actuators

MDPI

Article

Path Planning for Robots Combined with Zero-Shot and Hierarchical Reinforcement Learning in Novel Environments

Liwei Mei [1,†] and Pengjie Xu [2,*,†]

[1] School of Information Science and Technology, East China University of Science and Technology, Shanghai 200237, China; 21013181@mail.ecust.edu.cn
[2] School of Mechanical Engineering, Shanghai Jiaotong University, Shanghai 200240, China
* Correspondence: xupengjie194105@sjtu.edu.cn
† These authors contributed equally to this work.

Abstract: Path planning for robots based on reinforcement learning encounters challenges in integrating semantic information about environments into the training process. In unseen or complex environmental information, agents often perform sub-optimally and require more training time. In response to these challenges, this manuscript pioneers a framework integrating zero-shot learning combined with hierarchical reinforcement learning to enhance agent decision-making in complex environments. Zero-shot learning enables agents to infer correct actions for previously unseen objects or situations based on learned semantic associations. Subsequently, the path planning component utilizes hierarchical reinforcement learning with adaptive replay buffer, directed by the insights gained from zero-shot learning, to make decisions effectively. Two parts are trained separately, so zero-shot learning is available in different and unseen environments. Through simulation experiments, we compare the traditional hierarchical reinforcement learning method with the proposed method. The results prove that this structure can make full use of environmental information to generalize across unseen environments and plan collision-free paths.

Keywords: path planning; zero-shot learning; hierarchical reinforcement learning; adaptive agents

Citation: Mei, L.; Xu, P. Path Planning for Robots Combined with Zero-Shot and Hierarchical Reinforcement Learning in Novel Environments. *Actuators* **2024**, *13*, 458. https://doi.org/10.3390/act13110458

Academic Editor: Zhuming Bi

Received: 8 October 2024
Revised: 7 November 2024
Accepted: 13 November 2024
Published: 15 November 2024

1. Introduction

Robotic path planning involves algorithms that instruct a robot to take reasonable steps to approach a user-specified location in an unknown environment. This task is essential for ensuring the effective navigation and decision-making capabilities of robots and unmanned vehicles in the evolving landscape of autonomous systems. The ability to navigate through complex and dynamic environments is crucial for applications ranging from autonomous driving to warehouse automation and search-and-rescue operations.

Moreover, traditional path planning algorithms, including A* and Dijkstra's algorithm [1,2], although effective in static environments, often struggle in some environments where full information is not available. For instance, in warehouse automation, the A* algorithm can be used to find the shortest path for a robot to navigate from a storage location to a packing area. However, if an unexpected obstacle, such as a misplaced package, blocks the planned route, the algorithm must re-plan the path. This re-planning can introduce significant delays, reducing operational efficiency. Additionally, A* can sometimes find paths that are theoretically optimal but practically infeasible due to narrow corridors or tight turns that the robot cannot navigate, necessitating manual intervention. Apart from this, heuristic methods like Particle Swarm Optimization (PSO) [3,4], though showing the capacity to dynamically adjust to environmental changes without the need for complete map information, also present challenges. For example, in agricultural robotics, PSO has been applied to navigate drones for crop monitoring. However, the algorithm can suffer from premature convergence, where particles get trapped in local optima, leading to suboptimal paths. These algorithms generally require a pre-built map which is based on

simultaneous localization and mapping (SLAM) [5], and do not adapt well to changes, necessitating frequent re-planning, which can be computationally expensive and inefficient. The accuracy of SLAM heavily depends on the precision of the sensors used, which can be compromised by factors such as sensor noise, range limitations, and resolution constraints. Additionally, the high cost and instability of the equipment under various weather conditions are significant constraints that limit the practical deployment of these traditional methods in real-world scenarios [6].

Because of the mechanism of reinforcement learning (RL), a robot is able to make decisions by collecting interactions with the environment and then choosing an optimal policy by maximizing the collected rewards [7]. An existing problem in the reinforcement learning training process is how to provide appropriate rewards. Numerous methods can achieve this functionality, but they come with some compromises in some other performance. For instance, the approach of utilizing frequency and rewards to form gradient information guidance [8] may cause reinforcement learning to fall into local optima and decrease its exploration capabilities. To solve the mentioned challenges, this manuscript explores a method combining zero-shot learning and hierarchical reinforcement learning.

2. Related Work

The related work primarily encompasses three research directions: zero-shot learning, artificial potential field, and hierarchical reinforcement learning. In this section, the main content of these research directions, as well as some frontier studies in perception and decision-making within these areas, are introduced.

2.1. Zero-Shot Generalization

Zero-shot learning (ZSL) [9], a class of algorithms capable of performing well on new tasks without additional data collection, holds immense potential across various domains. In this manuscript, two concepts are mainly discussed, ZSL in image processing and zero-shot generalization (ZSG) in path planning.

ZSL in image processing: It enables models to recognize and classify previously unseen objects by leveraging semantic information encoded in class attribute descriptions and embeddings [10]. By bridging the gap through semantic knowledge transfer, zero-shot learning generalizes beyond the confines of training data to novel instances. The Contrastive Language-Image Pre-Trained (CLIP) model is a masterpiece which has been implemented in multiple areas. For instance, in feature extraction, Zhang et al. demonstrated that ZSL could be enhanced by training multi-modality embeddings using a deep learning model [11]. In environment detection, it combines 3D point clouds and RGB images to enhance the capability to identify 3D objects. Zhu et al. put forth PointCLIP V2 [12]. It projects 3D cloud points to a 2D plane, obtains the embedding by Bidirectional Encoder Representations from Transformers, and then compares the similarity between embedding and projecting the image and semantic information to form an identifying result. Additionally, in the multimedia area, Song et al. introduced MeshCLIP [13], which processes cross-modal information for 3D mesh data using zero-shot learning, thereby improving reconstruction quality. In this manuscript, CLIP is used to generate action guidance according to visual information.

Zero-shot generalization: It is an essential metric that evaluates the capability of a model to perform effectively on new, unseen tasks without additional training [14]. As shown in Figure 1, the data distribution of this problem can be divided into three situations: the training set and test set are the same, they follow the same distribution, or they follow different distributions. This capability is particularly crucial in novel environments where pre-defined training datasets cannot cover all possible scenarios. ZSG enables models to leverage prior knowledge and apply it to novel situations by relying on semantic information and transfer learning mechanisms [15]. In the context of RL, zero-shot generalization is formalized as the ability of a policy, trained in one set of contexts, to perform well in entirely new, unseen contexts. This involves specifying which subset

of ZSG problems is being addressed, as it encompasses a range of scenarios rather than a single specific problem. For instance, a robot trained to navigate in a particular type of environment should be able to adapt its navigation strategies to different environments it has never encountered before, relying on the transfer of learned semantic relationships and policy structures.

