Applied Intelligence (2023) 53:6195–6213
https://doi.org/10.1007/s10489-022-03503-6

Towards better generalization in quadrotor landing using deep reinforcement learning

Jiawei Wang[1,2] · Teng Wang[3] · Zichen He[1] · Wenzhe Cai[3] · Changyin Sun[2,3]

Accepted: 13 March 2022 / Published online: 7 July 2022

Abstract
In recent years, the autonomous landing of unmanned aerial vehicles (UAVs) has attracted extensive attention due to the widespread applications of UAVs. With the rapid improvements in machine learning and artificial intelligence, recent research has begun to explore deep reinforcement learning (DRL) to learn the landing policy directly from raw observation data. However, current DRL-based solutions tend to suffer poor generalization to unseen environments. To deal with this issue, we formulate the landing problem as a two-stage DRL problem and bootstrap the DRL procedures by augmenting regular DRL loss with an auxiliary localization task. The auxiliary localization task provides dense supervision signals that aid in landing-relevant representation learning. In particular, two marker localization approaches are delicately designed based on deep classification and regression models, and differences between the two configurations are explored, aiming to answer the fundamental question of how to exploit localization better for representation learning. Furthermore, we propose a novel and flexible sampling strategy called Dynamic Partitioned Experience Replay to stabilize and accelerate the training procedure. Experimental results show that the auxiliary localization tasks combined with the improved sampling strategy aid the trained model to generalize in unseen environments. In addition, the trained model can be seamlessly transferred to the real-world quadrotors and has achieved outstanding landing performances.

Keywords Deep reinforcement learning · Auxiliary task · Partitioned experience · Quadrotor landing · Generalization

1 Introduction

Unmanned aerial vehicles (UAVs), which can fly without the presence of onboard pilots, are finding applications in a broadening number of fields, especially in those occupations that require to execute boring, dirty, or dangerous missions, such as search and rescue [1], remote sensing [2], precision agriculture [3], surveillance [4], delivery of goods [5], cable detection [6] and small object detection [7]. Landing a UAV is acknowledged as the last and most critical stage for navigation in these applications. This work targets at the applications of goods delivery and focuses on the specific task of landing a quadrotor on a desired ground marker with a high level of accuracy.

To solve the landing task, previous works rely on numerous airborne sensors [8–10] or ground support equipment [11–13] to estimate the UAV attitude. However, airborne sensors are usually expensive or power-consuming, and ground support equipment is not always available. Recently, various vision-based UAV landing methods have been developed since UAV platforms usually include a monocular camera as standard equipment. For example, [14] designed a neural network to locate the marker in the captured images. [15] designed a novel form of fiducial marker and used it to estimate the relative location of the UAV camera with regard to the marker. However, these methods are susceptible to illumination and UAV attitude changes, or they are only suitable for specially designed landing markers.

Inspired by the notable breakthroughs of deep reinforcement learning (DRL) in the fields of games [16, 17] and robotics control [18, 19], researchers started to employ DRL to solve the problem of autonomous quadrotor landing. Learning to land directly in three-dimensional

✉ Changyin Sun
cysun@seu.edu.cn

1 College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China

2 Peng Cheng Laboratory, Shenzhen, 518055, China

3 Department of Automation, Southeast University, Nanjing 210018, China

environments by DRL is not trivial and faces the challenge of sparse and delayed rewards. To address these challenges, Polvara et al. [20] decomposed the quadrotor landing problem into marker alignment and vertical descent sub-tasks, with each solved by one specifically designed sequential deep Q-network (SDQN), and introduced a new partitioned experience replay to alleviate the issue of sparse and delayed rewards. Xu et al. [21] employed the deep Q-network (DQN) [17] in dueling architecture to further improve the stability of the training process. These DRL-based solutions can learn to land a quadrotor on a ground marker using raw RGB images from the down-looking camera, and achieve impressive performance comparable to human pilots. However, the learned landing policies can not be well generalized to unseen environments with diverse background appearances.

The poor generalization capability of previous DQN-based methods is because end-to-end training does not guarantee the model learns the accurate marker features. These methods neglected the valid information in the environment during training, such as the relative position between the marker and the quadrotor. The auxiliary tasks provide additional training signals and improve the data efficiency. In addition, the auxiliary tasks have been proven to facilitate domain adaptation by reducing the domain gap between the seen and unseen environments [22, 23]. Motivated by this, we designed auxiliary tasks to leverage the valid information in the environment and help the model to learn better features.

To achieve better generalization, we proposed a novel marker localization-assisted deep reinforcement learning model for each of the marker alignment and vertical landing tasks. This model jointly learns the goal-driven DRL problem with an auxiliary task to improve the learned landing policy's generalization capability. The marker localization shares the common feature representations with the landing policy, so that it could help the DRL agents to build useful features for localizing the landing marker with high accuracy, bootstrapping the model's generalization capability in unseen environments. In particular, we delicately formulate the marker localization as a classification and a regression task to facilitate representation learning and explore the difference between two configurations through feature visualization. Unfortunately, the sparse and delayed rewards make the vertical landing task in the form of Blind Cliffwalk[24]. To address this problem, we propose a novel and flexible sampling method called dynamic partitioned experience replay, which samples transition experiences from different partitions with varying sampling ratios. This sampling method is proved to make learning from experiences more efficient while maintaining low computational complexity compared with the previous sampling methods. The models are trained in the Gazebo simulator which includes a failure-and-recovery mechanism, and the models can be seamlessly transferred to a real quadrotor. Our contributions are summarized as follows.

1. We propose a novel marker localization-assisted deep reinforcement learning solution to learn the landing policy from raw observations. The auxiliary localization is designed to facilitate learning better feature representations for marker localization. Experimental results show that our proposed solution dramatically improves the learned policy's landing performance and generalization capability.
2. We propose a novel and flexible sampling method called dynamic partitioned experience replay to stabilize and accelerate the training procedure. Compared with the previous sampling methods, our method makes learning from experiences more efficient while remaining low computational complexity.
3. The model trained from simulation can be seamlessly transferred to a real-world quadrotor, and real-world experiments have shown the outstanding performance of our method.

The remainder of this article is organized as follows. Section 2 introduces the basics of reinforcement learning. Section 3 reviews the related work of autonomous landing. Section 4 describes the proposed method in detail. Section 5 presents the experiment. Section 6 discusses the results and limitations, which is followed by conclusions in Section 7.

2 Background

In the reinforcement learning (RL) problems, an agent learns the optimal policy that maximizes the expected rewards by interacting with the environment. The environment can be modeled as a Markov Decision Process (MDP) defined by a tuple $<\mathcal{S}, \mathcal{A}, P, R, \gamma>$. $\mathcal{S}$ denotes the state space and represents the set of environment states; while $\mathcal{A}$ denotes the action space and represents the set of available actions; $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is the state transition probability function; $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, which serves as the feedback signal to the agent from the environment; $\gamma \in (0, 1)$ is the discount factor, which quantifies the present value of future rewards [25]. The agent is in a state $s_t \in \mathcal{S}$ at each time step t, it performs an action $a_t \in \mathcal{A}$ according to the policy π. Based on the reward function $R(s_t, a_t)$, the agent receives a reward r_t from the environment and enters the next state s_{t+1} according to the state transition function $P(s_{t+1}|s_t, a_t)$. The agent repeats the above process until the end. The final goal is to find an optimal policy π^* that maximizes the expected discounted cumulative reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$.

