DEU FMD 27(80) (2025) 224-233

Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi
Dokuz Eylul University Faculty of Engineering Journal of Science and Engineering
Elektronik/Online ISSN: 2547-958X

RESEARCH ARTICLE / ARAŞTIRMA MAKALESI

Indoor Visual Navigation Based on Deep Reinforcement Learning with Neural Ordinary Differential Equations

Nöral Adi Türevsel Denklemler ile Derin Pekiştirmeli Öğrenmeye Dayalı İç Mekan Görsel Navigasyonu

Berk Ağın [1], Güleser Kalaycı Demir [2*]

[1] Dokuz Eylul University, The Graduate School of Natural and Applied Sciences, İzmir, TÜRKİYE
[2] Dokuz Eylul University, Department of Electrical and Electronics Engineering, İzmir, TÜRKİYE

Corresponding Author / Sorumlu Yazar *: guleser.kalayci@deu.edu.tr

Abstract

Recently, Deep Reinforcement Learning (DRL) has gained attention as a promising approach to tackle the challenging problem of mobile robot navigation. This study proposes reinforcement learning utilizing Neural Ordinary Differential Equations (NODEs), which offer effective training and memory capacities, and applies it to model-free point-to-point navigation task. Through the use of NODEs, we achieved improvements in navigation performance as well as enhancements in resource optimization and adaptation. Extensive simulation studies were conducted using real-world indoor scenes to validate our approach. Results effectively demonstrated the effectiveness of our proposed NODEs-based methodology in enhancing navigation performance compared to traditional ResNet and CNN architectures. Furthermore, curriculum learning strategies were integrated into our study to enable the agent to learn through progressively more complex navigation scenarios. The results obtained indicate that this approach facilitates faster and more robust reinforcement learning.

Keywords: *Neural ordinary differential equations, resnet, deep reinforcement learning, visual navigation, mobile robot*

Öz

Son yıllarda, Derin Pekiştirmeli Öğrenme (DPÖ), mobil robot navigasyonun zorlu sorunlarını çözmek için umut vadeden bir yaklaşım olarak ortaya çıkmıştır. Bu çalışma, etkili eğitim ve bellek avantajı sunan Nöral Adi Türevsel Denklemler (NATD) kullanarak pekiştirmeli öğrenme yöntemini önermekte ve modelden bağımsız, noktadan-noktaya navigasyon için uygulamaktadır. NATD kullanımıyla, navigasyon performansında artış ve kaynak optimizasyonu ile adaptasyonda iyileştirme sağlanmıştır. Yaklaşımımızı doğrulamak için, gerçek dünya iç mekan sahneleri kullanılarak kapsamlı simulasyon çalışmaları yapıldı. Sonuçlar, önerdiğimiz NATD tabanlı metodolojinin, geleneksel ResNet ve CNN mimarilerine göre navigasyon performansını artırmada etkili olduğunu göstermiştir. Ayrıca, müfredat öğrenme stratejileri çalışmamıza entegre edilmiş ve ajanın aşamalı olarak daha karmaşık navigasyon senaryoları üzerinden öğrenmesi sağlanmıştır. Elde edilen sonuçlar, bu yaklaşım ile daha hızlı ve daha gürbüz pekiştirmeli öğrenmenin gerçeklenebildiğini göstermektedir.

Anahtar Kelimeler: *Nöral adi türevsel denklemler, resnet, derin pekiştirmeli öğrenme, görsel navigasyon, mobil robot*

1. Introduction

In recent years, there has been a notable increase in the utilization of artificial intelligence in autonomous robots across a range of industries including logistics, finance, automotive manufacturing, and agriculture. Deep learning and reinforcement learning, in particular, have played crucial roles in advancing these systems' capabilities [1-6]. However, navigation remains a major challenge for both mobile and robotic platforms. To overcome this, it is essential to integrate the robot's perception and planning processes seamlessly, enhancing its overall navigational effectiveness.

Deep neural networks have proven instrumental in addressing the integration of perception and control, particularly in navigation tasks [7-9]. The navigation problem is a critical challenge for robotic systems. Traditional navigation algorithms, such as Simultaneous Localization and Mapping (SLAM) [10], path planning [11-13], and trajectory planning [14], are commonly used to address this challenge. Visual navigation is one of the key strategies for solving complex navigation problems, as vision-based activities enable mobile robots to understand dynamics and interact with their environments. Environments can be categorized as either map-known or map-less [15], and various navigation tasks include point-to-point (P2P), object-goal, and area-goal navigation [7] .

Combining visual navigation with deep learning methods provides effective solutions for these tasks. Numerous approaches have been proposed for navigation applications, such as obstacle avoidance [9], visual recognition [16], and deep learning-based localization [17]. Additionally, many Reinforcement Learning (RL) based techniques such as the Partially Observable Markov Decision Process (POMDP) have been used for visual navigation [8], [18]. Recent researches have

DOI: https://doi.org/10.21205/deufmd.2025278008

Geliş Tarihi / Received: 02.07.2024
Kabul Tarihi / Accepted: 17.08.2024

highlighted end-to-end navigation strategies for mobile robots [6], [19].

Deep neural networks include various architectures such as Convolutional Neural Networks (CNN) [20], ResNet [21], AlexNet [22], GoogleNet [23], and Neural Ordinary Differential Equations (NODEs) [24]. In the literature, several studies demonstrate the efficient learning performance and memory efficiency achieved by employing NODEs in conjunction with reinforcement learning approaches to tackle complex problems. One of the studies, by Ainsworth et al. [25], introduces the concept of Continuous-Time Policy Gradient (CTPG) to facilitate faster learning, providing an efficient and accurate gradient estimator for continuous-time systems. Another significant study, conducted by Yildiz et al. [26], proposes a continuous-time model-based reinforcement learning framework utilizing the actor-critic method, inferring state evolution differentials through Bayesian neural ODEs. The study which contributes to the literature on NODEs and reinforcement learning, presented by Meleshkova et al. [27], explores their application in the realm of robotics. Additionally, Du et al. [28] present a pioneering study on Model-based RL for Semi-Markov Decision Processes (SMDPs) using NODEs, incorporating actions and time intervals into neural ODEs to model continuous-time dynamics. Another study by Zhao et al. [29] introduce a primary controller that combines Control Barrier Function and Control Lyapunov Function frameworks with the Soft Actor-Critic (SAC) algorithm [30] for systems whose dynamics are approximated by NODEs. Lastly, Zhao et al. [29] presents an ODE-based recurrent model combined with a model-free reinforcement learning (RL) framework to address partially observable Markov decision processes (POMDPs), demonstrating efficacy across various continuous control and meta-RL tasks.

NODEs' efficient training and memory capabilities suggest promising enhancements in resource optimization, adaptability, and scalability, all essential for mobile robot navigation. Hence, this study proposes an autonomous agent model employing deep reinforcement learning augmented with NODEs. This strategy not only improves the agent's navigation ability but also maintains memory efficiency throughout training. Through extensive experimentation and evaluation, we validate the efficacy of our approach in improving navigation performance for point-to-point tasks, particularly in environments with limited prior knowledge or unpredictable layouts. To the best of our knowledge, we are the first to apply neural ODE-based network for navigation tasks. Our proposed approach also includes a range of supplementary techniques aimed at enhancing the efficacy of our autonomous agent model. Firstly, Curriculum Learning [31] is employed to facilitate a gradual learning process. Additionally, we integrate data fusion techniques to enrich the agent's understanding of its environment. Furthermore, we utilize reward shaping strategies to guide the agent towards more efficient navigation behaviors, incentivizing actions that contribute to successful indoor navigation tasks.

