robotics

MDPI

Article

Skill Fusion in Hybrid Robotic Framework for Visual Object Goal Navigation

Aleksei Staroverov [1,2,3], Kirill Muravyev [2], Konstantin Yakovlev [2] and Aleksandr I. Panov [1,2,*]

[1] AIRI, 105064 Moscow, Russia; staroverov.av@phystech.edu
[2] Federal Research Center for Computer Science and Control of Russian Academy of Sciences, 119333 Moscow, Russia
[3] Moscow Institute of Physics and Technology, 141707 Dolgoprudny, Russia
* Correspondence: panov@airi.net

Abstract: In recent years, Embodied AI has become one of the main topics in robotics. For the agent to operate in human-centric environments, it needs the ability to explore previously unseen areas and to navigate to objects that humans want the agent to interact with. This task, which can be formulated as ObjectGoal Navigation (ObjectNav), is the main focus of this work. To solve this challenging problem, we suggest a hybrid framework consisting of both not-learnable and learnable modules and a switcher between them—SkillFusion. The former are more accurate, while the latter are more robust to sensors' noise. To mitigate the sim-to-real gap, which often arises with learnable methods, we suggest training them in such a way that they are less environment-dependent. As a result, our method showed top results in both the Habitat simulator and during the evaluations on a real robot.

Keywords: navigation; robotics; reinforcement learning; frontier-based exploration

Citation: Staroverov, A.; Muravyev, K.; Yakovlev, K.; Panov, A.I. Skill Fusion in Hybrid Robotic Framework for Visual Object Goal Navigation. *Robotics* **2023**, *12*, 104. https://doi.org/10.3390/robotics12040104

Academic Editor: Hong Zhang

Received: 6 June 2023
Revised: 11 July 2023
Accepted: 14 July 2023
Published: 16 July 2023

1. Introduction

In this paper, we address the problem of object navigation in indoor environments, a task in Embodied AI that involves guiding an agent to an object of a target class. This ObjectNav task can be decomposed into several classical autonomous navigation skills, such as point navigation [1], exploration [2], flee [3], and semantic goal-reaching [4] that have a long history of research in robotics and computer vision [5].

Classical approaches to solving navigation problems typically involve a modular pipeline consisting of perception, mapping, localization, planning, and motion control modules. The effectiveness of these approaches depends on numerous factors, such as the quality of sensor data, environment complexity, and the amount of engineering patches (also known as hacks). Various navigation problems can be solved with the classical approaches in both simulation and on different robotic platforms [6,7].

In recent years, an orthogonal approach, based on end-to-end machine learning, became widespread. State-of-the-Art learnable methods are not only capable of navigating freely through complex scenes without explicit map reconstruction [1,8,9], but also have abilities for manipulation [10], long-horizon planning [11], building world models [12], and communicating with humans through natural languages [13]. The success of these methods comes mainly from the high amount of data used for training. In the context of robotic applications, various simulators are typically used to generate the data. This leads to a degrading performance in the real-world conditions that may differ from the ones on which the system was trained. Another disadvantage of such methods is that, typically, learnable methods rely only on the implicit memory representation of the environment. In practice, these representations are not well-suited for long-horizon tasks. In contrast, classical, non-learnable methods exploit the explicit representation of the environment through simultaneous localization and mapping (SLAM), and these representations allow the agent to conduct effective long-horizon planning in complex scenes.

In this work, we aim to bridge the gap between classical and learnable approaches by proposing a hybrid control architecture called SkillFusion, tailored to the ObjectNav problem (see Figure 1). We are not the first to rely on the hybridization of learnable and non-learnable methods. However, unlike common fusion methods that rely on modular pipelines [14], our approach takes into account the specific strengths of both paradigms and dynamically selects which navigational skill to execute based on the self-assessment of its usefulness. We implement these skills using both classical and learning-based methods, allowing the agent to adaptively choose the most appropriate paradigm on the fly.

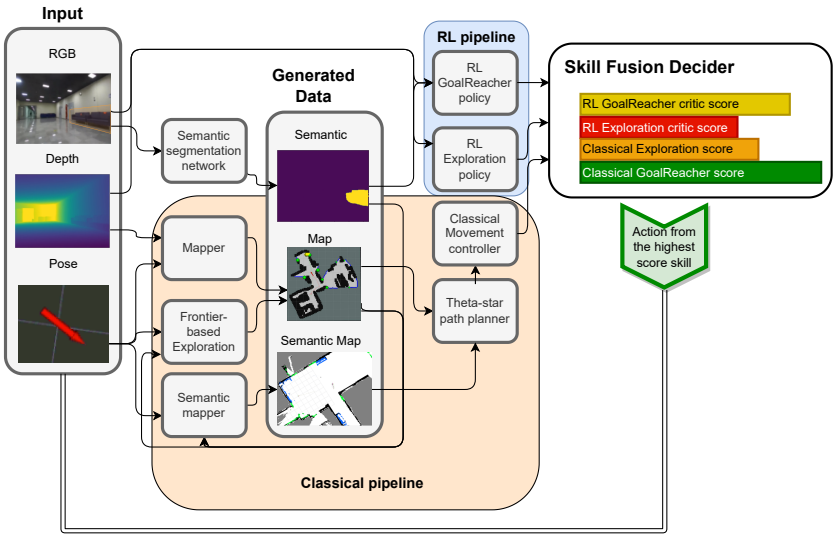

Figure 1. A scheme of our object navigation pipeline used on a real robot. The pipeline consists of classical and RL-based parts, and each part has exploration and GoalReacher skills. To choose the appropriate skill at each moment, we implement the skill fusion decider, which selects an action from the skill with the highest score.

Overall, our contributions are as follows:

- We decomposed the ObjectNav problem into two distinct skills: Exploration, where the agent must search the area for the goal object, and GoalReacher, where the agent must navigate to the detected goal object and stop near it. We then built a modular architecture that incorporates both classical and learning-based implementations of each skill. During episodes, our agent manages all skills, depending on their internal reward estimation and external conditions.
- To increase the robustness of our learning-based policy to external conditions, we used a pre-trained visual observation encoder and selected navigation tasks that require different behavior from the agent but have the same input. We trained these tasks together using an early fusion architecture [15] allowing our policy to use a single neural network with a shared recurrent embedding for all tasks and a separate head for each task.
- To address the problem of semantic noise (i.e., false detection of the goal objects, outliers, etc.), we incorporated a range of techniques for both non-learnable and learnable modules. For non-learnable modules, we utilized map-based erosion and fading. For learnable modules, we introduced a special "not sure" action that prevents a learnable policy from heading toward a falsely detected object by switching to exploration behavior.

