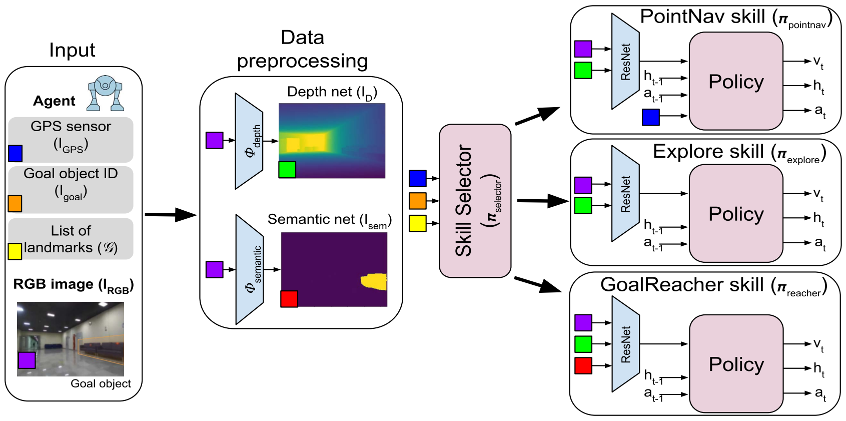


FIGURE 3. Hierarchical landmark policy optimization (HLPO) scheme. Our proposed approach consists of three main blocks: data prepossessing, a skill selector, and skill policies. Multicolored squares at the bottom of the elements mean what data it denotes (at the bottom left) and consumes (at the bottom right).

Algorithm 1 HLPO

```
1:  Given:
      $\pi_{selector}$: Skill selector,
      $\pi_{explore}$: Explore policy,
      $\pi_{reacher}$: GoalReacher policy,
      $\pi_{pointnav}$: PointNav policy,
      $\Phi_{semantic}$: Semantic segmentation model,
      $\Phi_{depth}$: Depth model,

2:  Input:
      $I_{RGBD}$: RGBD image from the camera,
      $I_{GPS}$: X,Y coordinates relative to the start point,
      $I_{goal}$ Goal type of object,
      $\mathcal{G}$: List of landmarks.

3:  while episode not ended do
4:      $I_{skill\_type}, I_{room\_cord} \leftarrow \pi_{selector}(\mathcal{G}, I_{goal}, I_{GPS})$
5:      $I_{sem} \leftarrow \Phi_{semantic}(I_{RGB}, I_{goal})$
6:      $I_D \leftarrow \Phi_{depth}(I_{RGB})$
7:      if $I_{skill\_type} = PointNav$ then
8:          $a \leftarrow \pi_{pointnav}(I_{RGBD}, I_{GPS}, I_{room\_cord}, a_{prev})$
9:      end if
10:     if $I_{skill\_type} = Explore$ then
11:         $a \leftarrow \pi_{explore}(I_{RGBD}, a_{prev})$
12:     end if
13:     if $I_{skill\_type} = GoalReacher$ then
14:         $a \leftarrow \pi_{reacher}(I_{RGBD}, I_{sem}, a_{prev})$
15:     end if
16:     Execute action $a$ in the environment
17: end while
```

the agent enters the Landmark zone, $\pi_{selector}$ activates $\pi_{explore}$ until the agent. will not leave the Landmark zone. If at any time $\Phi_{semantic}$ sees an object of the target type, then $\pi_{selector}$ will activate the $\pi_{reacher}$ skill to reach it. (Algorithm 1)

The data preprocessing module is two neural nets that do semantic segmentation and depth reconstruction.

Our algorithm uses the semantic segmentation model to get the mask of the goal object, which is input for policy. We have taken the SOLOv2 [25] approach for those categories that correspond to the Coco dataset [26] and RedNet [27] for others. RedNet was trained on collected images from the MP3D dataset and took as input an RGBD image. The SOLOv2 model uses weights pretrained by authors and rely only on an RGB image. SOLOv2 output far fewer noises during our experiments, which is crucial for our algorithm because of skill separation. If false-positive noise appears, our approach policy switcher fires at the wrong time.