The remainder of the paper is structured as follows: Section 1 presents the problem and review of relevant studies. Section 2 outlines the methodologies and technical background employed in addressing the reinforcement learning and Neural Ordinary Differential Equations. Section 3 provides insight into the implementation details of our proposed approach. In Section 4, we present the analyses and results derived from our experimental investigations. Lastly, Section 5 offers discussions and conclusions regarding the performance of the autonomous agent for P2P navigation.

2. Background

2.1. Reinforcement learning

Deep reinforcement learning is able to handle high-dimensional, continuous input and is capable of learning from interactions in complex environments. Therefore, it offers a powerful and flexible approach to obtain more capable/adaptable agents for navigation tasks [6, 15].

In our reinforcement learning setup, an agent interacts with an environment across multiple discrete time steps. At each time step $t = 0,1,2,\cdots$, the agent observes a state $s_t \in S$, receives a reward r_t and takes an action $a_t \in A$. When an agent reaches a terminal state, this process restarts. The return

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots \gamma^{T-t-1} r_T = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \tag{1}$$

is the total discounted reward from the current time t to the final time step T with a discount factor $\gamma \in (0,1]$. The goal of an agent is to learn a policy π that chooses actions such that the expected return is maximized.

At this point, it should be noted that if the agent only ever takes actions that it has already tried and found to be rewarding, it may miss out on better actions that it has not yet discovered. This is especially true in navigation problems where the space of possible actions is large, and the reward signal is sparse or noisy. Therefore, the process of actively seeking out new and potentially rewarding actions, called exploration, is incorporated into the learning algorithm to avoid getting stuck in suboptimal policies. The entropy that measures the uncertainty in the policy is one of the powerful tools for exploration[32], [33] . The maximizing the entropy of its policy encourages the agent to take more diverse and less predictable actions [32]. We augment the entropy into RL objective in such a way that the objective seeks to find a policy that maximizes both the expected reward and the entropy of the policy. Then the maximum-entropy RL objective function is

$$\pi^* = \arg\max_{\pi E_\pi} \left[\sum_{t=0}^{T} \gamma^t r_t + \alpha H^\pi \right] \tag{2}$$

where

$$H^\pi = -\sum \pi \log \pi \tag{3}$$

is the entropy of policy π. The parameter α controls the trade-off between the entropy term and the reward in the objective function and thus determines how much randomness the optimal policy exhibits.

The agent learns to directly optimize a policy that maps the current state of the environment to a probability distribution over possible actions in a policy-based approach [34]. The policy function (e.g., a neural network) is defined by a set of adjustable parameters θ that are updated in the direction that yields higher rewards.

2.2. Proximal Policy Optimization(PPO)

In our study, we use on-policy reinforcement learning algorithms instead of off-policy ones to be able to learn quickly and adapt to changes in the environment [34]. On-policy algorithms also tend to be more stable than off-policy algorithms since they update the policy based on the data they are currently collecting. Some common on-policy algorithms include policy gradient methods

[25], actor-critic methods [26], Proximal Policy Optimization (PPO) [34]. Especially, PPO provides simple, stable, and good-performance solutions in a wide range of reinforcement learning problems, including games, robotics, and natural language processing.

The main idea behind PPO is to make small updates to the policy at each iteration while ensuring that the new policy closely resembles the previous one. This is achieved by using a clipping mechanism that limits the ratio between the current and former policy in the range of $[1-\epsilon, 1+\epsilon]$. Here, the clipping threshold ϵ is a hyperparameter that determines how much the policy can change in each iteration. The clipped surrogate objective is given by

$$L^{CLIP}(\theta) = E\left[\min\left(\rho_t(\theta)\widehat{A_{\theta_\pi}}(s_t,a_t),\ clip(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A_{\theta_\pi}}(s_t,a_t)\right)\right] \quad (4)$$

where θ is the policy parameters,

$$\widehat{A_{\theta_\pi}}(s_t,a_t) = E_{s_{t+1:\infty};\alpha_{t+1:\infty}}\left[\sum_{l=0}^{\infty} r_{t+l}\right] - E_{s_{t+1:\infty};\alpha_\infty}\left[\sum_{l=0}^{\infty} r_{t+l}\right] \quad (5)$$

is the advantage function that estimates the advantage of taking a certain action, and

$$\rho_t(\theta) = \frac{\pi_\theta(s_t,a_t)}{\pi_{\theta_{old}}(s_t,a_t)} \quad (6)$$

is the probability ratio between the current policy $\pi_\theta(s_t,a_t)$ and old policy $\pi_{\theta_{old}}(s_t,a_t)$.

2.3. Neural Ordinary Differential Equations

Neural ordinary differential equations are a class of deep learning models that represent the dynamics of hidden states as a continuous function parameterized by neural networks [24]. The agent's behavior, or policy, is encoded within the neural network layers, making the quality of the policy dependent on the evolving variables of the network. The initial concept of NODEs was inspired by a single layer of ResNets [21] and the Euler discretization method. A residual layer updates the hidden state h_t at time t by adding a function $f(\theta_t, h_{t-1})$ over the previous state h_{t-1} in ResNets. Mathematically,

$$h_t = h_{t-1} + f(\theta_t, h_{t-1}) \quad (7)$$

Contrary to discrete updates in ResNets, Neural Ordinary Differential Equations take a continuous approach by parameterizing the derivative of the hidden state through a neural network $f(h(t), t, \theta)$:

$$\frac{dh(t)}{dt} = f(h(t), t, \theta) \quad (8)$$

Instead of updating the hidden state at fixed intervals, NODEs define a smooth evolution of the hidden state over time by solving an initial value problem using the neural network function f with the parameters θ. Thus, the hidden state at any time τ can be determined using an ODE solver:

$$h(\tau) = h(t_0) + \int_{t_0}^{\tau} f(h(t), t, \theta)dt \quad (10)$$

$$= \text{ODESolve}(h(t_0), f, t_0, \tau, \theta) \quad (11)$$

where $h(t_0)$ represents the initial conditions of the integration, $t \in (t_o, \tau)$ denotes the interval over which the integration is performed, and ODESolve is a numerical ODE solver.

Since this continuous framework can potentially offer improved accuracy and robustness in capturing complex patterns, we present reinforcement learning implemented with NODEs.

3. Learning Setup

We conducted visual navigation experiments within the GibsonEnv simulation environment [35], employing various neural network architectures and sensor inputs. The initial phase of our experiments utilized sensor data, including camera imagery and proprioceptive information extracted from the environment. The camera sensors provided RGB and depth images of the building scenes, while proprioceptive sensors offered data on the robot's position, orientation, as well as linear and angular velocity. These diverse inputs were then fed into the neural networks to derive an optimal navigation policy. Subsequently, we focused on enhancing navigation performance by leveraging robust neural network models. We investigated the efficacy of Multi-layer Perceptron (MLP), ResNets, and NODEs, comparing their respective performances. In the context of reinforcement learning, crucial components include observations, actions, rewards, and goals. The learning setup details of our implementation are provided in followings:

Observations: We utilize two types of observations in our experiments. The first type comprises camera images, including depth and RGB images, which are fed into the neural networks and employed in the reward function to facilitate obstacle avoidance. The second type consists of sensor data, which includes height difference, vertical and horizontal angles of the robot, linear velocities, roll and pitch angles of the robot's body, angular velocities at joints, and contact values of the robot's wheels. The latter consisted of eight different joint values that sense the robot's interaction with the ground. This proprioceptive sensor data is represented by a 23×1 dimensional vector.