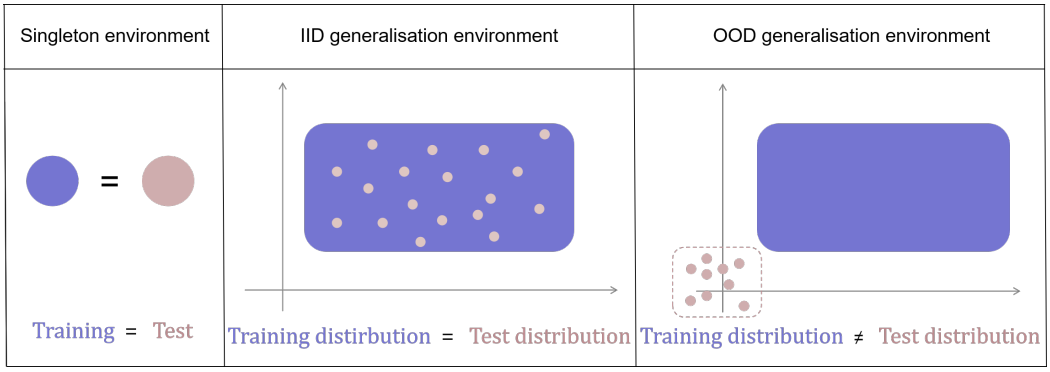


Figure 1. Zero-shot generalization data distribution. The singleton environment means that the training set is identical to the test set.The independent and identical (IID) generalization environment means that the two sets follow the same distribution while the out-of-distribution (OOD) generalization environment means that they follow different distributions.

2.2. Artificial Potential Field

The artificial potential field (APF) method is a widely used approach in robotic path planning [16]. It conceptualizes the environment as a potential field, where the robot is treated as a particle moving under the influence of this field. The method employs two primary components: an attractive potential field that pulls the robot towards the goal and a repulsive potential field that pushes it away from obstacles.

In robotic path planning, the APF method is advantageous due to its simplicity and real-time applicability, especially in partially known environments. Since this method can lead to local minima where the net force on the robot is zero, preventing further movement, we make some modifications to suit the hierarchical reinforcement learning process.

2.3. Hierarchical Reinforcement Learning

HRL draws inspiration from the way humans approach complex problems—by decomposing them into smaller, more manageable tasks [9]. HRL operates on multiple levels of decision-making, where high-level policies guide the overall direction toward the goal, and low-level policies handle the specific action. Such a structure mirrors the cognitive process of setting a general objective and executing detailed actions to achieve it [17]. This approach improves learning efficiency and enhances adaptability in dynamic environments. Recent advancements have seen HRL successfully applied across various domains, including robotic navigation, gaming, and autonomous driving [18,19]. These studies highlight the effectiveness of HRL in managing high-dimensional state spaces and executing complex sequential decisions, showcasing its potential in facilitating long-term planning and precision control. Christen et al. put forth an explicit task decomposition method [20], which can conduct a zero-shot of the planning layer across different low-level agents without retraining. Chen et al. proposed a soft actor–critic structure with a prioritized experience replay [21], solving the problem of low sample utilization. Ye et al. proposed a hierarchical policy learning with intrinsic–extrinsic modeling [22]. As analyzed above, information about environments, especially unseen ones, can impact the efficiency of an agent. Additionally, the hierarchical structure has shown great potential for the improvement of efficiency and generalization ability.

In the robotic visual servoing path planning task, the perception of vision sensors can be incomplete due to issues such as occlusion. To address this problem, we employ a hierarchical reinforcement learning framework. The high-level decision-making module is guided by visual information, and the visual guidance is constrained within a specific area. This approach ensures that the path planning process can effectively handle challenges arising from incomplete perception, improving the robustness and accuracy of the task.

To enhance agent performances, we explored a path planning method that integrates ZSL with HRL. The contributions of our work are described as follows:

(1) The proposed method first fuses ZSL into the reinforcement learning process for robot path planning. After integrating the unseen semantic information, the agent becomes more intelligent regarding path selection and training cost control.
(2) A reasonable fusion architecture is proposed specifically. ZSL is utilized on a high level to infer correct macro-actions for previously unseen objects or situations based on learned semantic relationships. Then, the HRL follows cues offered by zero-shot learning to make decisions effectively in specific path selections.
(3) Detailed performance analysis is provided for the proposed combined learning framework. The simulation results corroborate the proposed method's capability of leveraging visual cues for decisions and modifying agents' actions.

3. Proposed Method

The system architecture of the proposed method is shown in Figure 2. It is a two-stage learning process with a combination of zero-shot learning and hierarchical reinforcement learning. Part A and Part B belong to the zero-shot learning methods. Part C is the main structure of hierarchical reinforcement learning. The designed visual-act model utilizes a zero-shot learning image encoder to transfer images into embedding. Then, image embeddings and correspondent labels are sent to a multi-head self-attention network. This work is shown in Figure 3. The outputs of this stage are action instructions according to input images. It identifies the focusing parts of image embedding related to the exact actions in each direction. The latter stage is a hierarchical reinforcement learning process. It utilizes a high-level and a low-level policy to form paths. The high-level policy determines the plausible area while the low-level policy provides exact action. In the meantime, images captured by a visual sensor are sent to the pre-trained model. Then, the pre-trained model provides action instructions to correct decisions made by the reinforcement learning part.

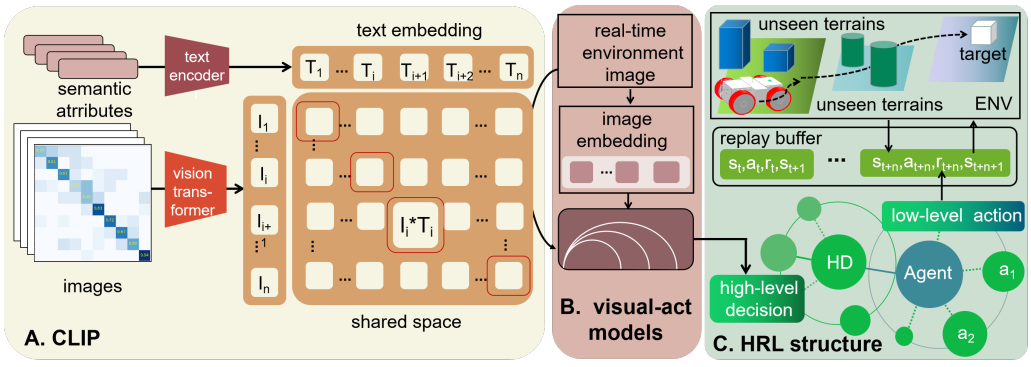


Figure 2. Main structure of the proposed method.

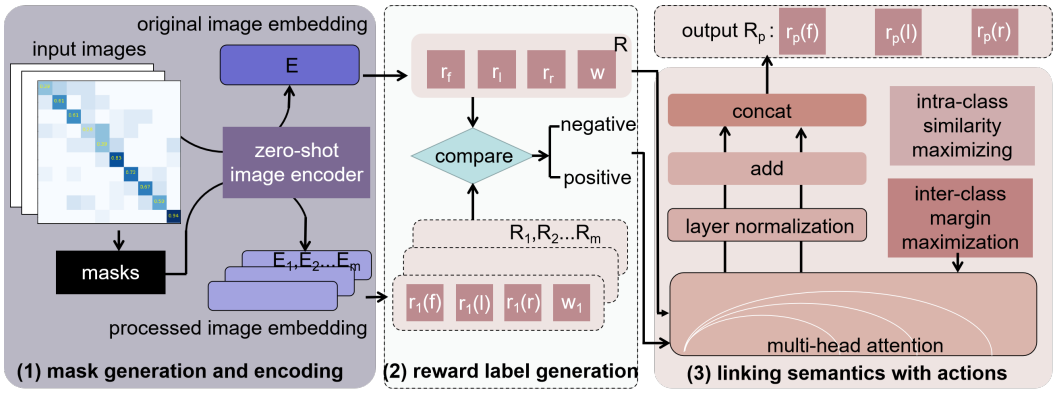


Figure 3. Pre-trained visual-act decision model

3.1. Contrastive Language-Image Pre-Trained Model.

The CLIP model leverages the relationships between images and their corresponding text descriptions, specifically their vector representations, to achieve zero-shot learning capabilities. In this section, we describe the core components and mechanisms of the CLIP model and how it contributes to our proposed method. As in Figure 1, the method can be described as follows:

Images I**:** training set images with the corresponding text descriptions.