With the development of deep learning, researchers use deep neural networks as function approximators to solve problems with high-dimensional inputs. This approach is known as deep reinforcement learning. DQN is a remarkably stable and flexible DRL algorithm that has shown success in tackling a wide range of problems with discrete action space [26–28]. DQN learns the optimal action-value function $Q_{\pi}^{*}(s_t, a_t)$ by utilizing deep convolutional neural network (CNN). DQN constructs a CNN architecture, consisting of a feature embedding network and a policy network parameterized by θ and α, respectively, to approximate the optimal action-value function $Q^{*}(s, a)$. DQN trains the whole CNN model by minimizing the loss function:

$$L_p(\theta, \alpha) = \mathbb{E}_{(s,a,r,s') \sim U(D)}\Big[\big(Y - Q(s, a; \theta, \alpha)\big)^2\Big], \tag{1}$$

where D is the experience replay buffer and contains a collection of experience tuples, usually denoted as (s, a, r, s') with states, actions, rewards, and successor states; $Q(s, a; \theta, \alpha)$ is the Q-value estimated by the current CNN approximator given the state s and action a; Y is the target for estimated Q-value. Standard DQN performs action selection and evaluation by using the same Q values, resulting in overoptimistic value estimations. To avoid this, Double Q-learning (DDQN) [29] decouples the action selection from action evaluation. This is achieved by learning two sets of network weights (θ, α) and (θ^{-}, α^{-}), one for action evaluation and the other for action selection. The network weights (θ^{-}, α^{-}) for action selection are synchronized with (θ, α) every C time steps. Therefore, the target Y could be expressed as follows based on the Bellman equation:

$$Y = r + \gamma Q\big(s', \operatorname*{argmax}_{a'} Q(s', a'; \theta, \alpha); \theta^{-}, \alpha^{-}\big). \tag{2}$$

In order to improve the sample efficiency, Haarnoja et al. [30] provided the Soft Actor-Critic (SAC) algorithm based on the maximum entropy reinforcement learning framework. SAC-Discrete is the discrete version of the SAC algorithm, and it is well suited for the UAV landing task with discrete actions. SAC-Discrete maximizes both the expected return and action entropy to extensively explore the environment while maximizing the rewards. By adding a temperature parameter to the entropy term, SAC-Discrete balances exploration and exploitation. SAC-Discrete improves the sample efficiency and performs competitively with the state-of-the-art algorithms on the Atari games.

3 Related work

Autonomous landing has been studied for many years and many approaches have been proposed. We focus on the vision-based methods which have shown unprecedented results. The vision-based methods can be split into four categories: classic vision based methods, special marker based methods, Kalman filter based methods, and deep learning based methods.

Classic vision based methods The classic vision based methods use feature-based extraction or homography-based approaches to estimate the relative position between the UAV and the landing marker. Olivares-Méndez et al. [31] used the homography of a specific helipad to predict the UAV's 3D position. The visual input was then processed by a fuzzy controller to generate control signals. Keipour et al. [32] designed a landing marker detector to recognize and track the circular shape in the images. They employed a visual servoing controller to land the UAV on the moving vehicle. Saavedra-Ruiz et al. [33] used a feature-based detector to compute the homography matrix between the image and marker template. They employed Kalman filter estimation to track the marker and used a PID-based controller to complete the landing task. However, the classic vision based methods are susceptible to illumination and UAV attitude changes, making them lack robustness.

Special marker based methods The special marker based methods use artificial landmarks which contain rich spatial information. In [34], a special landing pad was constructed, and a vision-based detection algorithm was proposed for predicting the UAV's 3D position relative to the landing pad. In [35], a special color marker was created. Also, a method that combined the Levenberg-Marquardt with the vanishing point was employed to estimate the UAV's position and relative position quickly and accurately. In [36], the on-image search algorithm and the accurate landing algorithm were combined. Special ArUco-marker was employed to estimate the UAV camera's position and orientation relative to this special marker. In [37], multiple-scale Quick Response-codes were utilized to determine the relative distance and direction between the ground vehicle and UAV. A quadratic programming problem was formulated to design the velocity controller. In [15], based on the original ArUco marker, a novel form of fiducial marker called e-ArUco was designed to predict the UAV camera's relative position with regard to the marker. Nevertheless, these methods need special landing markers, limiting the model's capability to generalize to new environments with different markers.

Kalman filter based methods Kalman filter based methods employed Kalman filters to combine extra information from multiple sensor sources and estimate the UAV states more accurately. Vankadari et al. [38] employed the perspective-n-point algorithm to estimate the 6-DOF pose of the quadrotor utilizing marker images captured by both front and bottom cameras as input. They also used Extended Kalman Filter to fuse the data from downward-facing sonar and the inertial odometry to better estimate the pose of the UAV. Then, they adopted the least-squares policy iteration method to minimize the error between the UAV's current state and target state, thereby controlling the UAV to land on the marker. In [39], images captured by the UAV camera were utilized to generate the helipad's pixel coordinates. The measurements from the Inertial Navigation System and laser rangefinder were combined by the Kalman filter to generate the speed measurement value. Kim et al. [40] used YOLOv4-tiny network to detect the moving platform. To estimate the relative location of the landing site precisely, they developed the Extended Kalman Filter to fuse multiple sensors' measurements including encoder, inertial odometry, and ultra wide band. However, the Kalman filter based methods have the drawback of requiring the UAV to be equipped with numerous sensors. These sensors are usually expensive or power-consuming, resulting in higher costs and shorter endurance times.

Deep learning based methods Deep learning based methods use CNN-based architectures to detect and track the landing marker in images. In [14], the images captured by the UAV camera were processed by a simple neural network to locate the landing marker, and a PID controller was used to calculate the UAV's kinematic control vector. Zhang et al. [41] designed an adaptive learning-based CNN controller to locate the position of the autonomous surface vehicle and lead the UAV to land on the target region. They used GPS for horizontal tracking and UAV camera for vertical regulation. In [42], CNN was utilized to detect the landing marker and estimate the UAV state. A model predictive control based controller was designed to plan the entire landing trajectory and a cascade incremental nonlinear dynamic inversion control method was adopted to track the planned trajectory. However, the CNN-based methods require thousand of human-labeled images based on the task. In contrast, DRL does not require labeled data. DRL is more robust and it also generates control commands using only images as input. Xu et al. [21] employed the DQN in dueling architecture to achieve autonomous landing, however, the landing task was greatly simplified. They fixed the quadrotor's z-axis speed and placed the landing marker on the floor with no texturing. The aforementioned simplifications make this method impractical. Polvara et al. [20] used two DQN to achieve autonomous landing. The model trained in the simulator could be seamlessly transferred to the real quadrotor without extra tuning. However, this method has limited generalization capability when the training set is insufficient. To improve generalization, our proposed method adds an auxiliary task to recognize the landing marker, which aids the convolutional layer in extracting better marker features and accurately locating the marker under diverse background textures.

4 Proposed method

4.1 Problem definition

Solving the quadrotor landing problem directly in three-dimensional space faces the challenge of huge state space, we therefore decompose the autonomous quadrotor landing problem into two sub-tasks: marker alignment and vertical landing. In both phases, we consider discrete actions although the action space of UAVs is continuous in the real world. Introducing high-level discrete actions makes it much easier to transfer a policy learned in a simulation domain to a real-world domain or learned in one platform to a different platform. In the marker alignment phase, the quadrotor moves only on the x-y plane aiming to align itself with the ground marker, and enters the vertical landing phase by a "trigger" action. In the vertical landing phase, the quadrotor descends along the z-axis to approach the ground marker and shifts on the x-y plane at the same time to maintain accurate alignment with the landing marker.