Though we use a ground truth depth sensor for all train and test phases at the simulator, we do not have that option at the Husky robot, so we reconstructed it from an RGB image [28].

The skill policies module takes as input agent's observation and outputs the action that pursues current needed skill. For our task, we identified three skills: point navigation, exploration, and goal reacher. In our experiments, the PointNav skill executes 50% of the time during the episode, Explore skill 22%, and Goalreacher skill 28%.

The PointNav skill was trained to reach a landmark zone (room of interest). The reward is proportional to the shortened distance to the goal coordinate. As an input it takes an RGBD image and the polar coordinates of the center of landmark zone. It's main goal is to perform 'stop' action when an agent is within 0.1m distance from the gool coordinates. (Fig. 5)

The Exploration skill determines the task's success more than others, especially when we have no landmarks. We trained an RL policy that effectively explores the nearby area (Fig. 7), so it fits perfectly to explore the room completely but has lower percentage coverage at big scenes. As a reward, we give +1 each time the agent steps into the new square meter zone.

The GoalReacher skill was trained to reach the goal-type object when the semantic sensor sees it (Fig. 6). It's main goal is to perform 'stop' action when an agent is within 1m distance from the goal-type object. To avoid overfitting, we generate episodes by initializing agents to a random place

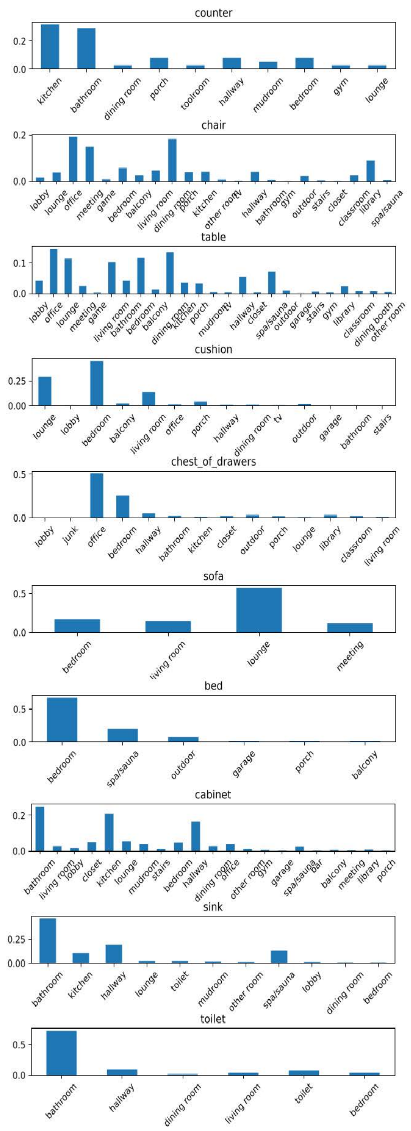


FIGURE 4. Statistics on the distribution of objects by rooms, obtained during the training of the skill selector module.

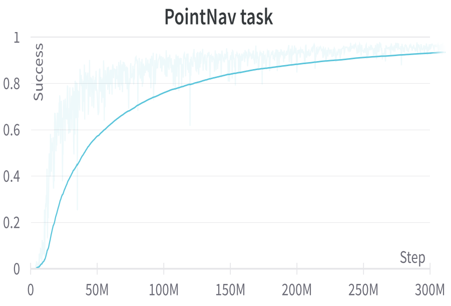


FIGURE 5. Success rate vs. steps on a training phase of PointNav skill. Higher is better.

and then ask them to reach the farthest seen object type. The reward is also proportional to the shortened distance to the seen goal-type object.

The agent's goal is to find the policy π that maximizes the expected return $\mathbb{E}_{\pi}[R_0 \mid s_0])$. The expectation is taken over

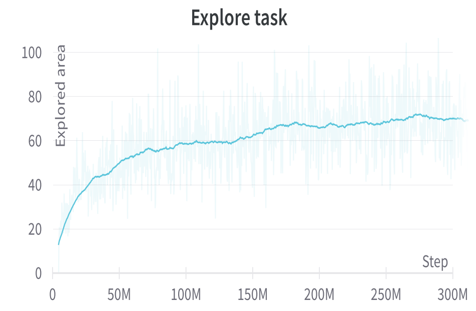


FIGURE 6. Explored area (m^2) vs. steps on a training phase of Exploration skill. Higher is better.