Semantic attributes T**:** different text descriptions, including those that relate to the image contents and those that do not.

Vision transformer $f(I)$ [23]: it interprets image information into vectors v_I.

Text encoder $g(T)$ [24]: it transforms semantic attributes into text embedding v_T.

Shared space: a mapping relationship formed by contrastive learning.

$$v_I = f(I), \quad v_T = g(T) \tag{1}$$

The similarity between an image and a text description is then quantified by a similarity metric S, typically the cosine similarity, which is computed as:

$$S(v_I, v_T) = \frac{v_I \cdot v_T}{\|v_I\| \|v_T\|} \tag{2}$$

Elements in $S(v_I, v_T)$ are separated into corresponding pairs and non-corresponding ones. During training, the model is optimized to maximize the similarity between corresponding images and texts while minimizing the similarity between non-corresponding pairs [25]. Such a semantic-based zero-shot learning approach allows the model to acquire the correct embedding of images when faced with previously unseen objects by leveraging the relationship between semantic information and image features.

3.2. Visual-Act Model

Because this encoder is trained using a contrastive learning strategy, the model can only focus on the relative relationships between images and semantic information and cannot attend to absolute information, such as the spatial coordinates of objects. This limitation is addressed by the training approach of the proposed method, which utilizes masks to specify exact objects related to decision-making. The visual-act model utilizes the zero-shot image encoder to transform images into embedding vectors. Then, as shown in Algorithm 1, the model figures out the relationship between visual information and action selection, which can extract semantic information related to decision-making, thus optimizing the path planning process. During training, the model receives raw images as input and uses the mask-processed images as an amplifier for the attention matrix error. It trains a neural network to predict the attention matrix and the resulting actions. During the inference task, the trained neural network directly processes raw images to output action predictions.

Algorithm 1 Optimized model for high-level macro-action decision making in action selection

Input: Image dataset $\mathcal{D}$ with annotations, Set of masks $\{M_i\}$, Number of epochs N
Output: Optimized model for action decision $R_p[r_p(f), r_p(l), r_p(r)]$, processed image embeddings I_1, I_2, I_3
Initialize the zero-shot image encoder Z
Initialize action-specific networks for each action O_a
Initialize parameters for contrastive loss
Define total loss function L_{total} with weighting factor λ
for each epoch **do**
    for each $(\text{Image}, R, \{R_i\})$ in dataset $\mathcal{D}$ **do**
        $E \leftarrow Z(\text{Image})$
        for each mask M_i in Set of masks **do**
            $E_i \leftarrow Z(\text{Image} \oplus M_i)$
        end for
        Initialize task-specific losses L_a for this batch
        for each action a in {forward, left, right} **do**
            $R_p, I_1, I_2, I_3 \leftarrow$ action-specific network(E, a)
        end for
        $L_a \leftarrow \|R_p - R\|_2$
        $L_c \leftarrow$ contrastive loss$(E, I_1, I_2, I_3, \{E_i\}, \{R_i\})$
        $L_{total} \leftarrow \sum L_a + \lambda \times L_c$
        Backpropagate L_{total} and update parameters
    end for
    Optionally evaluate the model on the validation set
end for
Save the optimized model parameters

(1) **Mask generation and encoding**

For each input image I, a series of object masks [26] are applied to block different regions. Specifically, some masks are applied to block certain vacant areas of the image, like grassland or remote objects, which is irrelevant to current action instructions. Some masks conform to the shapes of certain objects in the image, which are used to occlude specific objects in the image. The final number of masks is capped at sixteen by disregarding far-away and trivial objects. The original image and mask-processed images are represented as E and E_i. By putting masks on images, the zero-shot image encoder can enlarge the features of the mask-applied part, and it can be reflected in the output embeddings in a way that allows attention mechanism and contrast learning.

For each action $a \in \{a_\text{f}, a_\text{l}, a_\text{r}\}$, we calculate a reward label r_a considering several factors in this direction: the distance to obstacles, the dynamic/static nature of these obstacles, and the distance to the goal. The backward action is excluded from our model due to its focus on forward-moving scenarios. Mathematically, the reward label for action a can be expressed as:

$$r_a = f(d_\text{obs}, d_\text{g}, \text{type}_\text{obs}) \tag{3}$$

where d_obs denotes the distance to the nearest obstacle, d_g represents the distance to the goal, and type_obs indicates whether the obstacle is static or dynamic. The function f computes the reward label, encapsulating the trade-offs between navigating safely around obstacles and efficiently moving toward the goal.

The original image E and processed images E_i are labelled as R and R_i. For R_i, we focus on whether the masked object is in the corresponding direction, and then denote this with a difference as a mark for contrastive loss. A weight λ is stitched to R_i to decrease the loss value if the masked object belongs to the background.

(2) **Linking semantics of images with actions**

Based on the differences between the image labels R_i after mask processing and the original label R at each label element, the images that have undergone mask processing are labeled as positive samples x^+ for the positions with differences, while those without any changes are labeled as negative samples x^-. For example, the original label R is $[2, -1, -1]$, and the masked label R_i is $[2, -1, -2]$. Then, the masked image is a positive sample for the third element regression task.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

where Q, K, and V represent queries, keys, and values, respectively.

Then, these samples are sent to a masked multi-head attention network [27], which aims to forge a shared feature representation pivotal for ensuing action decisions. For every pre-defined action, action-specific networks O_a process this shared representation, engendering action determinations. The self-attention mechanism allows the model to focus on specific elements in semantic embeddings. This dynamic is orchestrated by incorporating each action's loss L_a into a cumulative loss function, where the contrastive loss L_c plays a crucial role in guiding the model towards discerning distinct actions. The contrastive loss L_c serves as an loss amplifier to help the model better focus on certain semantic attributes in the embedding that have crucial influences on the formation of the action instructions.

$$L_{\text{total}} = \sum_a L_a + \lambda \times L_{\text{c}} \tag{5}$$

$$L_{\text{c}} = -\log \frac{\exp(\text{sim}(x, x^+)/\tau)}{\exp(\text{sim}(x, x^+)/\tau) + \sum_{x^-} \exp(\text{sim}(x, x^-)/\tau)} \tag{6}$$

where $\text{sim}(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$ denotes the cosine similarity between two vectors x and y. x is the anchor sample, x^+ is a positive sample similar to the anchor, and x^- represents a negative sample dissimilar to the anchor. τ is a temperature scaling parameter that controls the separation between positive and negative pairs.

The model outputs serve as supplemental guidance for high-level policy in hierarchical reinforcement learning and provide instruction for high-level decisions.

3.3. Hierarchical Policy

In this section, we utilize HRL to complete path planning tasks under the guidance of leveraging visual cues for the action selection model. A two-level policy layer is conducted to increase efficiency without sacrificing exploring capability. The process is shown in Algorithm 2.