As a result, we were able to transfer our SkillFusion method from the simulator to the real world with no additional training by only implementing more appropriate motion execution at the actuator level.

2. Related Work

Learning-based navigation. Learning-based approaches typically involve the use of reinforcement learning (RL) algorithms to learn a value function or to directly map the state into action using policy gradient algorithms [16]. These algorithms extract a feature representation relevant to the specific task using approximation functions.

One approach to obtaining a more robust and transferable representation of the environment is through multitasking. This approach is well-suited for our multiple navigation skills setup, as it allows the agent to track and memorize different information from the same observational state [17]. Recent successes in multitasking have demonstrated the ability of a single network to perform a wide range of tasks [18]. Our experiments show that training multiple skills together in an early fusion manner [15] yields better results compared to end-to-end baselines [1].

Another approach to enhancing the representation is by disentangling the visual encoding of the state from the task policy. This can be achieved by training encoders on larger and more diverse datasets, resulting in less domain-specific encoders. These encoder parameters can then be frozen, allowing the RL algorithm to train fewer parameters and converge faster [19,20]. The use of pre-trained models, such as CLIP [21], as image encoders in navigation tasks has also been shown to be effective [22,23].

Classical navigation. A classical approach to navigation in previously unseen environments assumes the decomposition of the navigation task into the following subtasks: localization, mapping, path planning, and path following. Localization and mapping tasks are often solved in pair, using the simultaneous localization and mapping (SLAM) approach. Most of the SLAM methods [24–26] extract features from the camera's images or lidar clouds and track robot motion using feature matching. One of the most popular methods is RTAB-MAP [27]. It is able to work with different sensors (RGB-D camera, stereo camera, lidar) and builds both a 2D occupancy map and a 3D point cloud map.

For path planning, the most popular approach is using graph path search algorithms like A* [28] or Theta* [29] on an occupancy grid map built by SLAM. For path following, many control methods exist that are typically tailored to a specific robotic platform [30–32].

Another important task related to navigation is the exploration of an unknown environment. The mainstream in classical exploration methods is using frontiers [33,34]. Frontier-based methods search for frontiers between the free and the unexplored cells on an occupancy grid map and choose one of these frontiers as a goal. A convenient and effective implementation of a frontier-based exploration algorithm is provided in [35].

Fusion of classical and learning-based navigation. Numerous works showed that learning-based agents have inferior collision avoidance and memory management but are superior in handling ambiguity and noise [36,37]. One line of fusing learnable and non-learnable approaches is developing learning-based navigation modules, which can be integrated into the final pipeline. SLAM and planner can be formulated in a differential form and trained as one system [38]. For exploration and ObjectGoal tasks, some recent works [2,14,39] show the supremacy of modular approaches over end-to-end learning-based ones and pure classical ones. The authors implemented an RL-based global policy module that predicts frontiers and a classical planner local policy for navigation to those frontiers. In our approach, we do not specify each skill as being classical or learning-based, but instead implement each skill with both approaches without any additional information as in [40]. This allows the agent to determine which type of approach should be executed at each moment based on their cost function.

Another hybrid control strategy is Bayesian Controller Fusion (BCF) [41], which combines an uncertainty-aware stochastic RL policy with a hand-crafted controller. The Bayesian formulation allows the method exhibiting the least uncertainty to dominate control. In states of high policy uncertainty, BCF biases the composite action distribution towards the risk-averse prior, reducing the chances of catastrophic failure. However, we found that this approach is more suited for manipulation tasks with large action spaces where collision

risk is more important than task metrics. Our approach, on the other hand, relies more on reward estimation rather than uncertainty.

3. ObjectGoal Navigation Task

The indoor object navigation task (ObjectGoal) is generally defined as the task of navigating to an instance of the object category $C \in \{c_1, c_2, ..., c_n\}$ (e.g., a *couch*) in a previously unseen environment [4]. At each step, the agent obtains observation $S = (S_{RGBD}, S_{GPS+Compass}, C)$. The action space is discrete and consists of four types of actions: `callstop` (to end the episode), `forward` by 0.25 m , `turnleft`, and `turnright` by angle $\alpha = 30°$.

After the end of the episode, the agent is evaluated via three primary metrics: (1) Success, where an episode is considered successful if the agent executes the `callstop` command within 1.0 m of any goal-type object; (2) Success weighted by (the inverse normalized) Path Length (SPL), where success is weighted by the efficiency of the agent's path to the closest object from the starting point; (3) SoftSPL, where binary success is replaced by progress toward the goal.

ObjectGoal is a complex task that requires different behaviors from the agent. These behaviors can be treated as agent skills. We have distinguished two main skills that directly solve the ObjectGoal task: Explore skill and GoalReacher skill, and two additional ones: a PointNav skill and a Flee skill. For each of the main skills, we implemented both classical and learning-based ways (see Figure 2 for an example). The additional skills are used for the generalization effect of the learning-based policy.

- Explore skill has the objective to observe as many areas as possible in a limited amount of time.
- GoalReacher skill has an objective to navigate the agent to the given object that was observed somehow and execute the `callstop` action within 1.0 m from it.
- PointNav skill has an objective to navigate the agent to a given point and execute `callstop` action within a 1.0 m distance from it.
- Flee skill has an objective to execute the `callstop` action at the furthest point from the starting one.

Embodiment. The simulator embodiment has the LoCoBot's (a low-cost mobile robot) parameters. The base radius is 0.18 m. The ground-truth localization was provided via *GPS+Compass* sensor ($S_{GPS+Compass}$), and the action execution was also noise-free, but the coordinates of the goal objects or ground-truth semantic segmentation were unavailable to the agent.

The real-world embodiment is the four-wheel differential drive Clearpath Husky platform that has dimensions of 990 mm $\times$ 670 mm $\times$ 390 mm. As it significantly differs from the simulation embodiment, we showcase that our pipeline is able to easily adapt to a specific robotic platform without learning-based policy fine-tuning and module changes.