Action Space: We employed a discrete action space in our study. The agent has five possible actions: move left, move right, move forward, move backward, and stop.

Rewards: We investigate a combination of continuous and sparse rewards designed to facilitate complex navigation tasks. These rewards typically encourage advancing closer to the target point in a single step, overall progress, avoiding collisions with objects or obstacles, and maintaining a straight trajectory towards the endpoint.

Goals: The goal of the task is defined as point to point navigation without any collision.

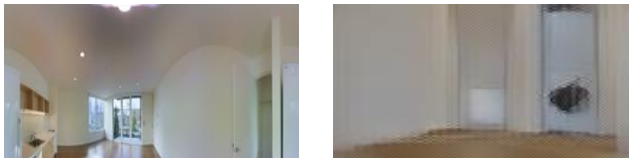
3.1. Environment

We use a Gibson Simulation Environment [35] for training and evaluating our model. Actions can be performed within this environment, and their outcomes observed, in a three-dimensional (3D) space with real-world image datasets. Examples of the environments, taken from the buildings 'Euharlee' and 'Aloha' are shown in Figure 1.

We import a husky unmanned ground vehicle as our agent into the Gibson framework. This agent operates within the constraints of space and physics, facilitated by integration with a physics engine [36]. While adhering to these constraints, the agent can perform a wide range of mobility tasks. Gibson continuously

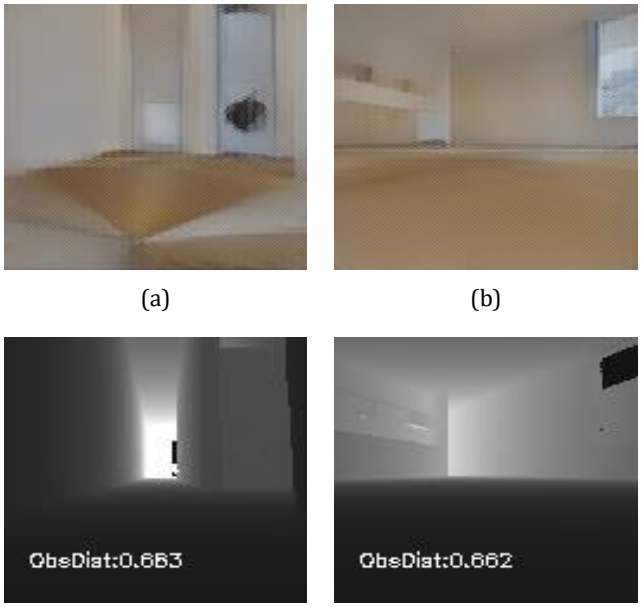
provides visual observations in the form of RGB images from various viewpoints, utilizing an onboard camera.

Alongside RGB images, Gibson also generates depth information and proprioceptive sensory data, including joint positions, angle velocity, robot orientation relative to the navigation target, position, velocity, and more. Examples of RGB and depth images utilized in this study are illustrated in Figure 2.



(a) (b)

Figure 1. Example the panoramic photos of the (a) Euharlee and (b) Aloha buildings [35].



(a) (b)

(c) (d)

Figure 2. Captured sample images: (a) 165th RGB image, (b) 312th RGB image (c) 165th depth image, (d) 312th depth image.

4. Visual Navigation

Visual information from the surrounding environment is used to determine and follow a path without requiring a map of the environment in our study. We address the point-to-point visual navigation task across various setups that differ in navigation complexity. Our proposed approach incorporates reinforcement learning with a NODEs-enhanced network structure for better environment understanding and decision-making processes. To further enhance the learning process, we integrate curriculum learning and reward shaping to encourage desirable behaviors. The details are given in the following subsections.

(a) (b)

Figure 3. Samples of navigation plans for (a) low complexity and (b) high complexity tasks. Large red and green balls show the starting and target points of the plans, respectively. Small red balls represent waypoints during navigation.

4.1. Curriculum Learning

We apply curriculum learning that involves organizing the training data in a simple to complex order. The idea behind curriculum learning is to allow the learning algorithm to gradually learn from simpler examples and then move on to more complex ones, mimicking how humans learn [31].

We quantify the complexity of navigation using Equation (12) which calculates the ratio between d_{SP} and d_{SD}.

$$A = \frac{d_{SP}}{d_{SD}} \tag{12}$$

where d_{SP} is the shortest path length taken by the agent to reach its destination and d_{SD} is the straight line distance between the agent's starting point and the destination. This ratio indicates the disparity between the shortest path distance d_{SP} and the actual distance traveled d_{SD} by the agent. A larger difference signifies a more intricate navigation task, indicating that the agent must navigate through a more obstacle-ridden environment to reach its destination. Figure 3 provides visualization examples of navigation plans within the environment, illustrating various levels of complexity.

Before starting to learning process, we organize the navigation tasks into a sequence of progressively more difficult navigation ones. Table 1 shows different P2P training tasks, their starting and goal locations, corresponding shortest path and straight path distances and determined task complexity using Equation 12.

Table 1. Example P2P navigation task order for curriculum learning.

Starting Point				Goal Point			d_{SD}	d_{SP}	A
X	Y	Z	Angle	X	Y	Z			
-1.97	3.56	0	3.10	-2.48	-3.39	0	6.97	6.97	1
-1.97	3.56	0	3.10	-2.57	-2.53	0	6.12	6.12	1
2.09	3.13	0	1.18	-1.92	1.2	0	4.45	4.45	1
-1.92	1.2	0	5.23	-2.47	-5.56	0	6.79	6.8	1.002
-0.05	3.57	0	2.50	-2.57	-2.53	0	6.60	6.78	1.04
2.09	3.13	0	1.18	-2.57	-2.53	0	7.33	7.84	1.07
2.09	3.13	0	1.18	-2.57	-4.23	0	8.71	9.54	1.10

After ordering the task, the agent starts learning using a simple P2P task with a navigation complexity level of one. As the agent progresses through subsequent episodes, the complexity level of these tasks is gradually increased in a stepwise fashion.

Experiments shows that, by using curriculum learning approach, our learning algorithm improves its effectiveness and converges to a better solution.

4.2. Reward shaping

Reward shaping is a crucial technique in reinforcement learning that involves modifying the reward function to provide additional feedback to the learning agent. In our study, we incorporate navigation-specific knowledge into the reward structure to enable the agent to explore the environment more efficiently and discover optimal policies more quickly. We observe in the experiments that reward shaping guides the agent away from suboptimal behaviors and towards more desirable actions, enhancing the overall performance and stability of the learning.