(1) **Environment representation**

The environments are represented in two forms, e.g., high-level grid form and low-level ones. The high-level network is designed to achieve global decision-making based on the above work, while the lower-level is used for better local planning.

High-level grid map: It selects the next macro-action, directing the agent towards a specific region on the grid. The grid map is divided into larger blocks, and each block is treated as a high-level state. The high-level policy determines the sequence of blocks to be explored based on the current state and the goal location. The settings of the high-level grid help agents avoid areas where visual information has not been fully detected and accelerate the training process in the task of robotic path planning.

Low-level grid map: Within each selected high-level block, the low-level strategy navigates through the individual grid cells. The policy network produces action instructions for robotics when visual and state information is obtained in grid cells.

$$s_{low} \in \{\text{cell}_{1,1}, \text{cell}_{1,2}, \ldots, \text{cell}_{m,m}\} \tag{7}$$

(2) **RL structure**

We model after the design of DQN and incorporate this structure into the proposed framework: a policy network and a target network. As shown in Figure 4, the policy network is updated continuously based on the Bellman equation and the target network and generates policies for explored areas $\pi_{explored}$. APF is applied to generate policies for unexplored areas $\pi_{unexplored}$. $\pi_{explored}$ and $\pi_{unexplored}$ are combined to form policy π_{global} for the target net. The target network's parameters are updated less frequently, specifically, every 100 episodes, by comparing the global strategy with the most recent policy network.

$$Q(s, a; \theta) \rightarrow \text{policy network} \tag{8}$$

$$Q'(s, a; \theta^-) \rightarrow \text{target network} \tag{9}$$

Algorithm 2 Hierarchical path planning

Initialize high-level and low-level policies
Initialize policy network $Q(s, a; \theta)$ and target network $Q'(s, a; \theta^-)$
Initialize experience replay buffer D with capacity N
for each episode **do**
  for each step in episode **do**
    Observe visual input I_t and current state s_t
    Generate low-level action $a_t = \pi(s_{low} \mid I_t)$
    if high-level block is unexplored **then**
      Generate direction $\vec{F} = -\nabla U$ using APF as Equations (14)–(16)
      Decompose direction into actions for unexplored blocks
    else
      Follow policy $\pi_{explored}$ for explored blocks
    end if
    Execute action a_t, observe reward r_t and next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer D
  end for
  Synthesize global strategy $\pi_{global} = \pi_{explored} \cup \pi_{unexplored}$
  Sample random batch from replay buffer D
  Compute target Q-value: $y = r_t + \gamma \max_{a'} Q'(s_{t+1}, a'; \theta^-)$
  Perform gradient descent step on $(y - Q(s_t, a_t; \theta))^2$
  Update policy network $Q(s, a; \theta)$
  if episode // 100 == 0 **then**
    Compare π_{global} with historical strategies
    Update target network $Q'(s, a; \theta^-) = Q(s, a; \theta_{recent})$
  end if
  Update policy network $Q(s, a; \theta)$ based on target network
  Dynamically adjust B and b based on R_{bar}
end for

The action selection is based on the ε-greedy policy, balancing exploration and exploitation. The Q-value updates follow the standard Bellman equation:

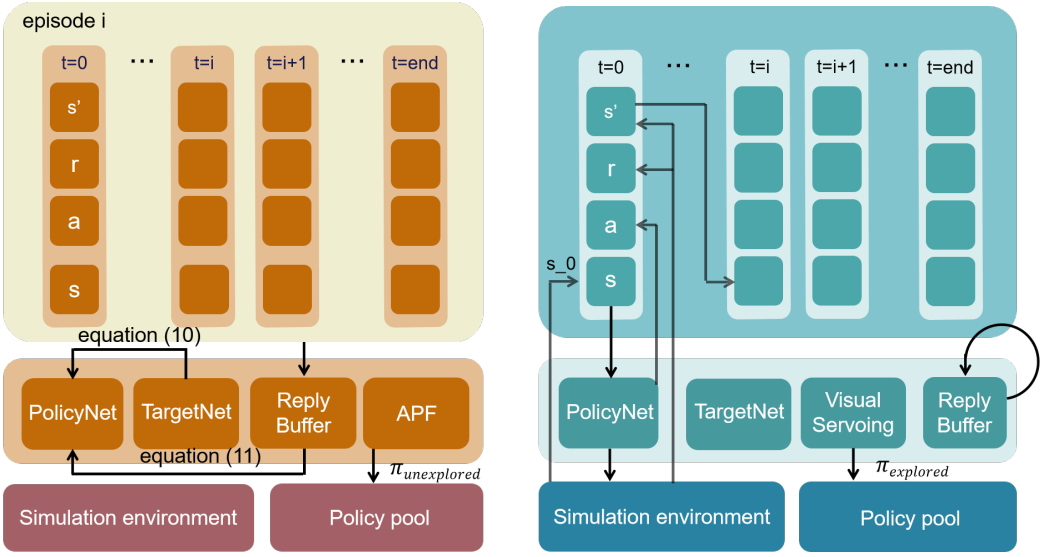
$$Q(s_t, a_t) = r_t + \gamma \max_{a'} Q'(s_{t+1}, a'; \theta^-) \tag{10}$$

where r_t is the reward at time t, γ is the discount factor, and θ^- are the parameters of the target network.

Additionally, the policy network parameters are updated based on the target network to ensure stability:

$$\theta \leftarrow \theta + \alpha(y - Q(s, a; \theta))\nabla_\theta Q(s, a; \theta) \tag{11}$$

where $y = r_t + \gamma \max_{a'} Q'(s_{t+1}, a'; \theta^-)$.



(**a**) Data flow for every time step (**b**) Data flow for every episode

Figure 4. Data flow of the hierarchical policy.

(3) **Base policy and reward settings**

Visual servoing: For areas already explored, we employ the visual-act model to map input images to action directives. The network processes the visual input and outputs an action for the agent. During every step, these action instructions are added to a policy pool, which serves as the optimal policy to determine the parameters of the target net.

$$a_{(x,y)} = \pi(s_{low} \mid I_{(x,y)}) \tag{12}$$

$$\pi_{explored} \leftarrow \bigcup_{(x,y)\in \mathbf{E}}^{n} a_{(x,y)} \tag{13}$$

where $a_{(x,y)}$ is the action at the coordination (x, y), $I_{(x,y)}$ is the visual input, and π is the policy function parameterized by the pre-trained visual-act decision model. $\pi_{explored}$ is the policy developed by the pre-trained model for the explored area.

Artificial potential field for unexplored regions: For high-level blocks that have not been explored, we utilize the artificial potential field method to generate navigation strategies. The APF method uses an attractive potential to pull the agent towards the goal and a repulsive potential to avoid obstacles. The direction generated by the APF is then decomposed to produce specific actions. The potential field is represented as U, which can be decomposed into two parts, U_{att} and U_{rep}. The formula is represented as follows:

$$U_{att} = \frac{1}{2} k_{att} d^2 \tag{14}$$

$$U_{rep} = \begin{cases} \frac{1}{2} k_{rep} \left(\frac{1}{d_o} - \frac{1}{d_{o0}} \right)^2 & \text{if } d_o \leq d_{o0} \\ 0 & \text{if } d_o > d_{o0} \end{cases} \tag{15}$$

where k_{att} and k_{rep} are constants, d is the distance to the goal, d_o is the distance to the obstacle, and d_{o0} is the influence distance of the obstacle.