Perception. In a simulator, the agent has a noise-free monocular RGB-D camera mounted at a height of 0.88 m. The camera's resolution is 480 $\times$ 640 pixels with a 90° horizontal field of view. Its depth range is 5 m.

To match those inputs in the real world, Husky has a Velodyne VLP-16 lidar and RGB-D camera sensors. The range of all the perception sensors is restricted by 5 m. The lidar is used for accurate localization and mapping in order to compensate for the absence of a *GPS+Compass* sensor on the robot.

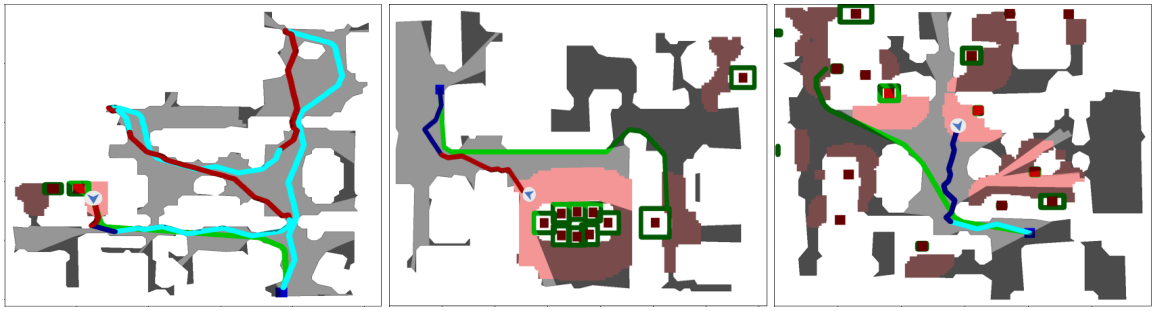

Figure 2. Examples of skill fusion work in a simulator. The red line is a trajectory that was attained by the classical pipeline, the light blue line was by RL exploration skill, and the dark blue was by RL GoalReacher skill.

4. Classical Pipeline

4.1. Exploration Skill

Our implementation of classical exploration skills consists of the following parts: goal setting, localization, mapping, path planning, and motion control.

Goal-setting module. The goal-setting module takes a current estimated robot position and a SLAM-built map and chooses the goal where the robot should go. If the target object is mapped by SLAM, exploration chooses the point nearest to it.

As a goal-setting module, we used our implementation of the frontier-based exploration approach, which was proposed in [35]. The algorithm looks for frontiers between the free and the unknown space on a 2D SLAM-built map. To find these frontiers, the breadth-first search (BFS) is used.

All the frontiers are assigned cost functions, and the centroid of the frontier with the lowest cost function is marked as the goal for the robot. The frontier's cost function considers the distance between the agent and the frontier's centroid, the size of the frontier, and the angle the agent has to turn before it starts moving to the frontier. We describe it formally below.

Let $q \in \mathbb{R}^2$ be the robot orientation vector, and $\pi = (p_0, p_1, \ldots, p_k)$ be a path from the robot to the centroid of i-th frontier of size n_i. In path π, p_0 is the robot position, and p_k is the centroid of the $i - th$ frontier. Our cost function is

$$cost_i = \alpha \sum_{j=0}^{k} ||p_{j+1} - p_j||_2 - \beta n_i + \gamma |\angle(q, p_1 - p_0)| \quad (1)$$

where α is the coefficient for path length, β is the coefficient for frontier size, and γ is the coefficient for the turn angle between the robot orientation and direction to the frontier.

Localizer module. In a simulation, we use ground-truth position data, so the localizer is not needed. On the real robot, to maximize the localization accuracy and avoid error accumulation, we use the LOL-odom lidar odometry algorithm [42] with an additional pose correction via a pre-built 3D map (this map is used for pose correction only).

Mapper module. The mapper module builds an occupancy grid map for navigation at the exploration stage and a semantic map for navigation to a goal object. In a simulation where the ground-truth data is available, we use the back projection of the depth map to build the occupancy grid. To build the semantic map, we use the back projection of the predicted semantic mask. All objects with a height of less than 0.2 m are mapped as ground cells, and all objects higher than 0.2 m are marked as obstacles. On the real robot, we use the RTAB-MAP SLAM algorithm [27] with a lidar as a perception source. Semantic information is added to the RTAB-MAP map from the semantic masks predicted from the onboard RGBD camera.

In both simulation and the real robot, we build a map with a resolution of 5 cm. To cover small "holes" in the map and reduce the number of redundant frontiers appearing due to the map gaps, we increase the cell size twice using the max pooling method with

cell value order "unknown < free < obstacle"; the map is decomposed into 2×2 squares, and each square forms a single cell in the resized map. This cell is marked as an obstacle if the square contains an obstacle cell. If the 2×2 square contains a free space cell and no obstacle cell, the resulting cell in the resized map is marked as a free space cell. And if the 2×2 square contains only unknown cells, the resulting cell of the resized map is marked as unknown. An example of such max pooling is shown in Figure 3.

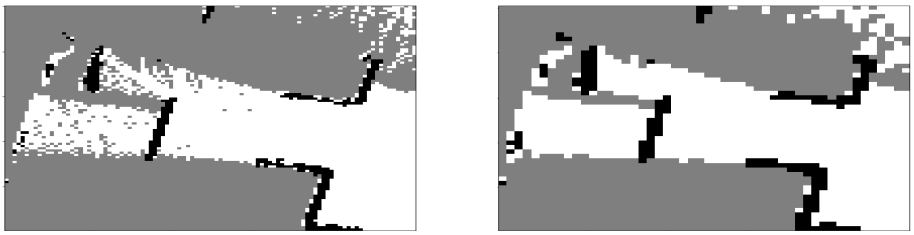

Figure 3. An example of max pooling of the built map. Gray denotes an unknown area, black denotes obstacle cells, and white denotes free space cells.

To prevent mapping outliers in predicted semantic masks, we implement semantic map erosion and semantic information accumulation. We erode goal objects on the semantic map by two cells. Also, we use continuous values of semantic map cells instead of binary ones. If a semantic object is projected onto a map cell, we increase the value in this cell by 1; otherwise, we multiply the value by a coefficient $\alpha < 1$. A cell is marked as a cell of the target object if its value increases the threshold T. In our experiments, we used $\alpha = 0.9$ and $T = 2$.