In both marker alignment and vertical landing phases, the next state of the agent (quadrotor) is determined only by the current state, and therefore the two sub-tasks can be considered as Markov Decision Processes where the agent is interacting with environments with a monocular down-looking camera. During both phases, the agent interacts with the environment through a down-looking camera, and therefore the input to the deep model (i.e., state s_t) is the monocular RGB image. In the marker alignment phase, the agent moves only on the X-Y plane, and the feasible actions a_t available at each time step t are "go forward", "go backward", "turn left", "turn right", and "trigger". In particular, taking the "trigger" action will lead the agent to enter the vertical landing phase. The agent receives a positive reward of $+1.0$ if it performs "trigger" in the pre-defined target area, and receives a negative reward of -1.0 if it performs "trigger" outside the target area. Besides, the agent receives a negative reward of -0.01 if it takes other actions, which encourages the agent to accomplish the marker alignment as soon as possible. Thus, the reward function for the marker alignment task can be written as follows:

$$r_1(s,a) = \begin{cases} +1.0, & if\ a = trigger \\ & \quad \&\ agent\ in\ the\ target\ area \\ -1.0, & if\ a = trigger \\ & \quad \&\ agent\ out\ of\ the\ target\ area \\ -0.01, & if\ a \neq trigger \end{cases} . \tag{3}$$

In vertical landing phase, the agent moves in a limited 3D space, and the action space consists of "go forward", "go backward", "turn left", "turn right", "descend", and "landing". The agent receives a positive reward of +1.0 if it accomplishes landing in the pre-defined target area centered on the marker, and receives a negative reward of −1.0 if it misses the landing. To avoid collision with the ground, a negative reward of −1.0 is given if the altitude h of the agent is lower than a pre-defined threshold h_{min}. Besides, a negative reward of −0.01 is given if the agent takes other actions except "landing", which encourages the agent to finish landing as soon as possible. Thus, the reward function for the second phase can be expressed as follows:

$$r_2(s,a) = \begin{cases} +1.0, & if\ a = land\ \&\ h \geq h_{min} \\ & \quad \&\ agent\ in\ the\ target\ area \\ -1.0, & if\ a = land\ \&\ h \geq h_{min} \\ & \quad \&\ agent\ out\ of\ the\ target\ area \\ -0.01, & if\ a \neq land\ \&\ h \geq h_{min} \\ -1.0, & if\ h < h_{min} \end{cases} . \tag{4}$$

DQN is chosen as the basic framework of our proposed method, considering that DQN is easier to learn the optimal policy and it has been successfully applied to autonomous landings [20, 21]. The previous DQN-based landing methods have suffered from poor generalization due to the inability to accurately extract the marker features. The auxiliary tasks have been proven to facilitate domain adaptation and improve statistical efficiency. Inspired by the impact of auxiliary tasks, we bootstrap the RL procedure by augmenting the policy gradient loss with an auxiliary localization task. The relative position between the marker and the quadrotor is used since it is vitally important for the landing task and can be easily derived from the simulator. The auxiliary localization task infers the location of the marker relative to the quadrotor and provides dense supervision signals that aid in landing-relevant representation learning. Specifically, we attach the auxiliary deep localization network on the top of the feature embedding network, such that the deep features extracted by the feature embedding network are used as intermediate representations shared by the marker localization network and policy network. In this way, the localization loss could guide the feature embedding network to learn efficient feature representations for marker detection and recognition, thus facilitating the learning of a better landing policy. In summary, the whole training objective of our end-to-end model could be described as follows:

$$L(\theta,\alpha,\beta) = L_p(\theta,\alpha) + \omega L_a(\theta,\beta), \tag{5}$$

where $L_a(\theta,\beta)$ is the loss for marker localization; β are learnable parameters from our marker localization network; ω is a scalar that weights the marker localization loss against the policy gradient loss. The following subsection describes the design of auxiliary tasks in detail.

4.2 Auxiliary marker localization

In both phases, we explore two different variants for the marker localization task. We first consider a classification task, where the marker location at each time step is discretized into different cells. The other choice is to consider marker localization as a regression task, which develops a deep model to directly predict the relative position between the quadrotor and the landing marker.

The marker localization problem could be formulated as a classification task by discretizing each observation image into different cells and assigning a unique cell label to each image to identify which cell the marker is located. In the marker alignment phase, the flight altitude of the quadrotor remains unchanged and the size of the marker on the image is the same. Therefore, in this phase, we divide the captured image at each time step evenly into $M = n \times n$ cells, with each cell representing a unique class, as shown in Fig. 1 (a). Here, n is set to 23 since the width (height) of the image is about 23 times of the marker under the given flight altitude. However, in the vertical landing phase, the quadrotor is allowed to move in a limited 3D space. The varying flight altitudes of the quadrotor result in an obvious change in the marker scales on the observation image, i.e., the lower the flight altitude, the larger the marker. To handle the dramatic changes in marker scales, we delicately divide the current observation image into $M = 7$ cells, which are organized in a radial structure, as shown in Fig. 1 (b). Performing discretization in such a way allows the index of the cell where the marker is located not to change dramatically when flight altitude changes, and eliminates the impact of altitude changes on the classification results. For each phase, we build a simple localization network consisting of two fully connected (FC) layers. The last FC layer's output is sent into a M-way Softmax which generates a probability distribution over the M class labels. We attach the localization network on top of the last convolutional layer of the backbone feature embedding network, such

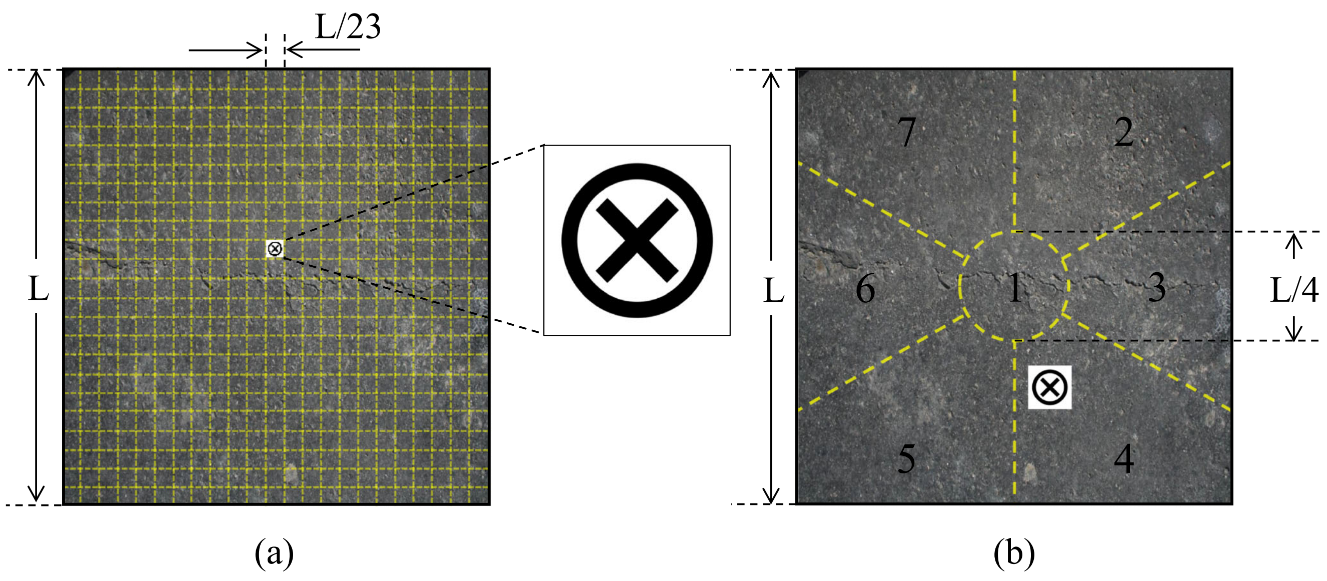


Fig. 1 Formulating the marker localization as a classification task. (a) In marker alignment phase, the captured image is evenly discretized into 23×23 cells; (b) In vertical landing phase, the captured image is delicately discretized into 7 cells, organized in radial structure

that the localization network and policy network share the intermediate feature representations from the backbone network. The overall structure of our CNN network with auxiliary localization task is shown in Fig. 2. Optimizing the localization network will guide the backbone network to learn a more efficient feature embedding for marker detection and recognition, thus facilitating the learning of a better landing policy. Here, we employ the following cross-entropy loss [43] to train our marker localization network parametrized by β:

$$L_a^c(\theta, \beta) = \sum_{i=1}^{M} \Big[-y_i \log(P(x_i; \theta, \beta)) - (1 - y_i) \log(1 - P(x_i; \theta, \beta)) \Big]. \quad (6)$$

To make a fair comparison between the classification and regression agents, we employ different classification networks for the two sub-tasks to guarantee that each classification agent has similar model complexity with the corresponding regression agent.