In our study, we utilized a combination of continuous and sparse rewards to facilitate complex navigation tasks. We observed that maintaining a critical balance between the different reward components was essential; if one component dominates the others, it leads to an unstable policy. The various reward components defined for effective reward shaping in this study are as follows:

i. We reward the agent based on its ability to move closer to the target point in one step. This reward accounts for the agent's potential and is defined in Equation (13).

$$r_{pot} = \frac{\sqrt{\left(x - x_{goal}\right)^2 + \left(y - y_{goal}\right)^2 + \left(z - z_{goal}\right)^2}}{f} \quad (13)$$

where $x, y\ and\ z$ represent the current coordinate of the robot, $x_{goal}, y_{goal}\ and\ z_{goal}$ are the goal coordinates and f is the frame rate of the camera.

ii. We also reward the agent for progress by calculating the difference between the old potential reward $r_{pot_{old}}$ and the new potential reward $r_{pot_{new}}$ as it approaches the target point:

$$r_{prog} = r_{pot_{old}} - r_{pot_{new}} \quad (14)$$

iii. The agent receives a binary living reward based on its ability to drive without tipping. This reward is sparse, taking a value of either 1 or 0. The agent earns a reward of 1 if it drives without tipping.

iv. We correlate the negative-valued obstacle reward $r_{obstacle}$with the depth images from the camera. The reward is scaled based on the mean value of the observation window (80 $\times$ 80) in the depth image.

v. If the agent hits any object or obstacle, it incurs a crashing penalty $r_{crash} = -1$.

vi. We encourage the agent to move as straight as possible by penalizing any attempts to turn left or right. For this purpose, we define a negative-valued steering reward and set it to -0.1.

vii. We also define an angle cost r_{return} to penalize the agent if the angle between its heading and the direction to the target point increases.

viii. The angle cost r_{return} and the steering cost $r_{steering}$ are used to fine-tune the heading angle and encourage the agent to move straight towards the target without excessive maneuvering. These cost values range from -0.5 to 0.5.

ix. Additionally, we use a binary terminal reward $r_{terminal}$ of 1 to encourage the agent to reach the endpoint.

4.3. Neural network architecture

PPO for reinforcement learning requires parameterizing the policy function, which maps states to actions. Neural networks have found frequent application in representing policies as continuous and differentiable functions, owing to their robust function approximation capabilities. Additionally, gradient information to update the policy parameters of PPO can be efficiently computed in neural networks through backpropagation.

In this study, we employ NODEs to ascertain the optimal policy for point-to-point navigation. Our input comprises RGB, depth, and proprioceptive sensor information. Multi-Layer Perceptron (MLP) architecture consistently serves as the choice for processing of proprioceptive sensor data in all experimental analyses. MLP structure has 4 hidden layers, with each hidden layer containing 128 neurons.

Nevertheless, apart from NODEs, distinct architectures (CNN and ResNets) are explored for the depth and/or RGB input data.

The architecture of the NODEs network, illustrated in Figure 4, processes depth and/or RGB images. It consists of ODE blocks that handle the entire feature extraction chain. In the NODEs network, we implement two convolutional layers as a pre-processing stage before the ODE blocks. These convolutional layers, with 8x8 and 4x4 convolutional filters respectively, help map the input image into an appropriate state space. The function within the ODE block is implemented to a standard residual block used in residual networks. Following the ODE blocks, the output is processed through a global average-pooling operation and then passed through a fully connected layer with softmax activation function.

Additionally, we also utilize CNN and ResNet architectures for processing depth and/or RGB images. These architectures provide a benchmark to evaluate the performance of the NODEs network. Figure 5 illustrates the construction of the ResNet architecture, showcasing its characteristic residual blocks that enable the training of very deep networks by addressing the vanishing gradient problem.

5. Experiments and Results

We conducted our indoor navigation experiments using the Gibson simulation environment on a setup equipped with an Intel i7-13700 processor, Nvidia GTX-4070 GPU, and 64 GB of RAM. We tested three different neural network architectures: CNN+MLP, ResNet+MLP, and NODEs+MLP. As previously discussed, MLP structure is used for proprioceptive sensor inputs, while CNN, ResNet, and NODEs are employed for RGB/Depth image inputs. The configuration parameters for our simulations are detailed in Table 2.

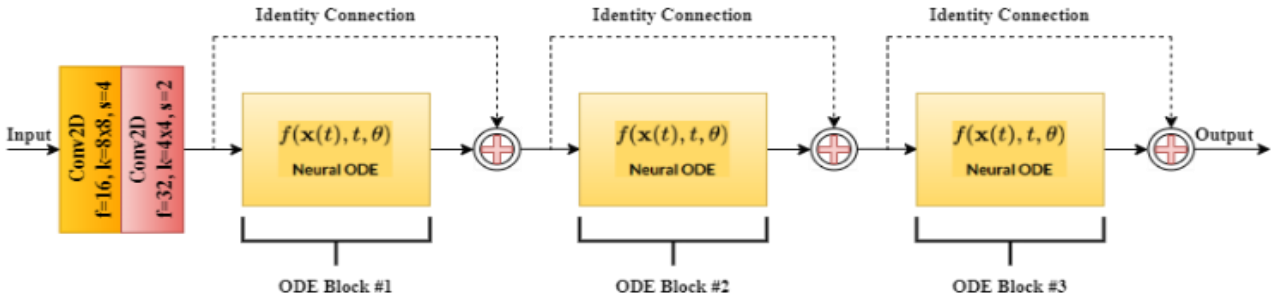


Figure 4. NODEs network architecture with 3 identity block.

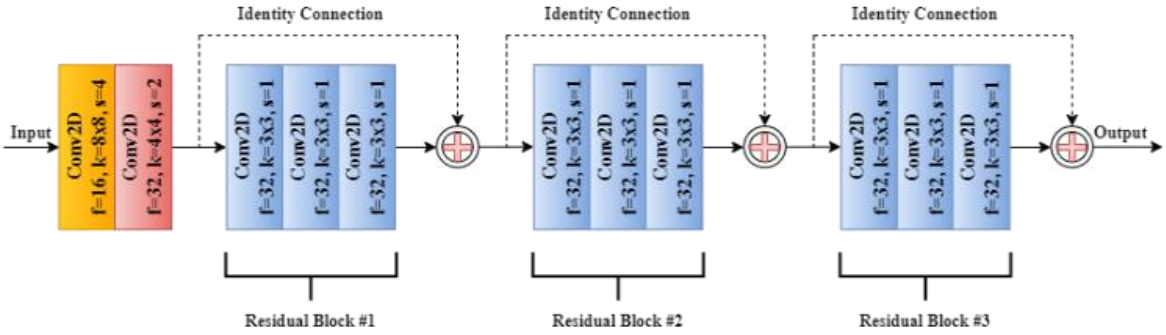


Figure 5. ResNet network architecture with 3 identity block.

Table 2. Configuration parameters of simulations.