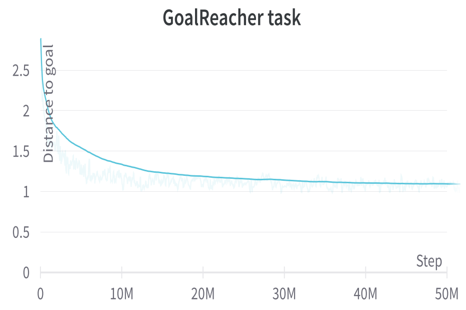


FIGURE 7. Distance to the goal type object (m) vs. steps on a training phase of GoalReacher skill. Closer to 1.0m is better.

the initial state distribution, the policy, and the environment transitions according to the dynamics specified above. The action-value function (Q-function) of a given policy π is defined as $Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[R_t \mid s_t, a_t]$. The state-value function (V-function) is defined as $V^{\pi}(s_t) = \mathbb{E}_{\pi}[R_t \mid s_t]$. The advantage is defined as $A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t - V^{\pi}(s_t)$ and informs if action a_t is better than the average action the policy π takes in the state s_t.

De facto, the policy and the value functions are represented as two neural networks. The first represents the current policy π, and the value network approximates the current policy's value function $V \approx V^{\pi}$.

A policy neural network is a two-head network, one for the action distribution (actor) and the other for the action value estimation (critic). The actor stream is one FC layer that outputs logits for each out of the four actions. An action to execute is picked as a categorical distribution of that logit. The critic stream is an FC layer that outputs value for the given state.

To compute the return, we use a generalized advantage estimator (GAE) with $\gamma = 0.99$ and $\tau = 0.95$.

For the policy loss function, we use the proximal policy optimization (PPO) [19], which still remains the SOTA solution in RL for the vast quantity of tasks. In our method, for training an RL agent, we use a gradient descent over a policy agent. Given a θ-parameterized policy π_θ and a set of trajectories collected with it (commonly referred to as a "rollout"), the agent updates π_θ as follows. Let $\hat{A}_t = R_t - \hat{V}_t$, be the estimate of the advantage, where $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ and $\hat{V}_t$ is the expected value of R_t, and $r_t(\theta)$ be the ratio of the probability of the action under the current policy and the