For each phase, we also regard the marker task as a regression problem, aiming to directly predict the relative position between the marker and the quadrotor on the X-Y plane. Since introducing the auxiliary localization task is to help the agent better capture the features of the marker and thus accurately locate the marker in its field of view, we perform predictions on the 2D plane. We build a simple CNN structure consisting of one Spatial Softmax layer followed by a FC layer to solve the regression problem. The FC layer's output is sent into a linear layer to produces a 2D vector, which encodes the relative coordinates of the marker to the quadrotor on the X-Y plane. Similarly, we attach the regression network on top of the backbone feature embedding network to facilitate the learning of a better landing policy. The overall structure of our CNN network with auxiliary localization regression is shown in Fig. 3. We employ the following L_2 loss to optimize the marker localization network:

$$L_a^r(\theta, \beta) = \sum_{i=1}^{b_s} \Big((\Delta x_i^* - \Delta x_i)^2 + (\Delta y_i^* - \Delta y_i)^2 \Big), \quad (7)$$

where b_s is the batchsize; Δx_i and Δy_i are predicted values of the relative coordinates in X and Y dimensions, respectively; Δx_i^* and Δy_i^* are ground-truth values for Δx_i and Δy_i, respectively.

4.3 Dynamic partitioned experience replay

In addition to inaccurate feature extraction, both the marker alignment and the vertical landing tasks have sparse and delayed rewards and are thus in the form of Blind Cliffwalk [24]. A positive reward is given to the quadrotor only when it successfully reaches the pre-defined target zone. Since the target zone is relatively tiny compared with the entire state space, it is extremely hard for the quadrotor to obtain a positive reward. This issue is severe for the vertical landing phase due to its 3D state space. As a result, there are few positive samples in the replay buffer D. Performing uniform sampling from such a replay buffer will miss the more useful positive-reward transitions with a high probability, making the policy learning unstable and less efficient. To alleviate this issue, Narasimhan et al. [44] considered such positive-reward transitions to have higher priority and used prioritized sampling to sample a fixed portion of transitions

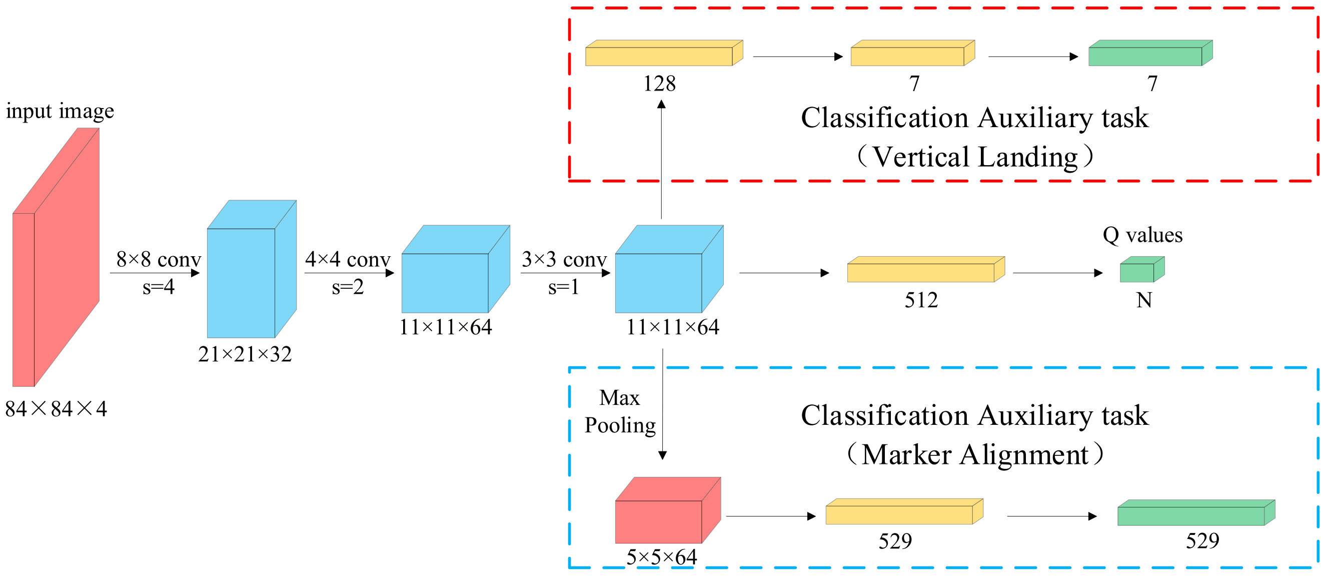


Fig. 2 The entire structure of the network with classification auxiliary task (AsDDQN-Cla). The input is four consecutive 84 × 84 grayscale images. The structures within the blue and red dotted line are used in the marker alignment phase and vertical landing phase respectively

from the higher priority pool at each learning step. Polvara et al. [20] employed three pools to keep track of positive, neutral and negative transitions separately. At each learning step, a fixed portion of transitions are sampled from each experience pool to ensure that there are enough more useful positive and negative transitions in each batch. Schaul et al. [24] measured the importance of the transitions by their temporal-difference (TD) error, and improved the learning efficiency by utilizing prioritized sampling to replay critical transitions more frequently. These replay methods have their limitations although they have achieved impressive performances. On one hand, these partition-based replay methods [20, 44] sample a fixed portion of transitions from each partition across the whole learning process. However, as learning proceeds, the importance of different partitions changes. Therefore, fixing the sampling probability will slow down the learning process. On the other hand, the TD-based replay method [24] updates the TD values of all experiences at each learning step, and thus suffers from high computational complexity.