The combined potential field U determines the direction $\vec{F}$ for navigation:

$$\pi_{unexplored} \leftarrow \vec{F} = -\nabla U \tag{16}$$

This direction is then translated into specific actions within the unexplored high-level blocks as the $\pi_{unexplored}$. The whole process is conducted after every episode is completed.

Global strategy: It is synthesized by combining the policies derived from both the explored and unexplored regions. The synthesized strategy is then compared with the historical strategy obtained from past episodes. The target network parameters are updated based on the comparison to ensure the new strategy optimizes over the past strategies.

$$\pi_{global} = \pi_{explored} \cup \pi_{unexplored} \tag{17}$$

Reward settings: The reward functions are listed below. s_{next} represents the next state of the agent. s_{target} represents the target state. $R_{collision}$ and R_{safe} represent the reward value that the agent receives in collision condition and safe condition, respectively.

(1) Target distance

$$R_1(s,a) = -\|s_{next} - s_{target}\| \tag{18}$$

(2) Obstacle avoidance

$$R_2(s,a) = \begin{cases} R_{collision} & \text{if collision} \\ R_{safe} & \text{otherwise} \end{cases} \tag{19}$$

(3) Whether the agent reaches the target

$$R_3(s,a) = \begin{cases} R_{target} & \text{if } s_{next} = s_{target} \\ R_{step} & \text{otherwise} \end{cases} \tag{20}$$

(4) **Adaptive experience replay management**

The management of the experience replay buffer [28] is adaptively adjusted based on the cumulative average reward, which serves as an indicator of the learning progress and the efficiency of the current policy. The cumulative average reward, R_{avg}, is computed as follows:

$$R_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} R_i \tag{21}$$

where R_i is the immediate reward received after the i-th action, R_g represents rewards of achieving the goal, R_o represents rewards of encountering obstacles, and N is the total number of actions taken up to the current point in time. If the agent achieves the goal or encounters obstacles, R_g or R_o is added to R_{avg} and this episode is instantly suspended.

Based on R_{avg}, we adjust the buffer size, B_{size}, and the batch size, b_{size}, of the experience replay buffer to enhance the learning efficiency. The adjustments are made according to the following rules:

$$B_{\text{size}} = B_{\min} + (B_{\max} - B_{\min}) \cdot \min\left(\frac{R_{\text{avg}}}{R_{\text{target}}}, 1\right) \tag{22}$$

$$b_{\text{size}} = b_{\min} + (b_{\max} - b_{\min}) \cdot \min\left(\frac{R_{\text{avg}}}{R_{\text{target}}}, 1\right) \tag{23}$$

where $B_{\min}$ and $B_{\max}$ represent the minimum and maximum buffer sizes, respectively; similarly, $b_{\min}$ and $b_{\max}$ denote the minimum and maximum batch sizes. R_{target} is a target average reward that indicates an optimal learning performance. These formulas ensure that as the average reward approaches the target, both the buffer size and the batch size increase, allowing the model to learn from a larger set of experiences. Conversely, if the performance drops, the model focuses on a smaller, potentially more relevant set of experiences to adjust its policy more rapidly.

4. Simulations

In this section, we build up joint simulation environments based on the V-rep platform and Pycharm. We conduct simulations on the environments and validate the following conclusions:

(1) The leveraging of visual cues for the action selection model does focus on certain objects in images to form correspondent advice for action selection.
(2) To assess the model's capability to generate across environments, we conduct path planning simulation in unseen environments, which proves that the proposed method can reduce training time and increase exploring efficiency.
(3) The proposed method performs well when combined with reinforcement learning.

4.1. Simulation Setting

The simulation settings, including equipment and environment settings, are listed below.

Equipment: The simulations are conducted on Windows 11 operating system powered by 13th Gen Intel(R) Core(TM) i9-13900HX, with training executed on an NVIDIA GeForce 4090 GPU. The simulation experiment is conducted on V-rep (version 4.6.0 Edu.) and Pycharm (version 2023.2.5, professional edition). **Simulation environment:** Table 1 presents the parameters of the model car and obstacles in the virtual simulation platform.

Table 1. Parameters of the robot.

Parameter	Value
Wheel radius (m)	0.5
Tread (m)	1
On-board camera resolution	512×512
Robot cabinet width (m)	0.9
Robot cabinet length (m)	2
Radius of obstacle (m)	1

Network parameters of the proposed method: Table 2 presents the parameters of the pre-trained visual information alignment model. The high-level grid size is calculated by the viewing range of the vision sensor.

Table 2. Hyper-parameters of the HRL model.

Parameter	Value
Learning rate (high-level) α_1	0.002
Learning rate (low-level) α_2	0.002
ϵ_{decay}	0.995
ϵ_{min}	0.1
Γ	0.99
High-level grid size	4×4
Low-level grid size	32×32
Goal reward	1000
Max step number	500
Obstacle reward	-200
Buffer size ascending rate	0.9
Buffer size ascending threshold	1.2
Buffer size descending rate	0.9
Buffer size descending threshold	1.0
Initialized buffer size	10,000
Initialized batch size	128
Buffer size (min–max)	1000–20,000
Batch size (min–max)	16–256

4.2. Simulation Process

The comparative and validation simulation experiments follow this routine:

Step 1: Communication between the Python terminal and V-rep is established, the simulation step size is set to 0.05 s, and the initial state of the robot is calibrated. The maximum range of the vision sensor is 5 m, and each grid covers 0.625×0.625 square meters, so

we would include 8 × 8 grids in a high-level grid. This grid is then used in the high-level grid map. The parameters of the grid map are initialized for path planning.

Step 2: The designed control algorithm runs on the Python side, generating the trajectory. This is then converted into control rates for the four wheels at each moment. The control laws are sent to the model car's rotational joints to execute the movement according to the control instructions.

Step 3: In the environment, the state of key robot nodes is recorded and this information is sent to the Python terminal through the application programming interface.

Step 4: Steps 2 and 3 are repeated, iterating through the set simulation time and frequency until the entire simulation is completed.

4.3. Assessment of the Visual-Act Model

In this part, we train our model on several diverse environments and test its performance on a separate set of unseen environments that follow different distributions from the training set. The training environments utilize cityscapes [29]. Labels are modified to fulfill the model input requirements. It includes various layouts, obstacle types, and goal locations in real road scenes.

As shown in Figure 5, observations of the loss curve reveal a contracting trend during the training process, indicating a consistent enhancement in model accuracy. The training process converges around 200 epochs because the zero-shot image encoder is perfectly pre-trained, which can well reflect the semantic and spatial information of objects in the image in the coding and accelerate the convergence speed in the proposed method's training process.