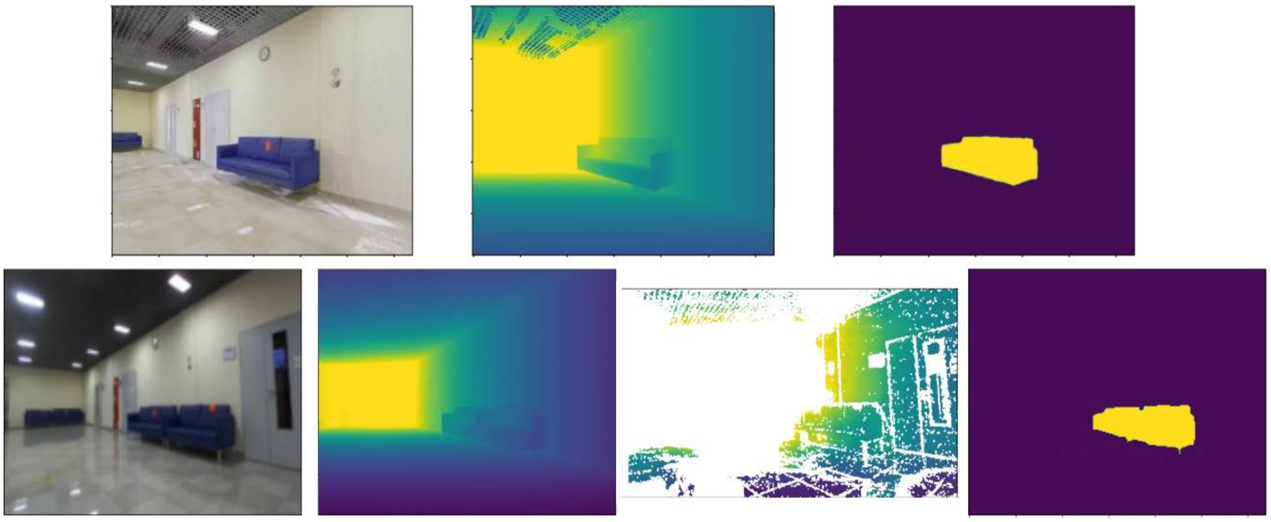


FIGURE 8. Top row is the agent view in a simulator. The middle image is the ground truth depth that was clamp to five meters. The bottom row is the real robot view. The second image is the depth obtained by the depth reconstruction module. The third image is shown to compare the quality of the neural net depth versus ZED camera depth. Both right images are the semantic mask of the couch class obtained by the semantic reconstruction module.

policy used to collect the rollout. The parameters are then updated by maximizing

$$J^{PPO}(\theta) = E_t\left[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right]. \tag{2}$$

As the parallelization method, we utilize the decentralized distributed proximal policy optimization (DD-PPO) way [18]. The synchronous method differs from the asynchronous method in that in the synchronous one, each worker collects experience from the environment and simultaneously updates its weights for the general model with all workers. The workers that take too long time to step at the simulator are interrupted so that the rest do not wait for them. As a general abstraction, this method implements the following: at step k, worker n has a copy of the parameters, θ_n^k, calculates the gradient, $\partial\theta_n^k$, and updates θ via

$$\theta_n^{k+1} = PU\left(\theta_n^k, AR(\nabla_\theta J^{PPO}(\theta_1^k), \ldots, \nabla_\theta J^{PPO}(\theta_N^k))\right), \tag{3}$$

where PU is any first-order optimization technique (gradient descent), and AR performs a reduction (mean) over all copies of a variable and returns the result to all workers.

V. EXPERIMENTAL SETUP

A. SETUP

To show the performance of our HLPO method at a Habitat simulator, we have singled out two datasets: test and validation. For the test dataset, we manually picked 100 episodes from 11 scenes that are complex and large enough to show that exploring them without landmark information results in low performance. These episodes have a medium geodesic distance to the closest goal, no less than 5m. As for indoor object categories, we took 10 to our experiments: a bed, a cabinet, a chair, a chest of drawers, a counter, a cushion, a sink, a sofa, a table, and a toilet. We ran all experiments three times and got the dispersion of the results no more than 0.05.

TABLE 1. Comparison of the different agents on training episodes.

Method	GT semantic Success	GT semantic SoftSPL	Learned semantic Success	Learned semantic SoftSPL
DDPPO	0.18	0.35	0.11	0.24
SemExp	0.24	0.26	0.11	0.17
Planning	0.31	0.26	0.15	0.18
Auxiliary RL	0.51	0.34	0.19	0.19
ExploreTillSeen	0.46	0.33	0.20	0.21
HLPO	**0.86**	**0.52**	**0.46**	**0.30**
HLPO (Noise)	0.85	0.46	0.45	0.28
HLPO (Map)	0.90	0.54	0.51	0.42

To demonstrate that our method outperforms the current techniques, we selected five baselines that are not use landmark information, but at the training phase, they had a goal to learn the semantical structure of the scenes natively.

- **DDPPO** - We trained an end-to-end DDPPO [18] algorithm with the input as the depth + GPS + GT semantic sensors with the reward proportional to the geodesic distance to the goal object closest to the start. As a backbone, ResNet50 and LSTM layer were used.
- **SemExp** - The SemExp [21] is the SOTA algorithm with the module structure that incorporates both RL and planning, which showed the best performance at Habitat challenge 2020. We use the author's weights, but because of the differences in goal objects' classes, we give the agent only the goal class and set the others as zero.
- **Planning** - A combination of planning modules that, at every step, update the obstacle map and explore it.