To tackle the above challenge, we propose a new experience replay method called dynamic partitioned experience replay, as shown in Fig. 4. We follow [20] to employ three pools, denoted as $(D^+, D^\sim, D^-)$ to separately keep track of positive, neutral, and negative transitions. Different from [20], we dynamically adjust the sampling

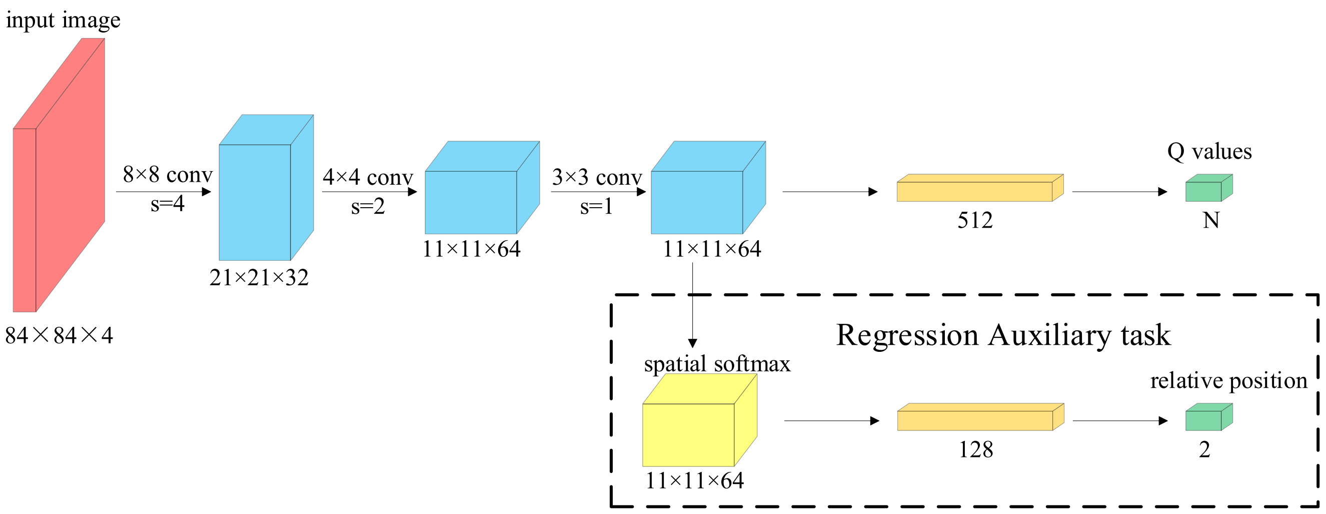


Fig. 3 The entire structure of the network with regression auxiliary task (AsDDQN-Reg) for both landing phases. Both the marker alignment and vertical landing share the same deep regression model

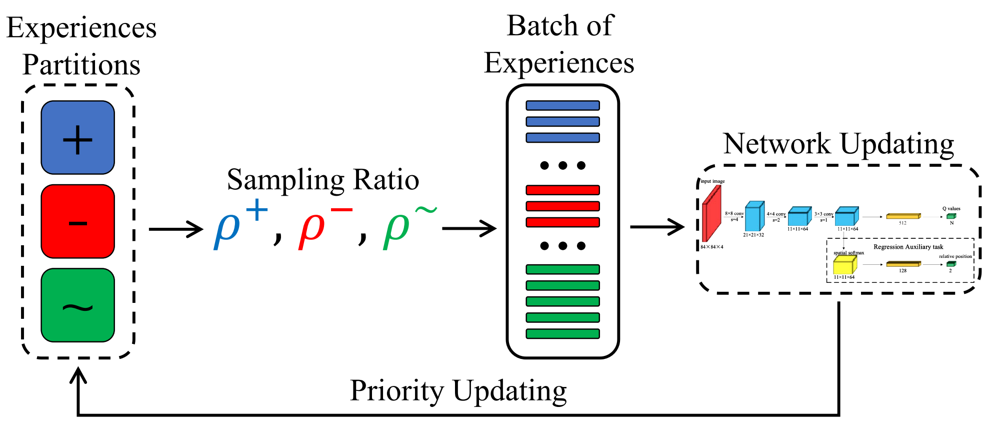


Fig. 4 The working principle of dynamic partitioned experience replay

ratio of each experience pool based on their average absolute TD errors during batch sampling. Our approach could be seen as a combination of partition based and TD based experience replay. Specifically, at each update step, we first compute the average absolute TD error of each partition and normalize the set of TD errors to make sure that their sum is equal to 1. Then, the transition selection is performed by a novel two-layer hierarchical sampling strategy. At a higher layer, we determine how many experiences are selected from each partition based on their normalized average TD error. Denote the normalized average absolute TD error of each partition as ρ^i, $i \in \{+, \sim, -\}$, a total of $b_s \times \rho^i$ transitions will be selected from each partition i when the batch size is set to b_s. At a lower layer, we measure the importance of the transitions in each partition by the magnitude of their TD error, and utilize prioritized sampling to replay important transitions more frequently. Once a set of b_s transitions $D = (e_1, ..., e_N)$ are sampled from the above three pools, we update the policy network using these transitions. After updating the policy network, we further update the TD errors of the b_s selected transitions instead of all samples, which helps to reduce computational complexity, and another new iteration starts.

5 Experiments

We first describe the implementation details in Section 5.1, then present the experimental results obtained with localization-assisted DQN in the marker alignment phase and vertical landing phase in Section 5.2, and finally provide an in-depth analysis of the experimental results through feature visualization in Section 5.3. We used the same quadrotor (Parrot BeBop 2) and simulation environment (i.e., Gazebo 7.7x, ROS Kinetic) in both training and testing. The control command sent to the quadrotor was transformed to a continuous speed vector $\in [-1, 1]$ that allows the quadrotor to move in the corresponding direction with a specific velocity. It is worth noting that the physics of the quadrotor had not been simplified.

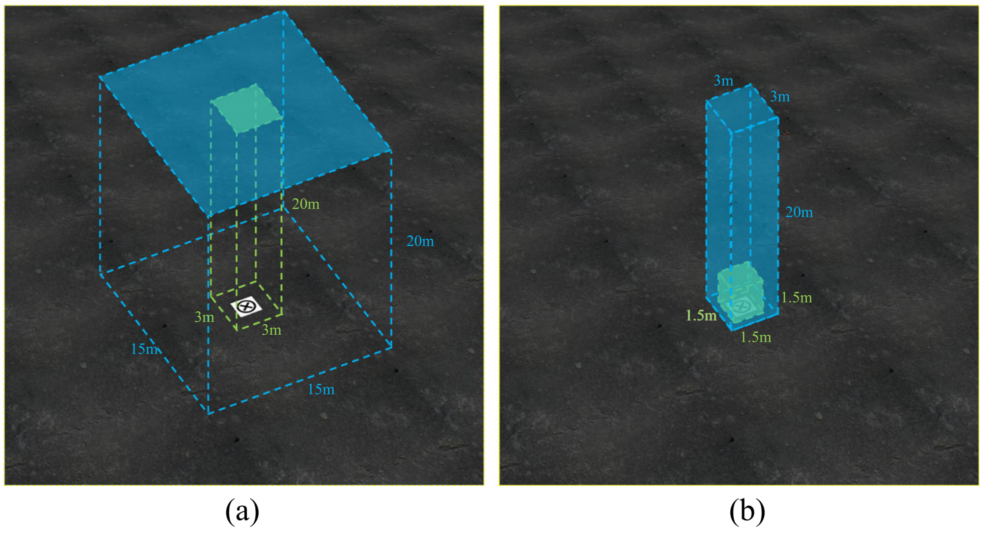


Fig. 5 Initialization area (blue) and target area (green) for (a) marker alignment and (b) vertical landing

5.1 Implementation details

5.1.1 Quadrotor

A 100×100m uniform background texture was placed in the simulation environment, with the ground marker located in the center. In the marker alignment phase, the agent moves only on the X-Y plane. We defined two bounding boxes, with the larger blue and the smaller green bounding boxes representing the exploration space and target zone, respectively, as shown in Fig. 5(a). We spawn the quadrotor at 20m of flight altitude with a random position inside the blue bounding box at the start of each episode. For each episode, the quadrotor is expected to perform a sequence of actions to reach the goal. Here, discrete action was used to stabilize the flight. Specifically, the quadrotor repeats each movement action for 2 seconds at a speed of 0.5m/s, leading to a displacement of 1m in the corresponding direction. The quadrotor receives a positive reward of $+1.0$ when it performs "trigger" inside the green bounding box and receives a negative reward of -1.0 when it performs "trigger" outside the green bounding box. The maximum episode length was 20 time steps. In the vertical descending phase, the quadrotor was allowed to move in 3D space, and thus the state space is considerably larger than that in the marker alignment phase. At the beginning of each episode, we spawn the quadrotor within a blue cube of size $3 \times 3 \times 20$m and expected the quadrotor to reach the target zone, represented by the small green cube of size $1.5 \times 1.5 \times 1.5$m, within a given time limit, as shown in Fig. 5(b). h_{min} was set to 0.3m. Here, the maximum episode length was 40 time steps.