Timesteps	# of Episodes	# of Iterations	Resolution
500	30	151	128

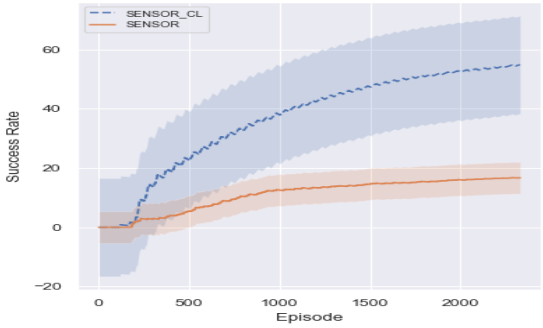
We used Success Path Length (SPL) to evaluate the navigation success of different learning strategies. Equation (15) defines the Success Path Length,

$$SPL = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{\max(p_i, l_i)} \tag{15}$$

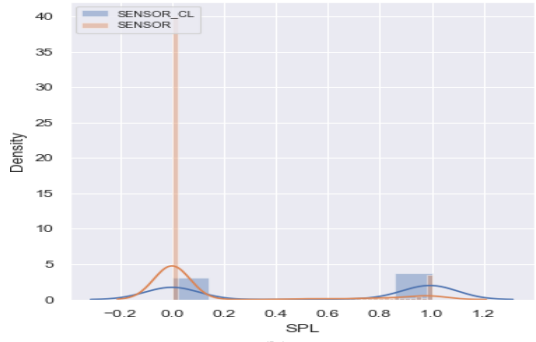
where N is the number of episodes, l_i is the shortest path length derived from the Euclidean distance, p_i is the total distance traveled by the agent, and S_i is a binary constant that indicates the success of the episode.

We analyze the experimental results through two SPL-related graphs: a histogram of Success Path Length (SPL) and a Success Rate metric. The SPL histogram shows the density of episodes corresponding to various SPL values, illustrating the efficiency and consistency of success paths during the experiments. The Success Rate indicates the overall effectiveness of the method and is calculated as the ratio of successful episodes to the total number of episodes.

To evaluate the impact of curriculum learning on the training process, we simulated neural network structures using i) only sensor data (blind mode) and ii) depth data. Figure 6 shows the Success Rate and SPL histogram results for scenarios where curriculum learning was either implemented or not. Figure 6-a illustrates that reinforcement learning without curriculum learning progresses at a slower pace and rapidly reaches a plateau. As a result, the success rate without curriculum learning remains around 18. In contrast, with the incorporation of curriculum learning, the success rate significantly increases to approximately 55. Figure 6-b shows that episodes with an SPL of zero occur more frequently in the absence of curriculum learning.



(a)



(b)

Figure 6. Comparison of (a) success rates and (b) histograms depicting success path lengths with and without curriculum learning (sensor_cl vs. sensor) in blind mode.

However, when curriculum learning is implemented, episodes with an SPL of 1 are considerably more common. To assess the impact of curriculum learning, the rewards obtained during reinforcement learning are also analyzed. As illustrated in Figure 7, the integration of curriculum learning consistently results in significantly higher reward values in blind mode.

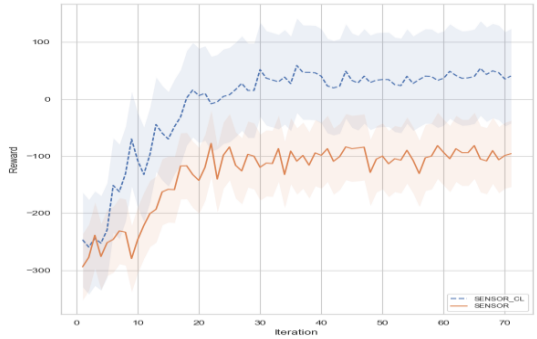
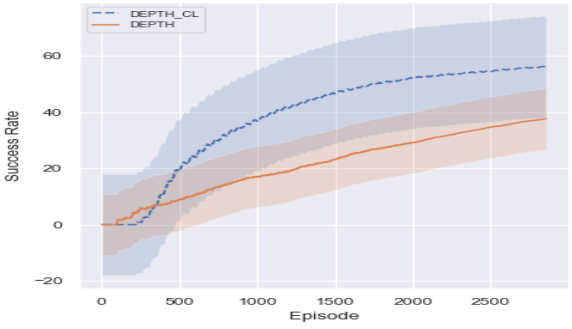
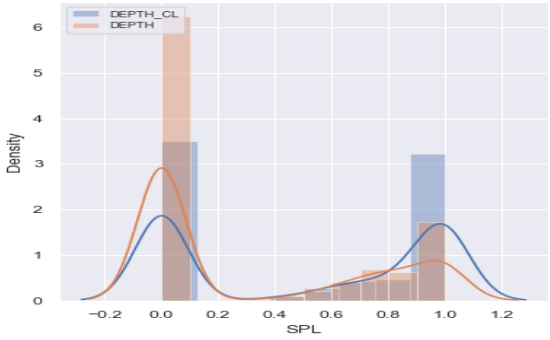


Figure 7. Rewards in reinforcement learning with and without curriculum learning (sensor_cl vs. sensor) in blind mode.

The success rate, SPL, and reward outcomes obtained when using depth images as input are presented in Figures 8 and 9. The results for depth images indicate higher success rates, a greater number of episodes where SPL equals one, and increased rewards with the integration of curriculum learning. For both input types (sensor and depth), the findings show that the integration of curriculum learning significantly enhances the performance, effectiveness, and robustness of navigation.



(a)



(b)

Figure 8. Comparison of (a) success rates and (b) histograms depicting success path lengths with and without curriculum learning (DEPTH_CL vs. DEPTH) for depth images.

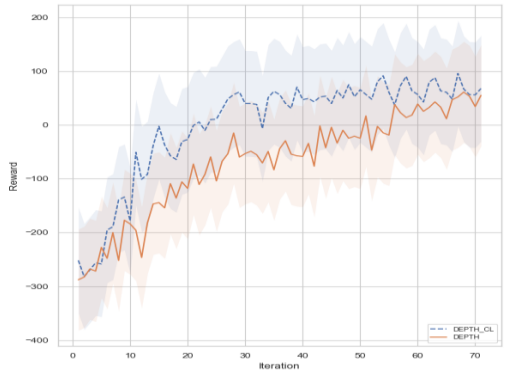


Figure 9. Rewards in reinforcement learning with and without curriculum learning (DEPTH_CL vs. DEPTH) for depth images.

The training results of different neural network architectures with curriculum learning are presented in Figure 10. The inputs for reinforcement learning consist of both depth images and proprioceptive sensor data. As previously noted, proprioceptive data is processed solely by the MLP network, while depth data is processed by either NODEs, CNN, or ResNets. To evaluate the results, we consider the cumulative reward, entropy loss, value function loss, and surrogate function value. Based on cumulative reward results, NODEs achieve higher rewards while ResNet collects the least reward. Entropic loss remains minimal in NODEs and CNN architectures, while NODEs exhibit the lowest value losses. Regarding surrogate function values, CNN and NODEs perform comparably, whereas ResNet shows lower values. The results in Figure 10 show that the NODE+MLP network architecture exhibits better training performance for the point-to-point navigation problem, as evidenced by higher rewards and lower entropy and value function losses.

In the learning phase of NODEs and CNN networks, the success rates indicate that roughly 80% of the episodes end successfully, whereas ResNet shows a comparatively lower success rate of around 70% (Figure 11-a). SPL calculations show that for both NODEs and CNN architectures, the agents reach their goal points following noticeably straighter paths compared to ResNet (Figure 11-b).