TABLE 2. Comparison of the different agents on validation episodes.

Method	GT semantic		Learned semantic	
	Success	SoftSPL	Success	SoftSPL
ExploreTillSeen	0.47	0.27	0.27	0.20
HLPO	**0.63**	**0.34**	**0.33**	**0.21**

When the semantic module sees the goal object, it gets spotted on a map, and the agent navigates to it through planning.

- **Auxiliary RL** - The generic learned policy [16] with the auxiliary learning tasks and an exploration reward. It is the SOTA end-to-end algorithm, which showed the best performance at Habitat challenge 2021.
- **ExploreTillSeen** - A combination of two RL skills: one explores the area (Explore skill) until the semantic sensor sees the target and the other RL follows the seen goal object (GoalReacher skill).

As a result, using only the Explore and GoalReacher skills approach showed similar performance to the SOTA methods, and the HLPO almost doubled the performance, which indicates that our choice of landmark information was right, and the agent itself could not learn it.

To prove that sensor and action noises do not spoil the performance, we added them at a training phase, so they are not a problem during validation (HLPO (Noise) row in Table 1), either. The HLPO (Map) showed that additional performance could be squeezed if the agent had access to a full map of the scene. But the gain is not that noticeable as the hardness of collecting the map.

At the validation phase, we take 1,126 episodes out of 2,195 from all 11 scenes. We filter episodes that do not contain at least three goal-type objects at the same floor level as the agent start (Table 2).

To prove the need for landmarks, we gave an example of the ExploreTillSeen versus HLPO trajectories (Fig. 12). The ExploreTillSeen example shows that it explores capabilities on a high level, and we can not squeeze more out of it, but its trajectories are far from optimal. If, for example, the real robot every time navigated through every room of the apartment to search the sink, we do not want that behavior. The HLPO trajectories are also not optimal. We could improve them by giving the entire map, but it is hard to collect, and it will need to be recollected after each scene changes. Considering all of it, we think landmarks are the right tradeoff.

There are several ways to train agents to navigate, from training in real-world scenarios to fully simulated environments. The former is too inefficient as it needs a lot of resources and could bring a lot of harm while the policy is not optimal. The latter is comfortable working with, and outstanding results could be squeezed out, but our overall task is to navigate in real-world scenes. In this case, the transfer process from a simulated environment to an actual robot could be even more challenging than the training itself. In the simulator, the agent does not suffer from many real-world sensors' imperfections.

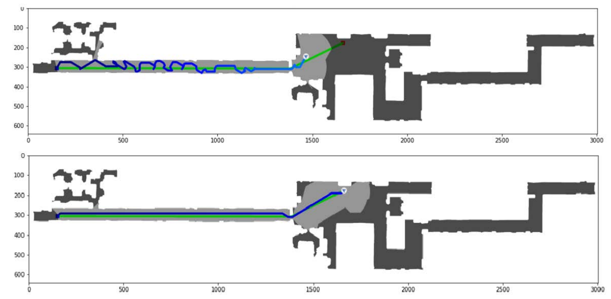


FIGURE 9. Comparison of the HLPO agent before and after adaptation.

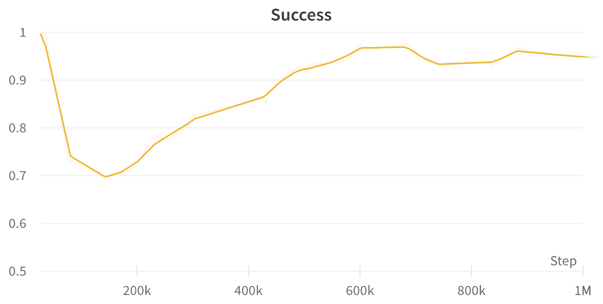


FIGURE 10. Success rate during scene adaptation.

As an intermediate approach, we use a photo-realistic environment where the scene is the one that was reconstructed by lidars and cameras.