Planner module. For path planning, we used Theta* algorithm [29]. This is an algorithm for pathfinding on grids that support any-angle paths. By default, Theta* plans a path for a zero-size agent, so we modified this algorithm to handle the robot size. In our modification, a robot is modeled as a disk of radius r. In practice, we set the size of the disk to be larger than the actual robot's footprint to accommodate the (extra) safety margin. If a path was not found, we decreased the safety radius to $0.8r$ and repeat the path planning. We set $r = 0.15$ for the simulation and $r = 0.6$ for the robot.

Controller module. For path following in simulation, we use a simple and straightforward algorithm. We compare the robot's orientation and the direction from the robot's pose to the next point of the path. If the angle between the robot orientation and the direction to the path point is under some threshold (e.g., less than 5 degrees in absolute value), we move the robot forward. If it is above the threshold and is negative, we turn the robot left. If it is above the threshold and is positive, we turn the robot right.

On the real robot, we use our implementation of the partial trajectory tracking method, which receives the path and the robot's position and outputs velocity commands for the robot. This partial trajectory tracking is like our simulation control approach, but it is adapted for continuous motion and has some heuristics to compensate for odometry errors.

4.2. GoalReacher Skill

For goal-reaching we use the same localization, planning, mapping, and controller modules as described above. For goal-setting, we again utilize a frontier-based approach, but instead of frontiers between free and unexplored map space, frontiers of the target objects are used.

5. Learning-Based Pipeline

We will define a learning-based policy through a Markov decision process (MDP) $< S, A, T, R, \gamma >$, where S is a set of states, A is a set of available actions, $T(s_{t+1}|s_t, a_t)$ is a transition function, R is a reward function, and γ is a discount factor. In our setting, the states s_t are not fully observable. Only incomplete information about the state is available to the agent at each time step—the observation o_t. We will assume that the agent obtains an approximator f of the state s_t from the observation history $s_t \approx f(o_t, o_{t-1}, \dots)$

(in practice, this is implemented as a part of the agent's neural network, which is in charge of decision making).

Same as the classical one, our learning-based pipeline consists of two components: an Exploration policy and a GoalReacher policy. To train those skills, we have leveraged Decentralized Distributed Proximal Policy Optimization (DD-PPO) as it shows promising results in similar visual navigation tasks [1]. The architecture to train the proposed RL pipeline is shown in Figure 4. Crucial to that result was a vast amount of training steps. That requires a fast-performing simulator that should also be photo-realistic to be able to transfer the resulting policy to the real world. To this end, we used the fastest photo-realistic simulator BPS [3] with the largest 1000-scene dataset, HM3D [43]. To train the RL Exploration skill, we take the train part of HM3D (800 scenes) and 145 scenes for the RL GoalReacher skill, as HM3D has only that number with available ground truth semantics.

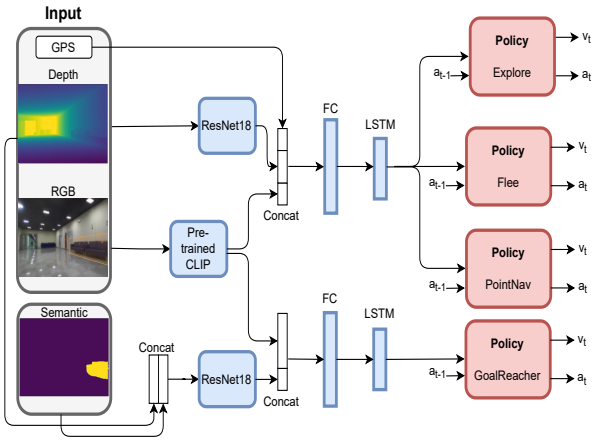

Figure 4. The proposed neural net architecture for fusing navigation tasks at the RL training phase.

5.1. Exploration Skill

Exploration is a challenging task, requiring precision, the ability to memorize past experiences, and future planning. In real-world scenes and photorealistic environments, this task is complicated by the significant variation in indoor environments' size, layout, furnishing, color, and other parameters. To generalize these peculiarities, an agent needs a highly capable encoder to extract all necessary information from a noisy RGB sensor. We addressed this problem by dividing it into two sections: obtaining a visually robust image encoder and developing a dynamic robust RNN encoder.

As an image encoder, we employed a pre-trained CLIP model [21] and froze it during the training phase. This approach has been used in previous works [22,23], and our experiments also demonstrate the ability to navigate solely based on an RGB sensor with a frozen CLIP encoder.

To address the dynamic robust RNN encoder problem, we fused all navigational skills and trained them with one neural network (Figure 4) with shared visual encoders and RNN layers and multiple heads in an early fusion manner [15]. The exploration skill needs to be aware of the area's spatial structure to avoid exploring already explored areas, while the flee skill focuses on distance measurements and orientation, and the pointnav skill mainly focuses on finding the shortest possible path to a point. The reward function of each task motivates these behaviors.

- The Exploration skill reward is set as +1 when the agent visited a previously unexplored 1 m^2 cell; otherwise, it was 0.
- The Flee skill reward is proportional to the distance from the starting point.
- The PointNav skill reward is proportional to the shortened distance to the goal, plus a reward if the agent executes a stop action within 1m of the goal point.

5.2. GoalReacher Skill

A crucial aspect of mastering the GoalReacher skill is the ability to differentiate between goal objects and non-goal objects. To convey information about the object, we opted to pass a binary segmentation mask of the object rather than its ID. This approach enabled us to train the semantic segmentation model independently. The downside of it is that we need to train the policy with the ground-truth semantic sensor to be able to give proper rewards. But in validation episodes, the semantic segmentation model will give a lot of noises and false positive objects that at some frame will appear and at some not.

During validation episodes, the semantic segmentation model may produce noise and false positive objects that appear intermittently. The GoalReacher skill is trained with a large positive reward for successful episode completion and a small negative reward for each step taken. As a result, noise objects may cause the agent to terminate the episode in a location where a noise object appears in the hope that it is the real one, resulting in a large negative reward if it is not. Alternatively, the agent may attempt to locate another more distant object and accumulate a large sum of small negative rewards by executing too many actions.