The training results for both RGB images and proprioceptive sensor data are given in Figure 12 and Figure 13. The results show that the use of NODEs presents high performance for P2P navigation training. Utilization of NODEs generally yields higher rewards and lower losses compared to CNN and ResNet. NODEs also demonstrates higher navigation performance according to success rate and SPL histogram metrics (Figure 13). It is noteworthy that CNN architecture does not perform as effectively on RGB images as it does on depth images.

The results of proposed NODEs integrated reinforcement learning are also examined in terms of the number of network parameters, average scores over the last 10 iterations, and total time elapsed. According to the results provided in Table 3, the NODE architecture has minimum parameter count (118,646) and achieve the highest average score (120.22). However, its training time is significantly higher compared to other architectures. The CNN architecture achieves the second highest average score using approximately six times more parameters than NODEs. However, its training time is approximately 22 times less than that of NODEs.

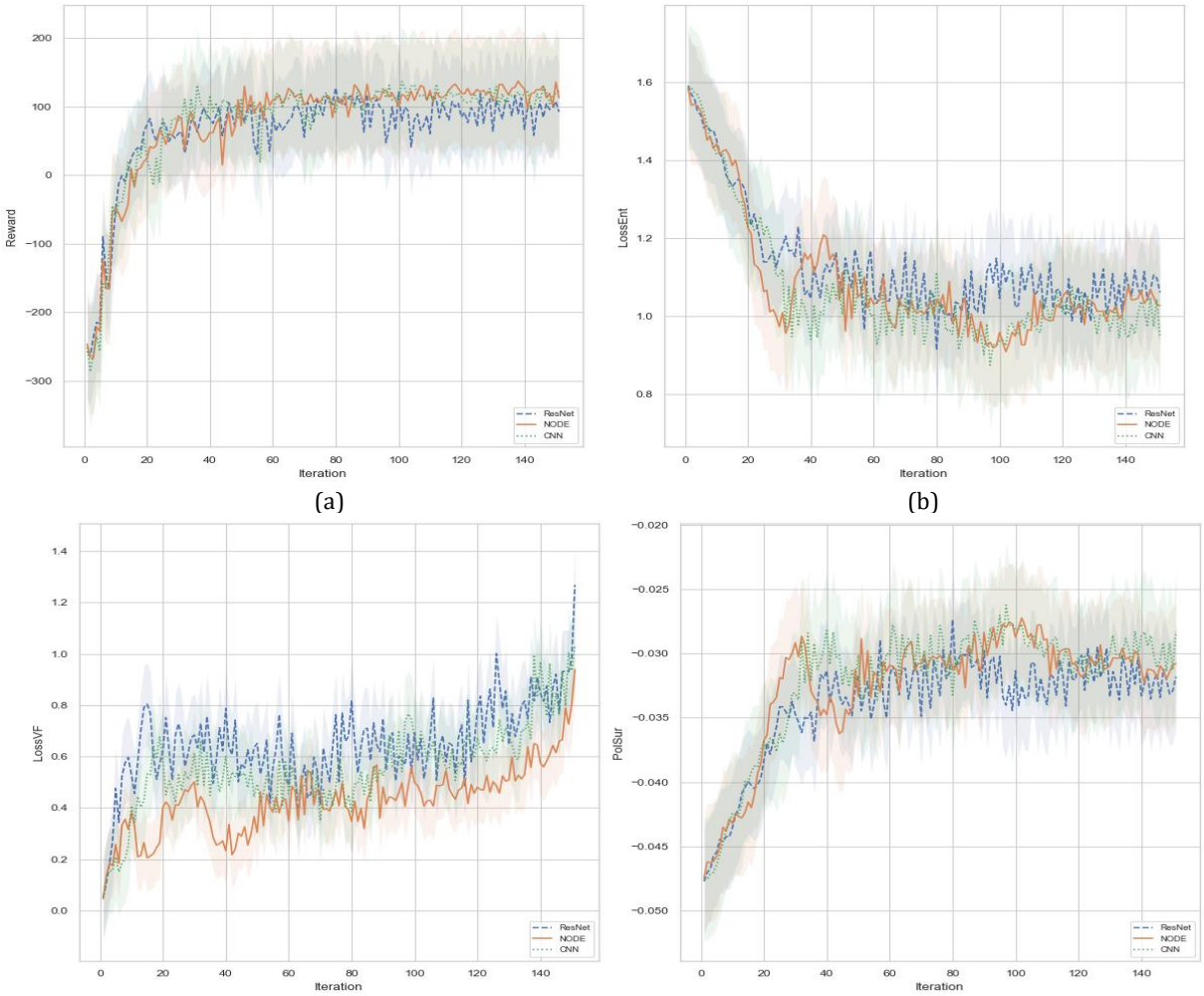


Figure 10. (a) Cumulative rewards, (b) entropy loss, (c) value function loss and (d) surrogate function values during the reinforcement training for both depth and sensor data inputs.

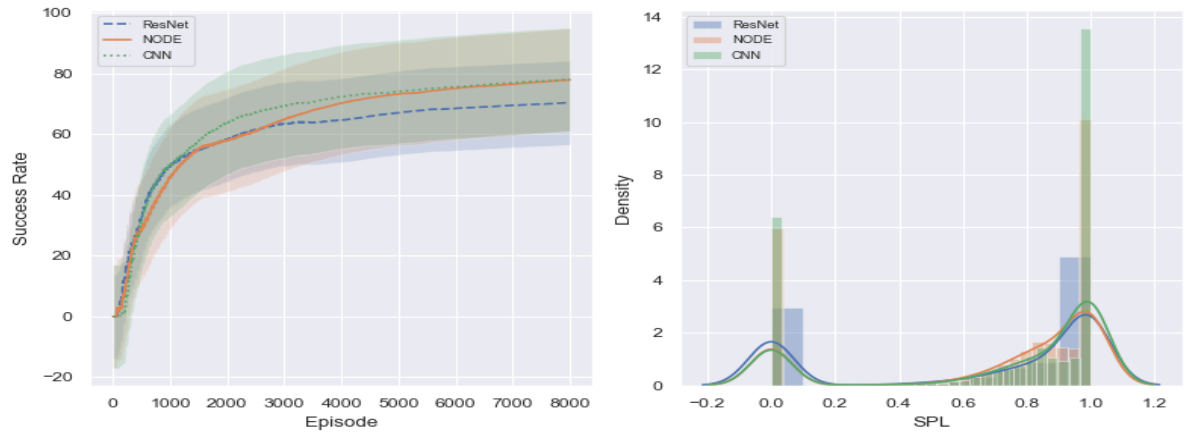


Figure 11. Comparison of (a) success rates and (b) histograms depicting success path lengths of different neural network architectures for both depth and sensor data inputs.