We train our agent on a maximum number of scenes as we get from the Matterport dataset. But scene differences are too big across all datasets, and 60 of them are not enough to take into account all the properties of any scene. That is what happened in our case in the laboratory area. Long identical corridors turned out to be underrepresented in the dataset, and the agent showed far from optimal behavior inside them (Fig. 9). Since we can't capture many of these kinds of scenes, we chose to overfit our agent on our reconstructed scene before the real tests. For indoor navigation, we found it suitable in cases like ours because such a calibration needs one time for each type of scene, and the user of our system could download the suitable weights for its kind of scene.

To reconstruct the scene, we first looked at how it was done at the mp3d dataset. Dataset creators used the Matterport Pro2 camera (134 megapixels with no lidar) and the Matterport proprietary soft, where one can upload photos, and they are automatically processed to the final.obj file. That could be formatted to.glb and be used by the Habitat simulator with no effort. This works fine unless the scene contains small details. The mp3d dataset itself had a lot of holes and texture inconsistencies. We wanted higher quality, especially more precise depth reconstruction, so the agent can predictively navigate in a narrow room space. Our solution was to use a professional laser scanner, Leica RTC360 3D. An additional upside is that we can manually edit our shots, delete background people, and manually check the quality of assembling the entire scene. To texture the final point cloud, we use a RealityCapture program (Fig. 11).

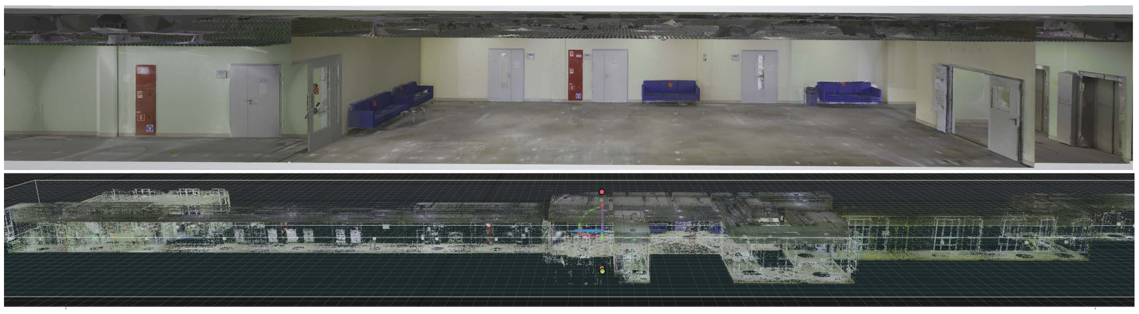


FIGURE 11. The bottom image is how the point cloud looks before texturing. The upper image is after texturing at RealityCapture.