To enable the agent to distinguish between noisy and real objects and avoid being trapped in unfavorable decision-making situations, we propose a novel “not sure" action. When the agent is uncertain about the validity of the segmentation mask, it can execute this action to switch back to the Exploration skill rather than pursuing a perceived noise object.

During the learning-based GoalReacher skill training phase, we sample episodes with noise object semantic segmentation with a probability of 0.2; the remaining episodes contain valid ground truth segmentation. In noisy episodes, the agent can receive an intermediate negative reward if it chooses the “not sure" action when ending the episode and a large negative reward if it executes the “stop" action elsewhere. The agent’s goal is to quickly recognize if semantic segmentation is valid and terminate the episode with the “not sure” action or pursue the goal, verify its validity, and execute the “stop” action near its position.

6. Skill Fusion Decider

To bring together the advantages of the classical and learning-based approaches, we propose a SkillFusion method as being separate; those approaches have several limitations.

Classical exploration methods based on frontiers on a 2D map are reliable and robust in the face of scene and camera changes; they require only precise odometry and map sources. However, learning-based approaches may be more effective in solving specific tasks on specific types of scenes because they can be trained to solve these tasks directly, e.g., using different RL strategies. In object navigation, RL-based approaches show great effectiveness in scenes where the goal can be found in a short time, but at long distances, recurrent neural networks start forgetting the information from the initial stages and making rounds in previously visited places. On the contrary, classical pipelines store entire information from the start as a 2D map and can explore new places for the whole operation time. Also, RL approaches are good at goal-finding and aiming, but they struggle to stop in the vicinity of the object.

To effectively leverage the advantages of each approach, the SkillFusion module must be aware of the benefits that each skill can provide at any given time. We already have cost functions for this purpose: the critic value function for RL learning-based skills and the frontier cost function for classical skills (Figure 5). We normalize these functions to the same order based on collected experience, and at each step, the agent executes the skill with the maximum value. Since the goal of the ObjectNav task is to execute a stop action when the agent is near the goal, we assign to the classical GoalReacher skill the highest constant value when the goal appears on the map after all filtering, as this skill is more relevant than any exploration.

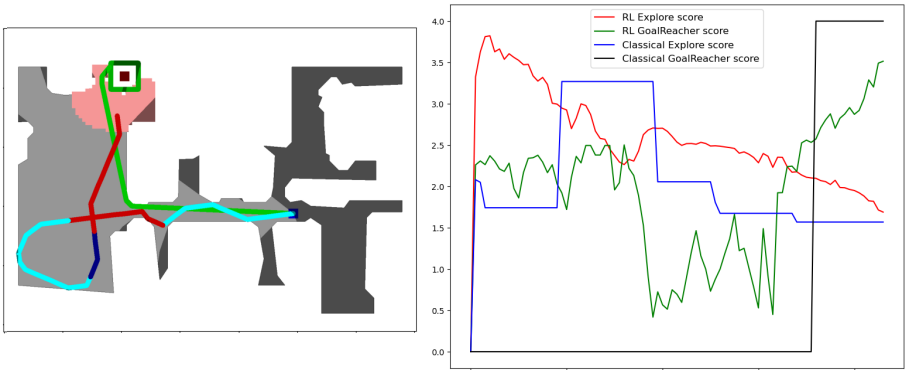

Figure 5. Example of skill management during the episode. The blue color of the trajectory is denoted RL Explore skill execution, dark blue denotes an RL GoalReacher skill, and a red color denotes classical skills.

As a result, the agent starts the exploration with the RL approach to efficiently cover large areas around the start position. After some time, RL forgets the information in its recurrent layers and starts to explore the already-explored areas. When it occurs, the RL Explore skill score gets lower than the classical score and the agent switches to the Classical skill. At this moment, the Classical exploration skill based on a current map navigates the agent to the highest value frontier, and after that, RL exploration score becomes larger than the classical, and the policy switches back to RL (Figure 2 left). We should note that we update the RL RNN layers at each step even when the agent executes classical skills.

If the goal object gets seen by semantic segmentation, the RL GoalReacher score rushes up and takes control until the object does not appear on the semantic map. Under RL GoalReacher skill, the agent has two options: to finish the episode by itself by executing the “stop" action (Figure 2 right), or to give back control to the Exploration skills by executing “not sure" if it decides that a goal object is a noise. When the goal object is on the segmentation map, we set the classical GoalReacher score to a constant high value, and the agent switches to that (Figure 2 middle).

7. Experiments

7.1. Simulator Experiments

Table 1 shows the results of our method compared to current state-of-the-art solutions in the Habitat simulator [44]. For validation episodes, we used 20 scenes from the Val part of the HM3D dataset [43] that the agent had never seen during its training phase. The resulting number of episodes was 120, with an equal distribution of goal categories (chair, bed, plant, toilet, monitor, sofa) and an average minimum distance to the closest object of 8 m. As a baseline for comparison, we chose three methods. DDPPO [1] is an end-to-end RL method provided by the Habitat simulator authors. SemExp [14] is a modular approach that incorporates the fusion of a learning-based global planner and a classical local planner. Auxiliary RL [8] is an RL approach augmented with auxiliary loss functions. For all baselines, we tested available open-source implementations and weights, but to make these methods applicable to the HM3D dataset and our goal type of objects, we used a ground truth semantic segmentation module instead of the baseline ones.

Table 1. Performance of SkillFusion as compared to the baselines on the HM3D dataset (right).

Method	Success	SPL	SoftSPL
DDPPO [1]	0.18	0.10	0.35
SemExp [14]	0.24	0.14	0.26
Auxiliary RL [8]	0.51	0.29	0.34
SkillFusion	**0.64**	**0.36**	**0.38**

The RGB encoder constitutes a large portion of the neural network's parameters, which slows down the training phase. Additionally, we have a limited number of scenes in the dataset, but we want the agent to navigate beyond them. Therefore, the robustness of the RGB encoder in novel scenes is a challenging problem. A solution to this problem could be to use an already-pretrained RGB encoder that will be frozen during the training phase. For this encoder, we chose a CLIP model [21]. To demonstrate that reducing learnable parameters still allows the agent to perform navigation tasks, we compared the GoalReacher task performance with a frozen CLIP versus a learnable ResNet model Figure 6. As our experiment shows, the performance of the agent even improved. To analyze the performance aid from depth sensors and show that the neural network relies on CLIP embeddings not only for object recognition but also for solving navigation tasks, we trained the GoalReacher skill exclusively on RGB input. Our architecture with this ablation still performed with a high score.