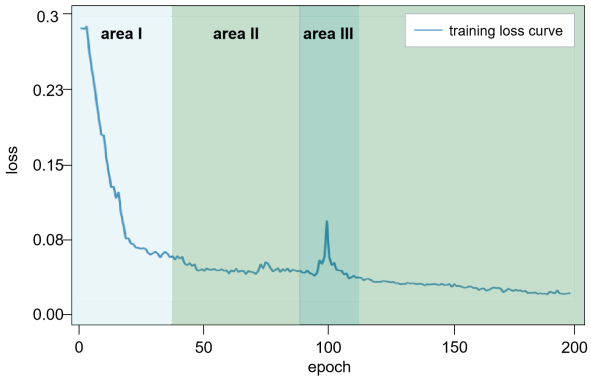
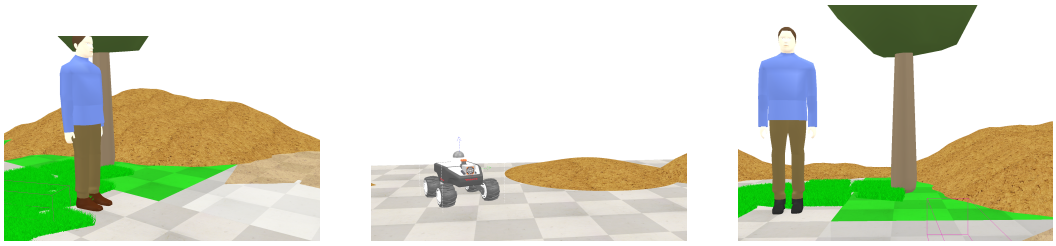


Figure 5. Loss curve of the pre-trained model

The loss of the model training drops rapidly in Area I, but it picks up in Area III. We speculate that when setting the loss function, we treat the similarity using the sigmoid function to guarantee the artificial number in logarithm loss calculation to be positive, and the gradient is amplified during the loss backward process; thus, the fluctuation occurs.

As shown in Figure 6, the test environments are designed to differ significantly from the training ones regarding layout complexity and obstacle placements, ensuring a rigorous assessment of the model's generalization ability. The virtual environment is primarily designed to imitate a wilderness setting, and its significant differences from the urban road image data in the training set and the inexperience of the pre-trained model in such a scenario further contribute to the rigorousness of the assessment.

We conduct a comparative simulation experiment to confirm the focused area of the visual-act model. I_0 represents the embeddings produced by the original image with CLIP. I_1, I_2, and I_3 represent the image embeddings processed by the visual-act model. The sim(I_0) value represents the similarity between the image embeddings and semantic vector of certain objects.



(**a**) (**b**) (**c**)

Figure 6. Test set samples. (**a**–**c**) The subfigures are test images samples.

In Table 3, data labeled in bold have the highest similarity to the corresponding object after processing with the attention mechanism in all directions. For example, as shown in Figure 7, the tree is in front of the robot, and the similarity comparison represents that the corresponding processed embeddings raise focus on this obstacle from 0.2607 to 0.4121 while decreasing focus in irrelevant directions. This shows that the pre-trained model focuses on certain obstacles related to the corresponding action while disregarding the unrelated ones.

Table 3. Similarity comparison.

Object	Sim(I_0)	Sim(I_1)	Sim(I_2)	Sim(I_3)
Tree	0.2607	**0.4121**	0.1298	0.0736
People	0.2473	**0.5223**	0.2954	0.1371
Floor	0.1849	0.0718	0.1035	0.1302
Sand	0.3001	0.1727	0.0931	0.1127
Grass	0.2131	0.1981	0.1703	0.0689

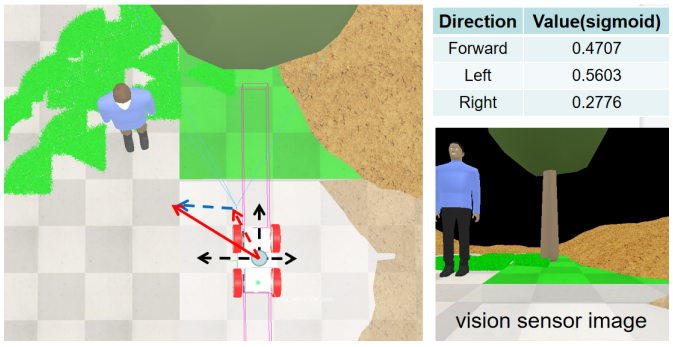


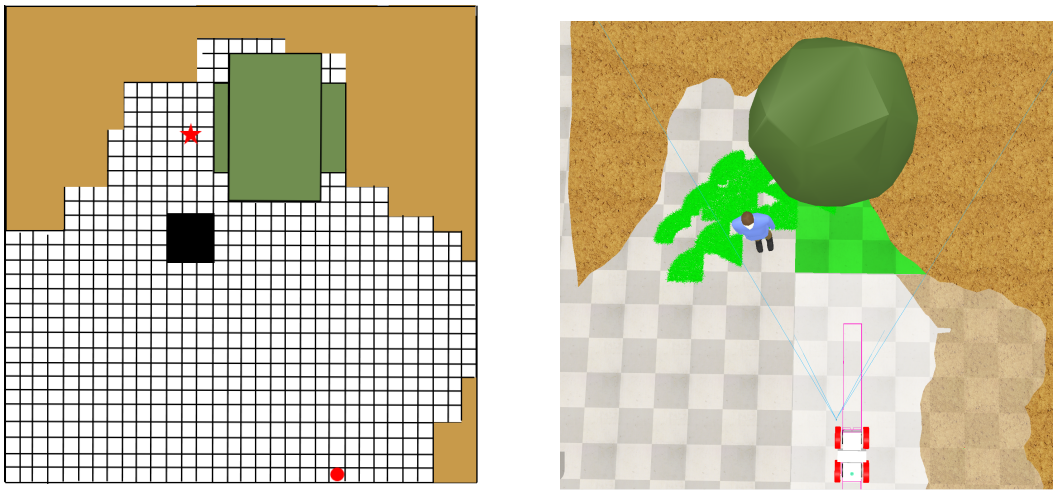
Figure 7. High-level policy combined with action instruction.

4.4. Performance Assessment Under Singleton Environment

In this part of the study, we conduct simulation experiments in a singleton environment, where the training and test datasets are identical. This experimental setup is designed to assess the robustness and effectiveness of our proposed reinforcement learning method for path planning, by directly comparing it to other conventional RL methods under controlled conditions.

(1) **Settings**

The singleton environment simplifies the problem space by using the same scenarios in both the training and test phases, allowing us to focus on the learning and optimization capabilities of the algorithms without the variability introduced by unseen test conditions. The experimental area measures 25 m $\times$ 25 m, and high-level grids are planned based on camera parameters to provide better guidance for macro-action. The representation of one train environment sample is shown in Figure 8.



(**a**) Grid form (**b**) Simulation environment sample

Figure 8. Environment representation; the simulation environments can be represented by grid forms. The red star represents the target, the green circle represents the starting point, the yellow blocks represent sand piles, the green blocks represent trees, and the black blocks represent people.

(2) **Metrics**

The metrics are listed below:

(1) Efficiency—the step costs compared to the shortest possible paths.

(2) Final convergence value—the value of cumulative average reward when the reward curve converges.

(3) Convergence degree—the fluctuation trend during the training process, the number of epochs each algorithm took to converge, and the final convergence value.

(3) **Outcomes**

Figure 9 illustrates the cumulative average reward curve when incorporating the visual-act network into hierarchical reinforcement learning. We can corroborate that the proposed method has a good convergence trend and a low degree of fluctuation. Compared to the traditional HRL method, the convergence episodes decrease from 780 to 480 and the final convergence value increases from 24 to 297, which shows great progress in performance.