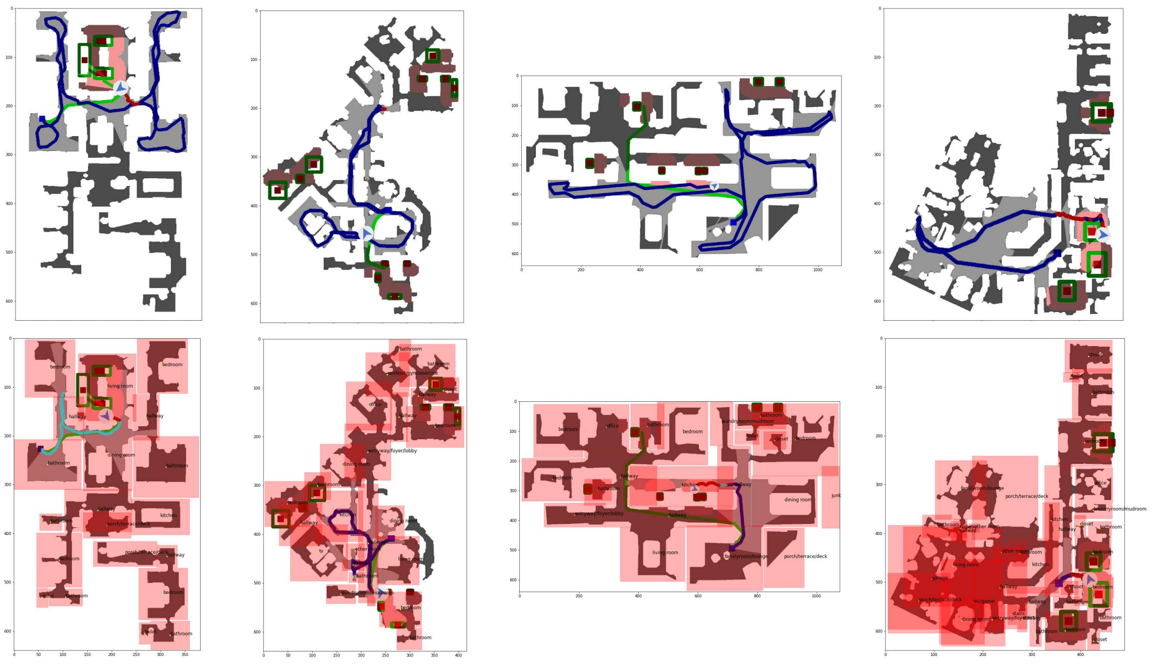


FIGURE 12. Comparison of ExploreTillSeen (up row) vs HLPO agent (down row).

TABLE 3. Comparison of the HLPO before and after the adaptation stage in our reconstructed scene.

Method	Success	Distance to goal
HLPO (Before adaptation)	0.6	11.8
HLPO (After adaptation)	**0.9**	**0.15**

Also, our laboratory reconstructed scene allowed us to calibrate the depth net (instead of simulator depth during the training phase, Fig. 8) and set proper clamps. Thus, the depth net is very close to the simulator depth. Then we collected episodes and retrained the agent on it during one million steps (Fig. 10)). Through this, we took into account the peculiarities of the scene.

After these adaptations, the success rate increased from 0.6 to 0.9 (Table 3). And at the real test on our robot Husky, we get 0.79 SPL, which indicates that our HLPO method performs equally well at the simulator and real-world tests.

VI. DISCUSSION AND CONCLUSION

We propose a novel approach to the ObjectGoal navigational task. With the standard formulation of the task, existing methods are limited to the point where, at large scenes, exploration with no information about the scene takes unreasonably much time. To solve this, we propose landmarks as a list of rooms' coordinates and their type.

With our updated task formulation, we have built a novel hierarchical policy that uses skills that could be stacked and reused in various navigational tasks with no changes. The success rate for our method doubles from 20% for the state-of-the-art method to 46% with the learned semantic, and from 51% to 86% for the ground truth semantic from the simulator. To accomplish the ObjectGoal task, we trained the agent three skills: PointNav, Exploration, and GoalReacher.

To make the transfer to reality process possible and predictable, we described how to reconstruct a scene into a simulator with a decent photo-realistic reconstruction quality using a professional Leica RTC360 scanner.

Our future work involves planning to build a global policy that would automatically select skills for a more complex task and move from discrete actions to continuous ones to control wheel-based robots more effectively.

REFERENCES

[1] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 229–241, Jun. 2001.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[3] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta: Any-angle path planning on grids," *J. Artif. Intell. Res.*, vol. 39, pp. 533–579, Oct. 2010.

[4] C. Wang, Y. Dai, N. El-Sheimy, C. Wen, G. Retscher, Z. Kang, and A. Lingua, "Progress on ISPRS benchmark on multisensory indoor mapping and positioning," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 42, pp. 1709–1713, Jun. 2019.

[5] F. Potortì *et al.*, "Off-line evaluation of indoor positioning systems in different scenarios: The experiences from IPIN 2020 competition," *IEEE Sensors J.*, vol. 22, no. 6, pp. 5011–5054, 2022.

[6] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "AI2-thor: An interactive 3D environment for visual AI," Allen Inst. AI, Univ. Washington, Stanford Univ., Carnegie Mellon Univ., Tech. Rep., 2019. [Online]. Available: https://arxiv.org/abs/1712.05474

[7] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun, "Minos: Multimodal indoor simulator for navigation in complex environments," Intel Labs, Princeton Univ., Tech. Rep., 2017. [Online]. Available: https://arxiv.org/abs/1712.03931

[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] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: Real-world perception for embodied agents," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9068–9079.