5.1.2 Datasets

We used 10 different uniform asphalt background textures to train the proposed deep model, and used another 3 different asphalt background textures to test the trained model. The materials in these background textures are visually similar. As mentioned before, the variety of ground textures in real-world environments poses a high requirement for the generalization capability of the learned landing policy to unseen environments. Here, we considered 6 different types of background textures to test the generalization capability of the learned landing policy, with each type containing 13 different samples. The materials in these texture backgrounds were completely different from those used in training, and they were not involved in the training process. Figure 6 shows the training background textures and part of the testing background textures. The complete test background textures are shown in the Appendix.

5.1.3 Training

In the training phase, we replaced the asphalt background texture every 50 episodes by randomly choosing one from the 10 training samples. The target network and policy network were synchronized every C time steps. In marker alignment phase, we employed partitioned buffer replay, and choosed $\rho^{+} = 0.25$, $\rho^{-} = 0.25$ and $\rho^{\sim} = 0.5$. However, in the vertical landing phase, we employed our proposed dynamic partitioned experience replay to further improve learning efficiency. Here, we followed [24] to set the priority exponent as 0.6, and linearly anneal the importance sampling exponent from 0.4 to 1; the values of ρ^{+}, ρ^{-} and $\rho^{\sim}$ were dynamically determined based on the average absolute TD error of the corresponding partition. Moreover, we collected a certain amount of preliminary experiences using a random policy before training. However, using the random policy generated very few positive-reward transitions. We employed the data augmentation technique to alleviate this issue. Specifically, we rotated each image by 90°, 180°, and 270°, and then flipped the original image and the three rotated images to expand the positive-reward examples by eight times. Finally, we put N experiences into the replay buffer, with N^{+} positive experiences, N^{-} negative experiences, and $N^{\sim}$ neutral experiences. The whole network was trained for N^{t} steps by using ϵ − greedy exploration strategy with ϵ decreasing from ϵ_b to ϵ_e over the first N^{g} steps and keeping ϵ_e unchanged afterwards. The L_2 loss weight ω decreased from 1 to 0 over the first N^{g} steps. Values of all hyperparameters are listed in Table 1. RMSProp was used as the optimizer. Xavier uniform initialization was utilized to initialize the network weights. The simulation was implemented on the platform with an i7-9700K CPU, RTX2080Ti GPU, and 32GB of RAM.

5.2 Experimental results

5.2.1 Baselines and evaluation matrics

We consider the sequential deep Q-network (SDQN, 2020) [20] and Soft Actor-Critic for discrete action settings (SAC-Discrete, 2019) [45] as the baselines. It is worth pointing out that in [20] Polvara et al. further enhanced the SDQN model using domain randomization (SDQN-DR) by training the model on more groups of environments. However, SDQN-DR contradicts our original intention of generalizing the trained model to novel environments using a single type of training data, and therefore we do not consider SDQN-DR for performance comparison. Following previous works

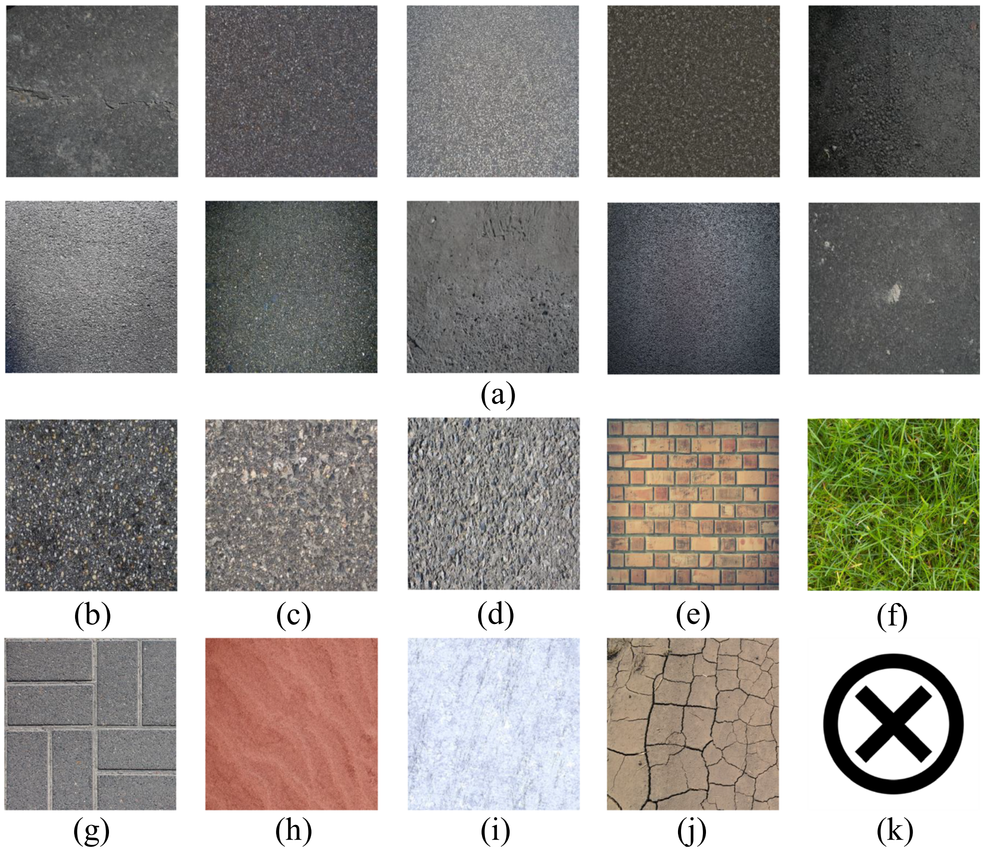


Fig. 6 Background textures used for training: (a) asphalt. Background textures used for testing: (b)-(d): asphalt, (e) brick, (f) grass, (g) pavement, (h) sand, (i) snow, (j) soil. (k) Marker

[20, 21], we adopt the mean episode reward as the evaluation metric. To facilitate explanation, we denote DDQN with auxiliary classification task and regression task as AsDDQN-Cla and AsDDQN-Reg, respectively.