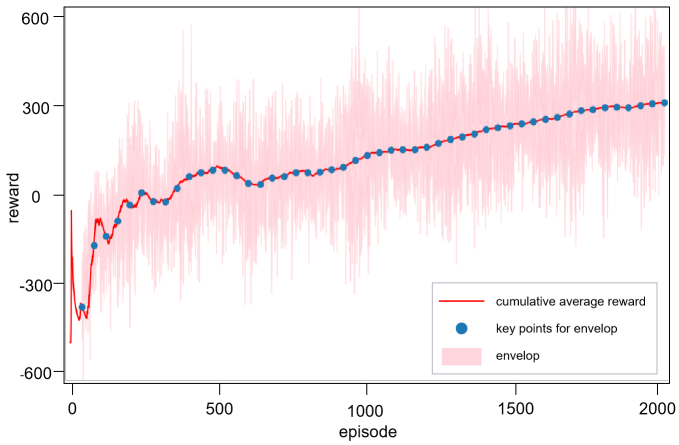
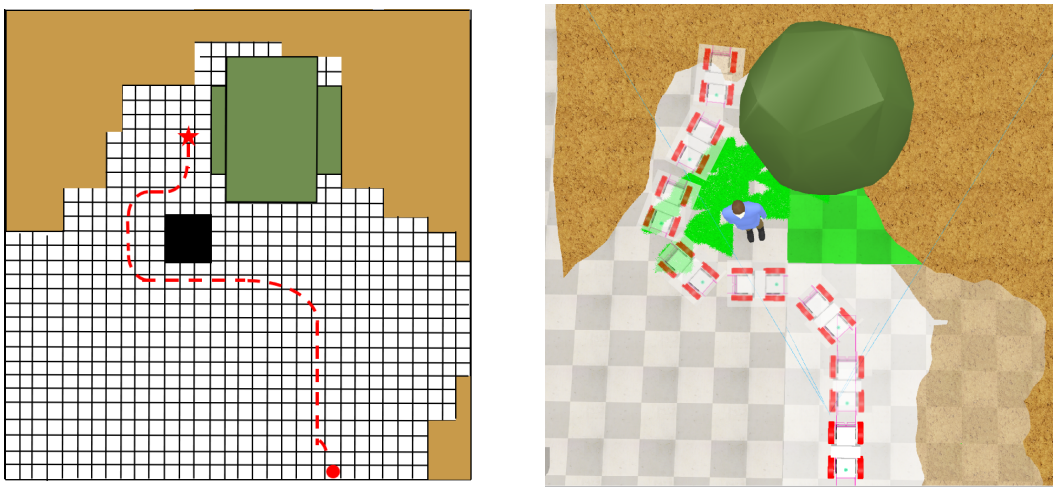


Figure 9. Cumulative average reward curve.

Figure 10 presents gradients formed in the high-level grid and path in novel environments. We can see that the proposed method adequately provides efficient high-level gradient information for path planning tasks, resulting in an efficient and collision-free final path planning outcome.



(**a**) Grid guidance (**b**) Path in simulation

Figure 10. Path planning with train environment.

4.5. Performance Assessment Under Test Environments

In this part, we conduct the generalization simulation under several environments modeled after the trained ones.

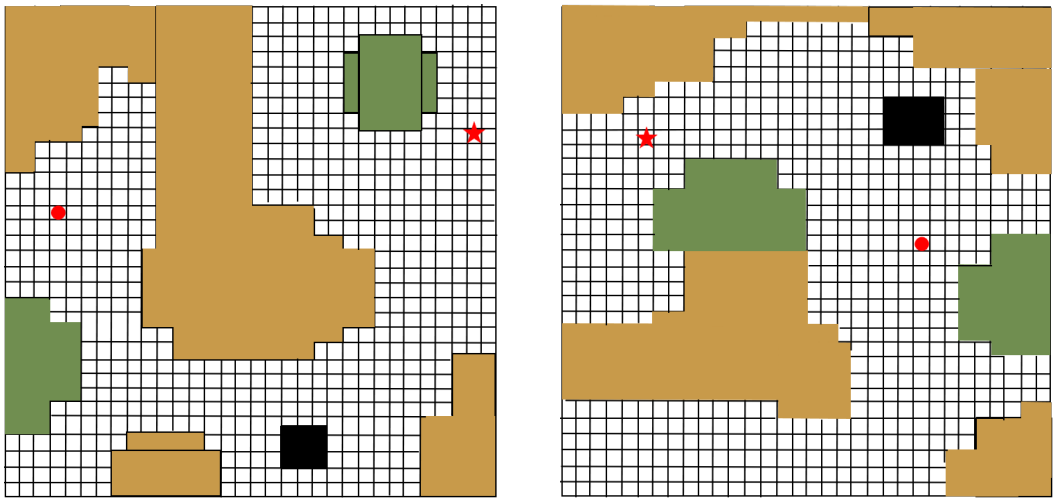
(1) **Settings**

The test environments differ from the training ones. As shown in Figure 11, they follow a different distribution. The tests are conducted without extra training within unseen test environments. The distinction between the training environments and test environments is listed below:

(1) The starting and target points follow different distributions from the training environments. Specifically, as shown in Figure 11a,b, they all lie in different directions.

(2) The patterns of the available paths are different from the training environments.

(3) The distributions of obstacles are different.



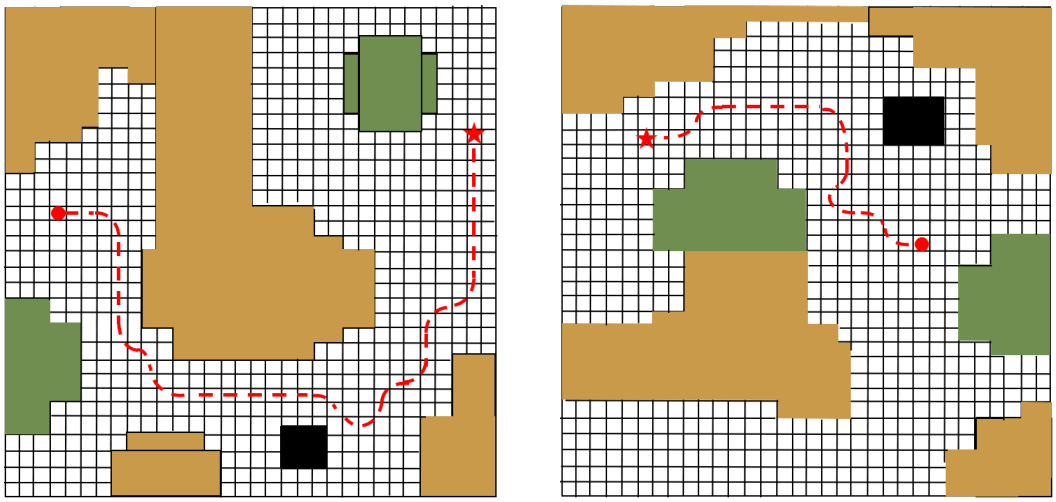
(**a**) (**b**)

Figure 11. Environment representation samples in grid form.

(2) **Metrics**

In this experiment, we emphasize the guiding role of ZSL and visual action at a high level. Therefore, the analysis is conducted on a grid map. We select success rate and collision rate as metrics. The specific results are shown in Table 4 and Figure 12.

(3) **Outcome**



(**a**) (**b**)

Figure 12. Path in novel environments.