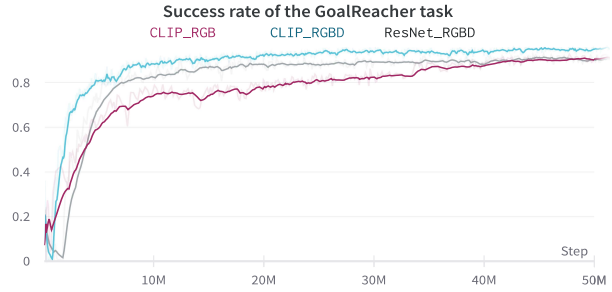

Figure 6. Comparison of GoalReacher skill training using a frozen CLIP encoder versus an unfrozen ResNet encoder.

To prove the need for the fusion of classical and RL approaches, we tested different combinations of classical and RL skills. The comparison results show that the pure RL is better than the pure classical approach at the SPL metric in a simulator but has a comparable success rate (see Table 2). This happened because the classical pipeline only relies on a map, is "blind" beyond Depth range limitations, and spends a lot of time exploring corners or dead-ends, while RL can see those by extracting information from RGB. The comparable success rate is due to the fact that, for a classical pipeline, it is easier to decide when the agent needs to stop the episode once the goal object is seen, as it can calculate a distance to it with a metric map.

Table 2. Ablation of the different available sets of skills during the episode.

Skill		Metrics		
Explore	**GoalReacher**	**Success**	**SPL**	**SoftSPL**
Classical	Classical	0.410	0.182	0.263
RL	RL	0.403	0.224	0.321
RL+Classical	Classical	0.511	0.299	0.309
RL+Classical	RL+Classical	0.547	0.316	0.365

The fusion of classical and learning-based methods for an exploration skill improves the exploration ability of the agent; thus, it finds more goal objects and increases both SPL and success metrics. And, the final addition of the GoalReacher RL increases the number of successful episodes in the final steps of trajectories, which also reflects positively on metrics.

As shown in Table 3, the SkillFusion model's performance is sensitive to the quality of a semantic sensor. With a ground truth (GT) semantic, the results almost doubly outperform the SegFormer ones, but with the implementation of semantic map filtering for the classical pipeline and the addition of the "not sure" action to the RL pipeline, we mostly compensate for that gap.

Table 3. Semantic segmentation module ablations.

Method	Success	SPL	SoftSPL
SkillFusion (with no semantic filtering)	0.324	0.207	0.327
SkillFusion (with semantic filtering)	**0.547**	**0.316**	**0.365**
SkillFusion (ground truth semantic)	0.647	0.363	0.384

7.2. Robot Experiments

Besides simulation experiments, we carried out a range of tests on the Clearpath Husky robot (Figure 7). The robot was asked to find different objects (like a chair or a sofa) in an unknown indoor environment (a university building). It performed the tasks without any network fine-tuning.

To detect the strengths and weaknesses of the classical and RL pipelines in the case of a real robot, we first tested them separately and then compared both pipelines with our SkillFusion pipeline. We ran all the pipelines on seven scenes with three different goal objects: a blue chair, a red chair, and a blue sofa. The blue chair was located in the corner of the hallway. In the test with the sofa, two large blue sofas were located in different corners of the hallway. The red chair was located behind a narrow passage in the hallway, and the robot had to explore the entire hallway before entering this narrow passage to reach the goal. For the blue chair and sofas, we ran the pipelines from three different starting locations, while for the red chair, we used only one starting location.

The trajectories for all the pipelines are shown in Figure 8, and the metric values are shown in Table 4. We measured five metrics: success rate, SPL, SoftSPL, the length of the path traveled by the robot, and the traveling time. With a pure RL approach, the robot failed to reach the goal object twice. In the test with a blue chair, the robot reached a sofa instead of a chair. This happened due to semantic mapping errors and a straightforward goal-reaching approach without semantic data filtering. In the test with a red chair, the robot ignored the narrow passage with the chair because RL was trained with a reward function proportional to the explored area. As a result, the reward function for entering a narrow passage was too low.

In one test out of three, the robot failed to find the blue chair using the classical pipeline (Figure 8). This was due to the restricted range of 5 m for semantic mapping and semantic prediction noises, which caused the target chair to be ignored. As a result, the robot explored the entire hallway without marking the chair as a goal.

Using our SkillFusion pipeline, the robot succeeded in all tests, achieving an average SPL and SoftSPL of 0.65. The average traveling distance was reduced to 23 m, compared to 27 m with the classical approach and 31 m with the RL approach. This demonstrates the successful transfer of the strengths of both classical and learning-based approaches to a real robot.

Figure 7. Robot platform based on the Clearpath Husky chassis with a ZED camera (**left**). We used it to evaluate our results in a real-world scene (**right**).

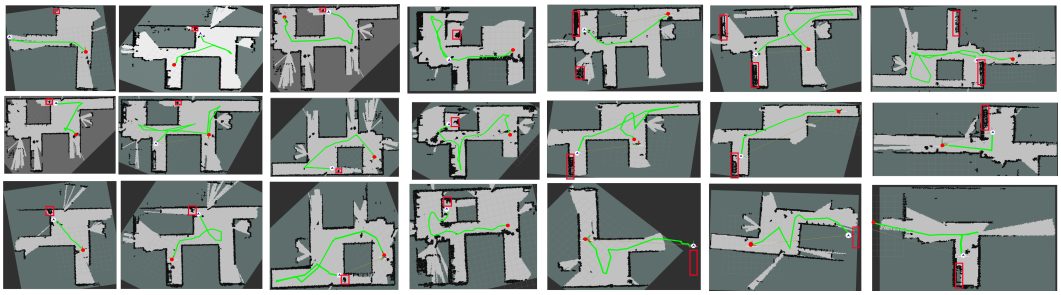

Figure 8. Robot trajectories: RL (**top**), classical pipeline (**middle**), and SkillFusion (**bottom**). The red box denotes the target object, the red circle denotes the start point, and the white circle with a blue arrow denotes the robot's termination point and direction.