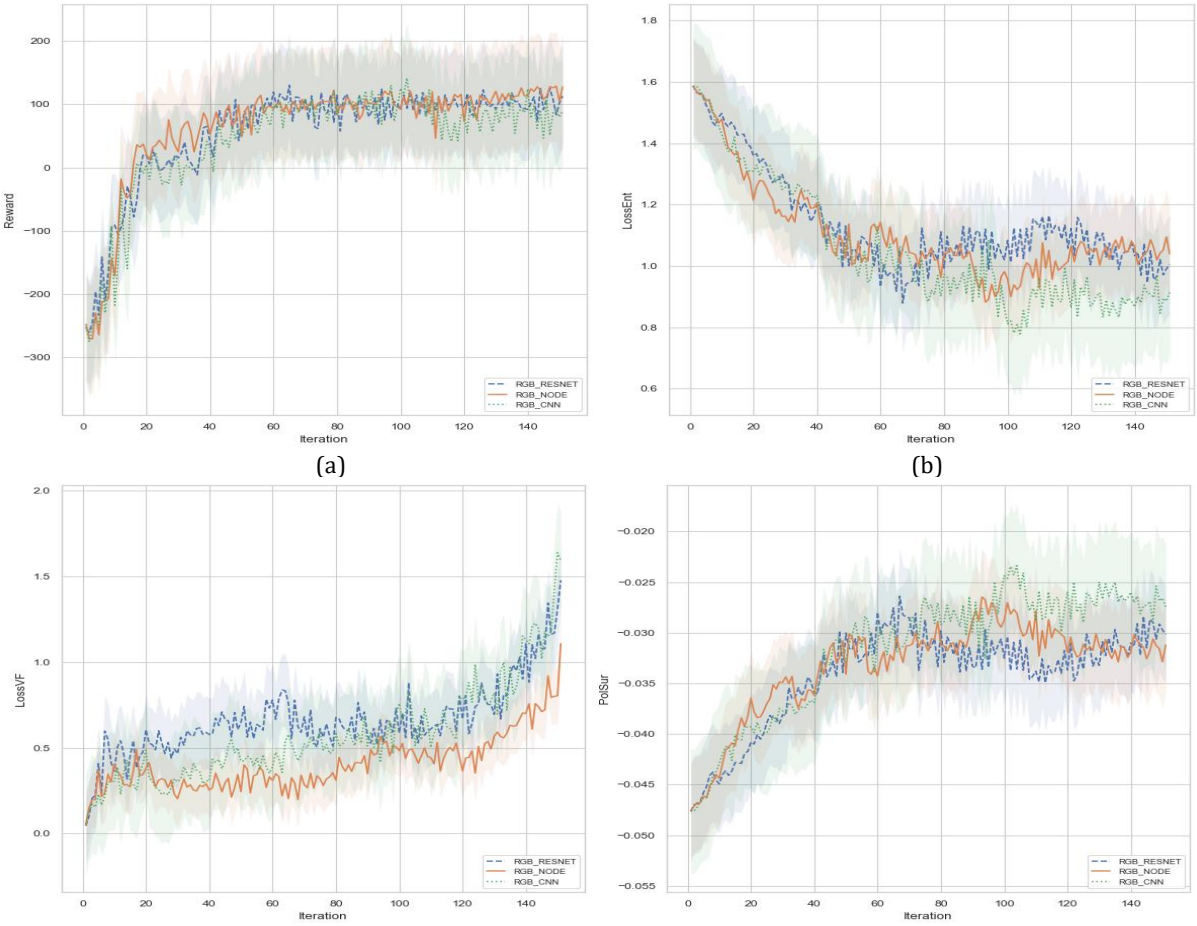


Figure 12. (a) Cumulative rewards, (b) entropy loss, (c) value function loss, and (d) surrogate function values during the reinforcement training for both RGB image and sensor data inputs.

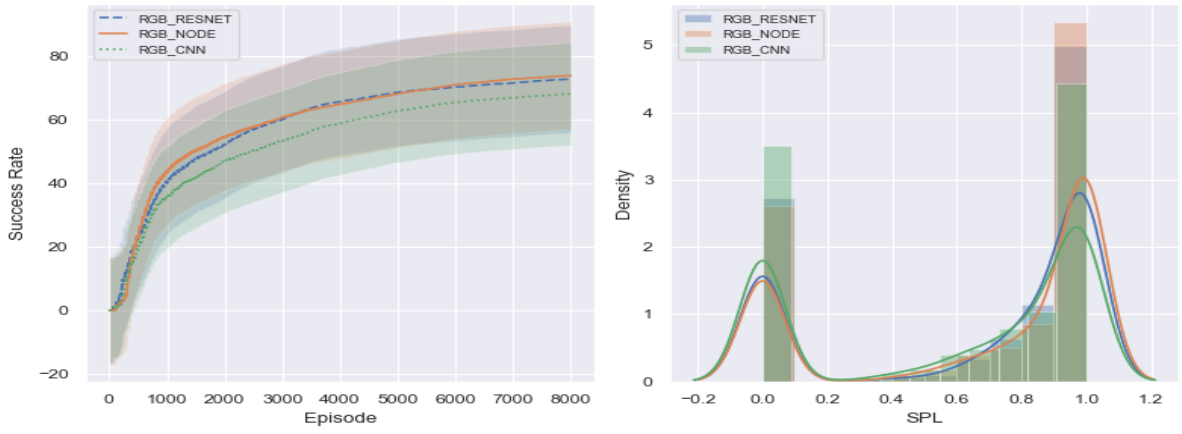


Figure 13. Comparison of (a) success rates and (b) histograms depicting success path lengths of different neural network architectures for both RGB image and sensor data inputs.

Table 3. Evaluation of NODEs, ResNet, and CNN Architectures

NN Architecture	# of Parameters	Average Score	Total Time Elapsed
NODEs	118646	120.22	22 Hour 24Min
ResNet	126726	94.16	1Hour 31Min
CNN	723366	113.52	1Hour 47Min

6. Conclusion

In this study, we have proposed an enhanced neural network architecture utilizing NODEs in model-free reinforcement learning for point-to-point navigation tasks. We performed several experiments with real-world perception in Gibson simulation environment and showed the effectiveness of integrating NODEs with a DRL approach. Our results indicate that NODEs outperform traditional ResNet, CNN architectures in terms of navigation performance and memory efficiency though NODEs require more training time. Additionally, leveraging

curriculum learning, we effectively guided the agent from low to high complexity paths. Our findings demonstrate that curriculum learning plays a pivotal role in having robust and powerful learning performance. Furthermore, our study achieves a balanced training process through the application of reward shaping techniques.

Our future research studies will focus on integrating semantic relation exploitation, necessitating the robot's comprehension of its surroundings and proactive decision-making. Additionally, we aim to enhance navigation performance by incorporating attention mechanisms into the fundamental blocks of Neural ODEs.

Ethics committee approval and conflict of interest statement

There is no need for an ethics committee approval in the current article. There is no conflict of interest with any person/institution in the current article.

Author Contribution Statement

This work is based on the first author's master's thesis titled 'Deep learning based visual navigation in indoor environments,' completed under the supervision of the second author. The first author contributed to the analysis, coding, simulation, and writing, while the second author contributed to the idea development, methodology, analysis, review and writing. Both authors contributed equally to this manuscript.

References

[1] Ferreira, B., Reis, J. 2023. A Systematic Literature Review on the Application of Automation in Logistics, Logistics, Vol. 7, no. 4, p. 80, DOI: 10.3390/logistics7040080.

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. 2013. Playing Atari with Deep Reinforcement Learning. arXiv, http://arxiv.org/abs/1312.5602 (Accessed: Jun. 12, 2024).

[3] Fox, I., Lee, J., Pop-Busui, R., Wiens, J. 2020. Deep Reinforcement Learning for Closed-Loop Blood Glucose Control, arXiv, http://arxiv.org/abs/2009.09051 (Accessed: Jun. 12, 2024).