Table 4. Similarity comparison.

Environment	S [1]	C [2]	S' [3]	C' [4]
(a)	9.25%	8.53%	21.62%	2.64%
(b)	3.31%	14.72%	11.91%	3.12%
(c)	16.28%	7.34%	31.88%	0.65%

[1] S represents the average success rate under several novel environments for the traditional HRL method. [2] C represents the average collision rate under several novel environments for the traditional HRL method. [3] S' represents the average success rate under several novel environments for the proposed method. [4] C' represents the average collision rate under several novel environments for the proposed method.

In novel environment tests, as shown in Table 4, the collision rate and success rate indicate that the proposed method has significantly reduced the collision rate, effectively lowering the collision rate of the intelligent agent in the path planning task. Additionally, a certain degree of enhancement is observed in the success rate. The paths planned under the corresponding samples are shown in Figure 12. Among environments (a), (b), and (c), the performance under environment (b) is not good enough because the target area cannot be seen most of the time during the process, and thus the designed visual-act cannot perform well in success rate, but the collision rate remains at a low level, indicating that the proposed method has excellent performance in obstacle avoidance.

5. Conclusions

In this manuscript, we propose a novel method that combines ZSL with HRL for robot path planning. Based on ZSL, the model manages to integrate visual information in the high-level policy of hierarchical reinforcement learning process to achieve better macro-actions. Furthermore, ZSL helps the robotic path planning process to alleviate the dependency on depth information. For HRL agents, we detail how the HRL architecture is implemented and propose an adaptive experience pool strategy to balance the exploration capability and convergence dynamically.

We conduct several simulation experiments to validate the proposed method. In cross-environment generalization assessment, we test the proposed method with unseen environments, demonstrating that the proposed method still has good convergence, thus exhibiting strong generalizing performance. The comparative analysis proves that the visual-act model focuses on areas related to decision-making, highlighting the model's capability to leverage visual cues for action selection. Finally, we evaluate the model's performance in path planning when encountering unseen environments. The proposed method improves

the robustness and efficiency of robotic path planning and helps to decrease the collision rate during the training process.

Author Contributions: Conceptualization, L.M. and P.X.; methodology, L.M. and P.X.; software, L.M.; validation, L.M. and P.X.; data curation L.M. and P.X.; writing—original draft preparation, L.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as potential conflicts of interest.

References

1. Duchoň, F. Path planning with modified a star algorithm for a mobile robot. *Procedia Eng.* **2014**, *96*, 59–69. [CrossRef]
2. Luo, M.; Hou, X.; Yang, J. Surface optimal path planning using an extended Dijkstra algorithm. *IEEE Access* **2020**, *8*, 147827–147838. [CrossRef]
3. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
4. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the IEEE International Conference on Evolutionary Computation, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73.
5. Grisetti, G.; Kümmerle, R.; Burgard, W. A tutorial on graph-based SLAM. *IEEE Intell. Transp. Syst. Mag.* **2010**, *2*, 31–43. [CrossRef]
6. Zang, S.; Ding, M.; Smith, D. The impact of adverse weather conditions on autonomous vehicles: How rain, snow, fog, and hail affect the performance of a self-driving car. *IEEE Veh. Technol. Mag.* **2019**, *14*, 103–111. [CrossRef]
7. Wang, Y.; Zou, S. Policy gradient method for robust reinforcement learning. In Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; pp. 23484–23526.
8. Zhang, J.; Koppel, A.; Bedi, A. Variational policy gradient method for reinforcement learning with general utilities. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 4572–4583.
9. Lampert, C.; Nickisch, H.; Harmeling, S. Learning to detect unseen object classes by between-class attribute transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 951–958.
10. Wang, W.; Zheng, W.; Han, Y. A survey of zero-shot learning: Settings, methods, and applications. *Acm Trans. Intell. Syst. Technol.* **2019**, *10*, 1–37. [CrossRef]
11. Zhang, L.; Xiang, T.; Gong, S. Learning a Deep embedding model for zero-Shot learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3010–3019.
12. Zhu, X.; Zhang, R.; He, B. Pointclip v2: Prompting clip and gpt for powerful 3d open-world learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 2639–2650.
13. Song, Y.; Liang, N.; Guo, Q. MeshCLIP: Efficient cross-modal information processing for 3D mesh data in zero/few-shot learning. *Inf. Process. Manag.* **2023**, *60*, 103497. [CrossRef]
14. Kirk, R. A survey of zero-shot generalisation in deep reinforcement learning. *J. Artif. Intell. Res.* **2023**, *76*, 201–264. [CrossRef]
15. Ball, L.; Parker, H. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In Proceedings of the International Conference on Machine Learning, Graz, Austria, 18–24 July 2021; pp. 619–629.
16. Chen, Y. UAV path planning using artificial potential field method updated by optimal control theory. *Int. J. Syst. Sci.* **2016**, *47*, 1407–1420. [CrossRef]
17. Botvinick, M.M. Hierarchical reinforcement learning and decision making. *Curr. Opin. Neurobiol.* **2012**, *22*, 956–962. [CrossRef] [PubMed]
18. Eppe, M.; Kerzel, C. Intelligent problem-solving as integrated hierarchical reinforcement learning. *Nat. Mach. Intell.* **2022**, *4*, 11–20. [CrossRef]
19. Wöhlke, J.; Schmitt, F.; Van, H. Hierarchies of planning and reinforcement learning for robot navigation. In Proceedings of the IEEE International Conference on Robotics and Automation, Xi'an, China, 30 May–6 June 2021; pp. 10682–10688.
20. Duan, J.; Eben, S.; Guan, Y. Hierarchical reinforcement learning for self-driving decision-making without reliance on labeled driving data. *Intell. Transp. Syst.* **2020**, *14*, 297–305. [CrossRef]
21. Christen, S.; Jendele, L.; Aksan, E. Learning functionally decomposed hierarchies for continuous control tasks with path planning. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3623–3630. [CrossRef]
22. Ye, X.; Yang, Y. Efficient robotic object search via hiem: Hierarchical policy learning with intrinsic-extrinsic modeling. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4425–4432. [CrossRef]
23. Lin, B.; Zhu, Y.; Chen, Z. Adapt: Vision-language navigation with modality-aligned action prompts. In Proceedings of the the IEEE Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 15396–15406.
24. Kotei, E.; Thirunavukarasu, R. A systematic review of transformer-based pre-trained language models through self-supervised learning. *Information* **2023**, *14*, 187. [CrossRef]

25. Jaiswal, A.; Babu, A.; Zadeh, M. A survey on contrastive self-supervised learning. *Technologies* **2020**, *9*, 2. [CrossRef]
26. Huo, X.; Karimi, H.; Zhao, X. Adaptive-critic design for decentralized event-triggered control of constrained nonlinear interconnected systems within an identifier-critic framework. *IEEE Trans. Cybern.* **2021**, *52*, 7478–7491. [CrossRef] [PubMed]
27. Vaswani, A.; Shazeer, N.; Parmar, N. Attention is all you need. In Proceedings of the Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
28. Tiwari, R.; Killamsetty, K.; Iyer, R. Gcr: Gradient coreset based replay buffer selection for continual learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 99–108.
29. Marius, C.; Mohamed, O.; Sebastian, R. The cityscapes dataset for semantic urban scene understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 3213–3223.