5.2.2 Marker alignment

We present the learning curves of our proposed models AsDDQN-Cla and AsDDQN-Reg, as well as the baselines

Table 1 Parameter settings for simulation

Parameter	Value	
	Marker Alignment	Vertical Landing
Synchronization frames(C)	10000	10000
Positive experience ratio(ρ^+)	0.25	-
Negative experience ratio(ρ^-)	0.25	-
Neutral experience ratio($\rho^\sim$)	0.5	-
Replay buffer size(N)	6×10^4	3×10^5
Positive experiences(N^+)	1.5×10^4	7.5×10^4
Negative experiences(N^-)	1.5×10^4	7.5×10^4
Neutral experiences($N^\sim$)	3×10^4	1.5×10^5
Training frames(N^t)	4.5×10^5	5×10^5
ϵ start value(ϵ_b)	1.0	1.0
ϵ stop value(ϵ_e)	0.1	0.1
ϵ change frames(N^g)	2×10^5	2×10^5
Discount factor(γ)	0.99	0.99
Batch size(b_s)	32	32

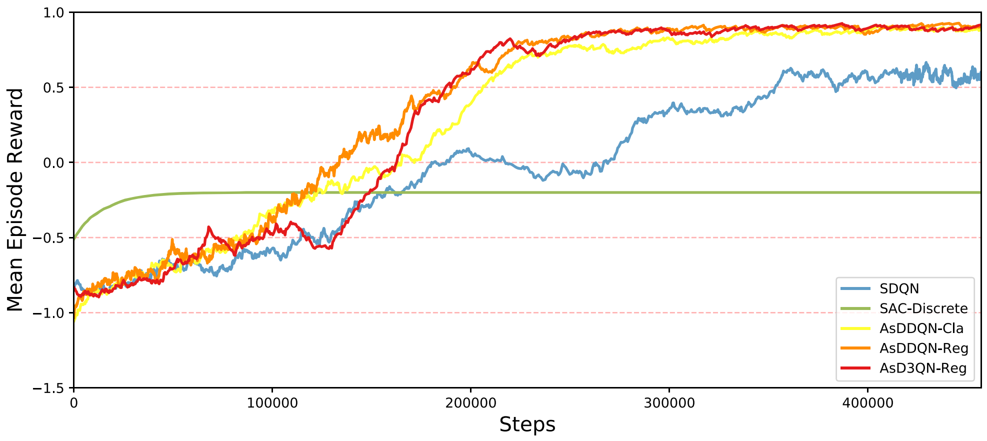


Fig. 7 Learning curves of all agents in marker alignment phase (Sampling using partitioned experience replay method). The abscissa is the number of training steps. The ordinate is mean episode reward, averaged every 10 episodes

in Fig. 7. We note some particular results from these learning curves. First, as compared to the SDQN, the average rewards of AsDDQN-Cla and AsDDQN-Reg increase faster and more stable, and the ultimate convergence results are better. This proves the advantage of adding marker localization as an auxiliary task. The advantage can be explained by the fact that the auxiliary task helps the feature embedding network to learn better feature representations for marker recognition, thus facilitating the learning of a better policy faster. Second, Figure 7 also explores the difference between the two formulations of marker localization, as a classification or a regression task. We can see that the regression agent (AsDDQN-Reg) performs better than the one that does classification (AsDDQN-Cla) in terms of convergence speed. Third, SAC-Discrete converges to -0.2 quickly, which means the quadrotor hovers in the air until a maximum number of 40 steps is finished. This can be explained by insufficient exploration. In the initial training phase, the agent receives a large negative reward for performing "trigger" action incorrectly. In order to maximize the expected return of the policy, the probability of taking "trigger" action is reduced in later period. At the same time, minimizing the error between the action entropy and the target entropy causes the temperature parameter to be gradually reduced to zero as training procedure proceeds. Due to the above two facts, there are few "trigger" action samples in the replay buffer. Without enough "trigger" action samples, the SAC-Discrete algorithm fails to converge to the optimal solution. In comparison, other algorithms use $\epsilon-$greedy for exploration and are not affected by policy updates. Besides, we further enhance the best-performing agent AsDDQN-Reg by using the dueling network architecture [46], named as AsD3QN-Reg. Experimental results show that using the dueling framework brings slight improvements compared to AsDDQN-Reg. This proves that the auxiliary task improves the stability and convergence speed of the model, making the dueling framework unnecessary.

Test on alignment After training, we test the five models on three other asphalt background textures as shown in

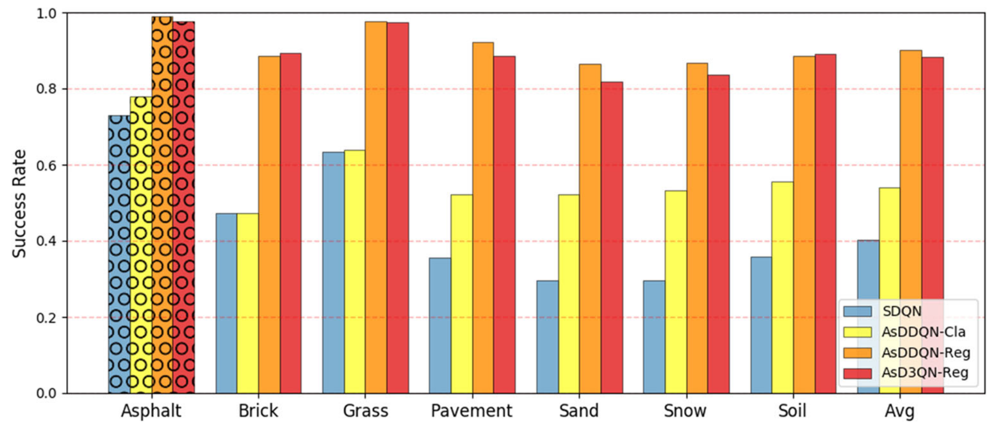


Fig. 8 Test results of marker alignment under different background textures. Asphalt is used to test the performance of the trained model. Brick, grass, pavement, sand, snow and soil are used to test the generalization capability of the trained model, and 'Avg' is the average success rate of these 6 background textures. The model performance test is represented by bar chart with bubbles, and the generalization capability test is represented by bar chart with solid color. The content of the following test result figures has the same meaning

Fig. 6(b)-(d)), which are not involved in training. We test 100 times on each background texture. Every time we spawn the agent inside the pre-defined flying space with a random position and orientation. The bar chart with bubbles in Fig. 8 compares the average success rates of the AsDDQN-Cla, AsDDQN-Reg, SDQN, and AsD3QN-Reg. Here, we do not show the results of SAC-Discrete because it has a success rate of zero in all three environments. We observe that AsDDQN-Cla, AsDDQN-Reg and AsD3QN-Reg all outperform the baseline SDQN, which is consistent with the earlier results obtained from the training procedure. We also notice that the average success rate of the AsDDQN-Reg exceeds the AsDDQN-Cla by a large margin, about 27%, although they achieve similar performances during training. This suggests that the regression agent performs much better in unseen environments compared to the classification agent.

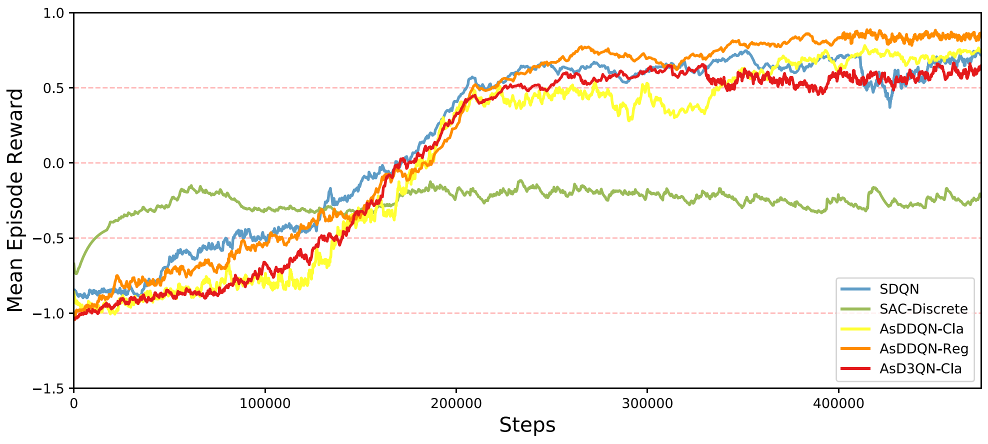
Test on generalization To provide a comprehensive analysis of the generalization capabilities of these different agents, we further test the five models on six different types of background textures, including brick, grass, pavement, sand, snow, and soil, as shown in Fig. 16, all of which are entirely different from the asphalt ones used for training. Each type of texture contains 13 instances, and we test each agent 100 times on each background sample (i.e., 7800 tests in total). The marker alignment performances of all the agents are reported in Fig. 8. Here, we do not show the results of SAC-Discrete because it has a success rate of zero in all test environments. We make the following two observations. First, the addition of marker localization as an auxiliary task significantly enhances the generalization capability of the learned policy. Second, formulating the marker localization as a regression problem (AsDDQN-Reg) further improves the generalization capability of the learned policy by a large margin. In detail, the average success rates of the regression agent on all background textures are all above 0.85. We explain the observation by the fact that formulating the marker localization as a regression task provides stronger supervision signals on the marker's location, thus helping the agent better infer the relative position between the marker and itself. Furthermore, the average success rate of the AsDDQN-Reg model has almost reached its success rate on asphalt. This means that the model learns the marker's characteristics more quickly and accurately by adding the auxiliary task.