Table 4. Results of separate tests of classical and learning-based approaches on the real robot, in comparison with the SkillFusion method.

Method	Scene	Success	SPL	SoftSPL	Path Length, m	Time, s
RL	Chair 1	0.67	0.53	0.42	22	170
	Chair 2	0	0	0.42	36	300
	Sofa	1	0.39	0.39	38	190
	Average	0.71	0.39	0.41	31	200
Classic	Chair 1	0.67	0.35	0.41	28	180
	Chair 2	1	0.40	0.40	33	220
	Sofa	1	0.75	0.75	24	110
	Average	0.86	0.53	0.56	27	170
SkillFusion	Chair 1	1	0.61	0.61	25	150
	Chair 2	1	0.61	0.61	22	140
	Sofa	1	0.77	0.77	22	110
	Average	**1**	**0.65**	**0.65**	**23**	**130**

8. Conclusions

In this work, we have considered a challenging problem of visual navigation. In this problem, an embodied agent (either virtual or real robot) is asked to reach an object belonging to a predefined class (e.g., a table, a chair etc.) in an unknown environment, which can be locally observed by the vision sensor(s) of the agent. We have examined both the non-learnable (classical) and learnable methods needed to solve this problem efficiently. Our work provides clear evidence that classical and learning-based approaches to robot navigation do not contradict but complement each other. We have implemented them both in the simulator and in real-world conditions at a high level and identified their strengths and weaknesses.

To increase the classical pipeline's performance, we have tuned and improved Frontier-based Exploration. To adopt classical skills in real-world conditions, where ground-truth depth and GPS sensors are unavailable, we used lidar localization and mapping modules that allowed us to achieve similar performance. To increase the performance of the RL skills, we have introduced a fusion training scheme and a novel "not sure" action, which not only increases the metrics, but makes the model more robust to external conditions. These improvements also have made RL transferable into the real world without any scene adaptation.

We have implemented each part of the classical and learning-based algorithms as an agent's skills and propose a skill decider module that, based on each skill's intrinsic reward model, switches between classical and learning-based skills during an episode. Our proposed fusion mechanism relies on fundamental differences between the classical and learning-based pipelines. In our experiments, RL is good for quickly exploring large areas

and is able to extract more information from the sensors for more intelligent exploration, but it is not so precise, and its memorization abilities are far behind the classical method. So, whenever those abilities are needed to accomplish a goal in the most effective way, the agent can execute an appropriate skill. In the future, we will investigate semantic map reconstruction with interaction segmentation and planning beyond the current map through 3D scene reconstruction and make an extended version of it.

Author Contributions: Conceptualization, A.I.P.; methodology, K.Y. and A.I.P.; software, A.S. and K.M.; investigation, A.S., K.M. and A.I.P.; writing—original draft preparation, A.S. and K.M.; writing—review and editing, K.Y. and A.I.P.; visualization, A.S.; supervision, A.I.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Science and Higher Education of the Russian Federation under Project 075-15-2020-799.

Data Availability Statement: Video and code for our approach can be found on our website: https://github.com/AIRI-Institute/skill-fusion (accessed on 13 July 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wijmans, E.; Kadian, A.; Morcos, A.; Lee, S.; Essa, I.; Parikh, D.; Savva, M.; Batra, D. DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames. *arXiv* **2019**. [CrossRef]
2. Chaplot, D.S.; Gandhi, D.; Gupta, S.; Gupta, A.; Salakhutdinov, R. Learning to Explore using Active Neural SLAM. *arXiv* **2020**. [CrossRef]
3. Shacklett, B.; Wijmans, E.; Petrenko, A.; Savva, M.; Batra, D.; Koltun, V.; Fatahalian, K. Large Batch Simulation for Deep Reinforcement Learning. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual Event, 3–7 May 2021.
4. Batra, D.; Gokaslan, A.; Kembhavi, A.; Maksymets, O.; Mottaghi, R.; Savva, M.; Toshev, A.; Wijmans, E. ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects. *arXiv* **2020**, arXiv:2006.13171.
5. Bonin-Font, F.; Ortiz, A.; Oliver, G. Visual Navigation for Mobile Robots: A Survey. *J. Intell. Robot. Syst.* **2008**, *53*, 263–296. [CrossRef]
6. Kadian, A.; Truong, J.; Gokaslan, A.; Clegg, A.; Wijmans, E.; Lee, S.; Savva, M.; Chernova, S.; Batra, D. Are we making real progress in simulated environments? Measuring the sim2real gap in embodied visual navigation. *arXiv* **2019**, arXiv:1912.06321.
7. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]
8. Ye, J.; Batra, D.; Das, A.; Wijmans, E. Auxiliary tasks and exploration enable objectgoal navigation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 16117–16126.
9. Xue, H.; Hein, B.; Bakr, M.; Schildbach, G.; Abel, B.; Rueckert, E. Using Deep Reinforcement Learning with Automatic Curriculum Learning for Mapless Navigation in Intralogistics. *Appl. Sci.* **2022**, *12*, 3153 . [CrossRef]
10. Fugal, J.; Bae, J.; Poonawala, H.A. On the Impact of Gravity Compensation on Reinforcement Learning in Goal-Reaching Tasks for Robotic Manipulators. *Robotics* **2021**, *10*, 46. [CrossRef]
11. Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* **2020**, *588*, 604–609. [CrossRef]
12. Hafner, D.; Lillicrap, T.; Norouzi, M.; Ba, J. Mastering Atari with Discrete World Models. *arXiv* **2020**. [CrossRef]
13. Padmakumar, A.; Thomason, J.; Shrivastava, A.; Lange, P.; Narayan-Chen, A.; Gella, S.; Piramuthu, R.; Tur, G.; Hakkani-Tur, D. TEACh: Task-driven Embodied Agents that Chat. *arXiv* **2021**. [CrossRef]
14. Chaplot, D.S.; Gandhi, D.; Gupta, A.; Salakhutdinov, R. Object Goal Navigation using Goal-Oriented Semantic Exploration. *arXiv* **2020**. [CrossRef]
15. Gadzicki, K.; Khamsehashari, R.; Zetzsche, C. Early vs late fusion in multimodal convolutional neural networks. In Proceedings of the 2020 IEEE 23rd International Conference on Information Fusion (FUSION), Rustenburg, South Africa, 6–9 July 2020; pp. 1–6.
16. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
17. Gordon, D.; Kadian, A.; Parikh, D.; Hoffman, J.; Batra, D. SplitNet: Sim2Sim and Task2Task Transfer for Embodied Visual Navigation. *arXiv* **2019**. [CrossRef]
18. Reed, S.; Zolna, K.; Parisotto, E.; Colmenarejo, S.G.; Novikov, A.; Barth-Maron, G.; Gimenez, M.; Sulsky, Y.; Kay, J.; Springenberg, J.T.; et al. A Generalist Agent. *arXiv* **2022**. [CrossRef]
19. Yadav, K.; Ramrakhya, R.; Majumdar, A.; Berges, V.P.; Kuhar, S.; Batra, D.; Baevski, A.; Maksymets, O. Offline Visual Representation Learning for Embodied Navigation. *arXiv* **2022**. [CrossRef]