[4] Yang, S. 2023. Deep Reinforcement Learning for Portfolio Management, Knowledge-Based Systems, Vol.278, https://doi.org/10.1016/j.knosys.2023.110905.

[5] Liu, X.-Y., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., Wang, C. 2022. FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance, arXiv, http://arxiv.org/abs/2011.09607. (Accessed: Jun. 12, 2024).

[6] Shi, H., Shi, L., Xu, M., Hwang, K.-S. 2020. End-to-End Navigation Strategy With Deep Reinforcement Learning for Mobile Robots, IEEE Trans. Ind. Inform., Vol. 16, no. 4, pp. 2393-2402, doi: 10.1109/TII.2019.2936167.

[7] Anderson, P., Chang, A., Chaplot, D.S., Dosovitskiy, A. et al. 2018. On Evaluation of Embodied Navigation Agents. arXiv, https://arxiv.org/abs/1807.06757 (Accessed: Sep. 23, 2023)

[8] López,, M.E., Bergasa, L. M., Escudero,M.S. 2003. Visually augmented POMDP for indoor robot navigation, in Applied Informatics, pp. 183-187.

[9] Gaya, J., Gon√ßalves, L., Duarte, A., Zanchetta, B., Drews-Jr, P., Botelho, S. 2016. Vision-based Obstacle Avoidance Using Deep Learning, doi: 10.1109/LARS-SBR.2016.9.

[10] Thrun, S. 2008. Simultaneous localization and mapping. In Robotics and cognitive approaches to spatial mapping, pp. 13-41, Springer Berlin Heidelberg.

[11] Balakrishnan, K., Chakravarty, P., Shrivastava, S. 2021. An A* Curriculum Approach to Reinforcement Learning for RGBD Indoor Robot Navigation. arXiv, http://arxiv.org/abs/2101.01774, (Accessed: Aug. 07, 2023).

[12] Stentz, A. 1994. Optimal and efficient path planning for partially-known environments, IEEE International Conference on Robotics and Automation, 1994, pp.3310-3317Vol.4. doi: 10.1109/ROBOT.1994.351061.

[13] LaValle, S.M. 1998. Rapidly-exploring random trees: A new tool for path planning, Technical Report 98-11, Comput. Sci. Dept, Iowa State Univ.

[14] Gasparetto, A., Boscariol, P., Lanzutti, A., Vidoni, R. 2012. Trajectory planning in robotics, Math. Comput. Sci., Vol. 6, no. 3, pp. 269-279.

[15] Bonin-Font, F., Ortiz, A., Oliver, G., 2008. Visual navigation for mobile robots: A survey, J. Intell. Robot. Syst., Vol. 53, no. 3, pp. 263-296.

[16] Pan, M., Liu, Y., Cao, J., Li, Y., Li, C., Chen, C.-H. 2020. Visual Recognition Based on Deep Learning for Navigation Mark Classification, IEEE Access, Vol. 8, pp. 32767-32775, doi: 10.1109/ACCESS.2020.2973856.

[17] Ayyalasomayajula R., Arun, A., Wu, C., Sharma, S., Sethi, A.R., Vasisht, D., Bharadiaet D. 2020. Deep Learning Based Wireless Localization for Indoor Navigation," 26th Annual International Conference on Mobile Computing and Networking, doi: 10.1145/3372224.3380894

[18] Ocaña, M., Bergasa, L. M., Sotelo, M., Flores, R. 2005. Indoor robot navigation using a POMDP based on WiFi and ultrasound observations, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2592-2597.

[19] Parker-Holder, J., Rajan, R., Song, X., Biedenkapp, A., Miao, Y., Eimer, T., Zhang, B., Nguyen B., et al., 2022. Automated Reinforcement Learning (AutoRL): A Survey and Open Problems, J. Artif. Intell. Res., Vol. 74, pp. 517-568, doi: 10.1613/jair.1.13596.

[20] LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep Learning, Nature, Vol. 521, pp. 436-444, doi: 10.1038/nature14539.

[21] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep Residual Learning for Image Recognition, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, doi: 10.1109/CVPR.2016.90.

[22] Krizhevsky, A., Sutskever, I., Hinton, G.E. 2012. ImageNet Classification with Deep Convolutional Neural Networks, in Advances in Neural Information Processing Systems, pp 1097-1105.

[23] Szegedy C., et al., 2015. Going deeper with convolutions, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1-9, 2015. doi: 10.1109/CVPR.2015.7298594.

[24] R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. 2018. Neural ordinary differential equations. Advances in neural information processing systems, 31.

[25] Ainsworth, S., Lowrey, K., Thickstun, J., Harchaoui, Z., Srinivasa, S. 2021. Faster Policy Learning with Continuous-Time Gradients. In Learning for Dynamics and Control, pp. 1054-1067, PMLR

[26] Yıldız, Ç., Heinonen, M., Lähdesmäki, H. 2021. Continuous-time model-based reinforcement learning. In International Conference on Machine Learning, pp. 12009-1201, PMLR.

[27] Meleshkova, Z., Ivanov, S.E., Ivanova, L. 2021. Application of Neural ODE with embedded hybrid method for robotic manipulator control, Procedia Comput. Sci., Vol. 193, pp. 314-324, doi: 10.1016/j.procs.2021.10.032.

[28] Du, J., Futoma, J., Doshi-Velez, F. 2020. Model-based Reinforcement Learning for Semi-Markov Decision Processes with Neural ODEs. Advances in Neural Information Processing Systems, 33, 19805-19816.

[29] Zhao, L., Miao, K., Gatsis, K., Papachristodoulou, A. 2024. NLBAC: A Neural Ordinary Differential Equations-based Framework for Stable and Safe Reinforcement Learning. arXiv http://arxiv.org/abs/2401.13148, (Accessed: Apr. 29, 2024).

[30] Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., Levine, S. 2018. Composable Deep Reinforcement Learning for Robotic Manipulation, IEEE International Conference on Robotics and Automation (ICRA), pp. 6244-6251. doi: 10.1109/ICRA.2018.8460756.

[31] Soviany, P., Ionescu, R.T., Rota, P., Sebe, N. 2021. Curriculum Learning: A Survey. International Journal of Computer Vision, Vol. 130:6,pp. 1526 - 1565, doi:10.1007/s11263-022-01611-x

[32] Zhao, R., Sun, X., Tresp, V., 2019. Maximum Entropy-Regularized Multi-Goal Reinforcement Learning, International Conference on Machine Learning, K. Chaudhuri and R. Salakhutdinov, Eds., in Proceedings of Machine Learning Research, Vol. 97. PMLR, pp. 7553-7562.

[33] Ziebart, B. D. 2010. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. Carnegie Mellon University.

[34] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. 2017. Proximal Policy Optimization Algorithms, arXiv preprint arXiv:1707.06347.

[35] Xia, F., Zamir, A.R., He, Z., Sax, A., Malik, J., Savarese, S., 2018. Gibson Env: Real-World Perception for Embodied Agents, IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9068-9079. doi: 10.1109/CVPR.2018.00945.

[36] Coumans, E., Bai, Y. 2016. Pybullet, a python module for physics simulation for games, robotics and machine learning.