[10] 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. 1–25.

[11] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-semantic data for indoor scene understanding," Stanford Univ., Univ. California, Berkeley, Tech. Rep., 2017. [Online]. Available: https://arxiv.org/abs/1702.01105

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[13] D. Mishkin, A. Dosovitskiy, and V. Koltun, "Benchmarking classic and learned navigation in complex 3D environments," Intel Labs, Czech Tech. Univ., Tech. Rep., 2019. [Online]. Available: https://arxiv.org/abs/1901.10915

[14] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents," Austral. Nat. Univ., Princeton Univ., Carnegie Mellon Univ., Intel Labs, UC Berkeley, George Mason Univ., Facebook, Allen Inst. AI, Stanford, Tech. Rep., 2018. [Online]. Available: https://arxiv.org/abs/1807.06757

[15] 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," Allen Inst. Artif. Intell., Facebook AI Res., Georgia Inst. Technol., Robot. Google, Simon Fraser Univ., Tech. Rep., 2020. [Online]. Available: https://arxiv.org/abs/2006.13171

[16] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable ObjectGoal navigation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 16117–16126.

[17] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom. Towards New Comput. Princ. Robot. Autom. (CIRA)*, Jul. 1997, pp. 146–151.

[18] E. Wijmans *et al.*, "DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames," in *Proc. Int. Conf. Learn. Represent.*, 2019.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," OpenAI, Tech. Rep., 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[20] E. Wijmans, I. Essa, and D. Batra, "How to train pointgoal navigation agents on a (sample and compute) budget," in *Proc. 21st Int. Conf. Auton. Agents Multiagent Syst.*, 2022.

[21] D. S. Chaplot *et al.*, "Object goal navigation using goal-oriented semantic exploration,"Â in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 4247–4258.

[22] J. Kim, Y. Seo, and J. Shin, "Landmark-guided subgoal generation in hierarchical reinforcement learning,"Â in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 28336–28349.

[23] C. Hoang *et al.*, "Successor feature landmarks for long-horizon goal-conditioned reinforcement learning,"Â in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 26963–26975.

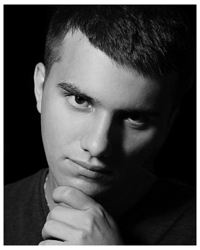
[24] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, "Objectnav revisited: On evaluation of embodied agents navigating to objects," Allen Inst. Artif. Intell., Facebook AI Res., Georgia Inst. Technol., Robot. Google, Simon Fraser Univ., Tech. Rep., 2020. [Online]. Available: https://arxiv.org/abs/2006.13171

[25] X. Wang *et al.*, "Solov2: Dynamic and fast instance segmantation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol, 33, 2020, pp. 17721–17732.

[26] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014.

[27] J. Jiang, L. Zheng, F. Luo, and Z. Zhang, "RedNet: Residual encoder–decoder network for indoor RGB-D semantic segmentation," School Automat. Sci. Eng., South China Univ. Technol., Tech. Rep., 2018. [Online]. Available: https://arxiv.org/abs/1806.01054

[28] W. Yin, J. Zhang, O. Wang, S. Niklaus, L. Mai, S. Chen, and C. Shen, "Learning to recover 3D scene shape from a single image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 204–213.



ALEKSEI STAROVEROV 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. The research thesis focuses on the methods and algorithms for the automatic determination of subgoals in a reinforcement learning problem for robotic systems. Since October 2019, he has been 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.

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" of the Russian Academy of Sciences. Since 2018, he has been the Head of the Cognitive Dynamic System Laboratory, Moscow Institute of Physics and Technology. Since 2021, he has been joined the Research Group on Neurosymbolic Integration, Artificial Intelligence Research Institute. He authored three books and more than 90 research papers. His research interests include behavior planning, reinforcement learning, semiotics, and robotics.