20. Baker, B.; Akkaya, I.; Zhokhov, P.; Huizinga, J.; Tang, J.; Ecoffet, A.; Houghton, B.; Sampedro, R.; Clune, J. Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos. *arXiv* **2022**. [CrossRef]
21. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning Transferable Visual Models From Natural Language Supervision. *arXiv* **2021**. [CrossRef]
22. Khandelwal, A.; Weihs, L.; Mottaghi, R.; Kembhavi, A. Simple but Effective: CLIP Embeddings for Embodied AI. *arXiv* **2021**. [CrossRef]
23. Deitke, M.; VanderBilt, E.; Herrasti, A.; Weihs, L.; Salvador, J.; Ehsani, K.; Han, W.; Kolve, E.; Farhadi, A.; Kembhavi, A.; et al. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. *arXiv* **2022**. [CrossRef]
24. Campos, C.; Elvira, R.; Rodríguez, J.J.G.; Montiel, J.M.; Tardós, J.D. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
25. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-time loop closure in 2D LIDAR SLAM. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1271–1278.
26. Sumikura, S.; Shibuya, M.; Sakurada, K. OpenVSLAM: A versatile visual SLAM framework. In Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 21–25 October 2019; pp. 2292–2295.
27. Labbé, M.; Michaud, F. 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.* **2019**, *36*, 416–446. [CrossRef]
28. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
29. Nash, A.; Daniel, K.; Koenig, S.; Felner, A. Theta*: Any-angle path planning on grids. In Proceedings of the AAAI, Vancouver, BC, Canada, 22–26 July 2007; Volume 7, pp. 1177–1183.
30. Faulwasser, T.; Weber, T.; Zometa, P.; Findeisen, R. Implementation of nonlinear model predictive path-following control for an industrial robot. *IEEE Trans. Control Syst. Technol.* **2016**, *25*, 1505–1511. [CrossRef]
31. Soetanto, D.; Lapierre, L.; Pascoal, A. Adaptive, non-singular path-following control of dynamic wheeled robots. In Proceedings of the 42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475), Maui, HI, USA, 9–12 December 2003; Volume 2, pp. 1765–1770.
32. Guo, H.; Cao, D.; Chen, H.; Sun, Z.; Hu, Y. Model predictive path following control for autonomous cars considering a measurable disturbance: Implementation, testing, and verification. *Mech. Syst. Signal Process.* **2019**, *118*, 41–60. [CrossRef]
33. Santosh, D.; Achar, S.; Jawahar, C. Autonomous image-based exploration for mobile robot navigation. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 2717–2722.
34. Gao, W.; Booker, M.; Adiwahono, A.; Yuan, M.; Wang, J.; Yun, Y.W. An improved frontier-based approach for autonomous exploration. In Proceedings of the 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, 18–21 November 2018; pp. 292–297.
35. Muravyev, K.; Bokovoy, A.; Yakovlev, K. Enhancing exploration algorithms for navigation with visual SLAM. In Proceedings of the Russian Conference on Artificial Intelligence, Taganrog, Russia, 11–16 October 2021; pp. 197–212.
36. Kojima, N.; Deng, J. To Learn or Not to Learn: Analyzing the Role of Learning for Navigation in Virtual Environments. *arXiv* **2019**. [CrossRef]
37. Mishkin, D.; Dosovitskiy, A.; Koltun, V. Benchmarking Classic and Learned Navigation in Complex 3D Environments. *arXiv* **2019**. [CrossRef]
38. Gupta, S.; Tolani, V.; Davidson, J.; Levine, S.; Sukthankar, R.; Malik, J. Cognitive Mapping and Planning for Visual Navigation. *arXiv* **2017**. [CrossRef]
39. Staroverov, A.; Yudin, D. A.; Belkin, I.; Adeshkin, V.; Solomentsev, Y. K.; Panov, A. I. Real-Time Object Navigation with Deep Neural Networks and Hierarchical Reinforcement Learning. *IEEE Access* **2020**, *8*, 195608–195621. [CrossRef]
40. Staroverov, A.; Panov, A. Hierarchical Landmark Policy Optimization for Visual Indoor Navigation. *IEEE Access* **2022**, *10*, 70447–70455. [CrossRef]
41. Rana, K.; Dasagi, V.; Haviland, J.; Talbot, B.; Milford, M.; Sünderhauf, N. Bayesian Controller Fusion: Leveraging Control Priors in Deep Reinforcement Learning for Robotics. *arXiv* **2023**, arXiv:2107.09822.
42. Rozenberszki, D.; Majdik, A.L. LOL: Lidar-only odometry and localization in 3D point cloud maps. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 4379–4385.
43. Ramakrishnan, S.K.; Gokaslan, A.; Wijmans, E.; Maksymets, O.; Clegg, A.; Turner, J.; Undersander, E.; Galuba, W.; Westbury, A.; Chang, A.X.; et al. Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI. *arXiv* **2021**. [CrossRef]
44. Savva, M.; Kadian, A.; Maksymets, O.; Zhao, Y.; Wijmans, E.; Jain, B.; Straub, J.; Liu, J.; Koltun, V.; Malik, J.; et al. Habitat: A Platform for Embodied AI Research. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019.