5.2.3 Vertical landing

Figure 9 shows the learning curves of SDQN, AsDDQN-Cla, AsDDQN-Reg and SAC-Discrete in the vertical landing phase when the partitioned experience replay is employed for sampling. We note from these curves that the convergence of SDQN is fast but not stable, especially in the latter part of its training procedure. SAC-Discrete doesn't converges to the optimal results due to insufficient exploration of "descend" action like in the marker alignment phase. The convergence of the classification agent AsDDQN-Cla is unstable, but it converges to a better policy than SDQN. Among the three agents, the regression agent AsDDQN-Reg performs the best during training. Its convergence process is stable, and it converges to a pretty good solution. Besides, we further enhance AsDDQN-Cla by using the dueling network architecture, named as AsD3QN-Cla. Experimental results show that the convergence of AsD3QN-Cla becomes more stable than AsDDQN-Cla, but it converges to a worse solution. This could be explained by the fact that different actions lead the quadrotor to reach the target state faster or slower, having a significant impact on the Q values. In contrast, the dueling framework better suits scenes where the state determines the Q values.

Test on landing After training, we repeat the testing procedure as described in Section 5.2.2 to test the five models on 3 different asphalt background textures that

Fig. 9 Learning curves of all agents in vertical landing phase (Sampling using partitioned experience replay method). The abscissa is the number of training steps. The ordinate is mean episode reward, averaged every 10 episodes



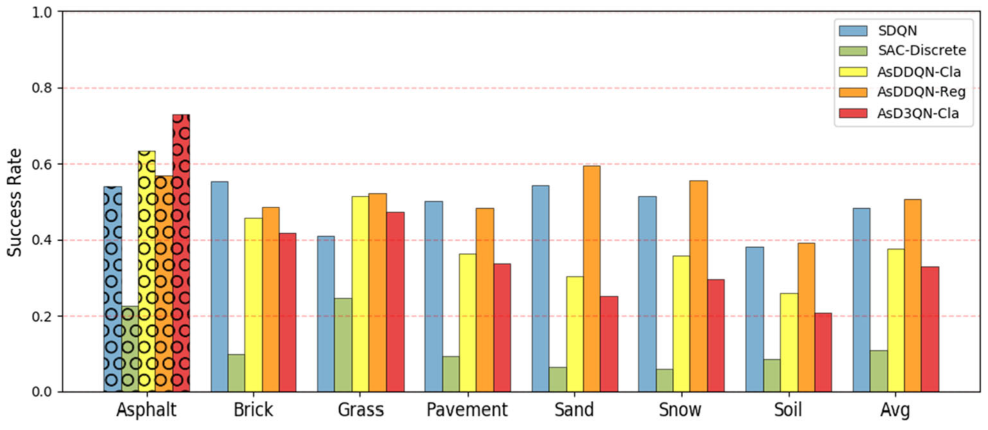


Fig. 10 Test results of vertical landing under different background textures

are not involved in the training. The bar chart with bubbles in Fig. 10 shows the average success rates of the five agents. We note that the average landing success rates of AsDDQN-Cla and AsDDQN-Reg are significantly improved compared with SDQN, especially for AsDDQN-Cla. This proves the advantage of adding marker localization as an auxiliary task in the vertical landing phase. The incorporation of auxiliary localization task provides additional guidance for the feature embedding network to capture more efficient feature representations for marker detection. The policy learned from such feature embeddings could rely on the relative position to better infer actions in different states. SAC-Discrete has a much lower success rate, and the successful vertical landing happens mainly when the initial position of the quadrotor is close to the maker and there is no need to perform the "descend" action. Another notable observation is that the average landing success rate of AsD3QN-Cla is decreased due to using the dueling framework. The dueling framework is proved to be effective in tasks where the states mainly determine the Q values. However, in our vertical landing task, different actions have a significant impact on the Q values.

Test on generalization In order to test the generalization capability of the five agents, we further test them on 6 different types of textures as in the previous phase. The average success rates of the five agents in these unseen environments are shown in Fig. 10. We observe that the regression agent AsDDQN-Reg outperforms the SDQN by a large margin in all six types of environments, about 27% on average. This suggests the strong generalization capability of the learned landing policy from AsDDQN-Reg. However, the classification agent performs worse than the SDQN in all environments except for the grass ones, although it achieves the best performance on the asphalt background textures. The poor performance of AsDDQN-Cla could be explained as follows. As the altitude of the quadrotor decreases, the size of the captured marker becomes larger. When the flight altitude of the quadrotor decreases to some extent, the marker will occupy multiple cells on the observation image. It is tough for the classification agent to recognize which cell the marker locates in at these moments. As a result, the agent will converge to a local optimal solution and eventually overfit in subsequent training. The SAC-Discrete agent shows bad generalization capability since it doesn't converge to good results. In addition, we find that in both

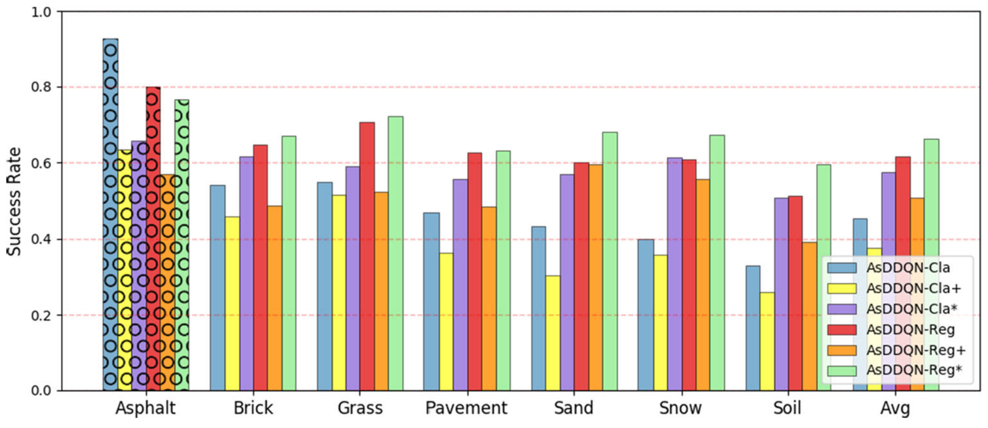


Fig. 11 Comparison of test result of vertical landing under different background textures. AsDDQN-Cla and AsDDQN-Reg use partitioned buffer replay method for sampling, AsDDQN-Cla$^+$ and AsDDQN-Reg$^+$ use prioritized buffer replay method for sampling, AsDDQN-Cla* and AsDDQN-Reg* use dynamic partitioned experience replay method for sampling

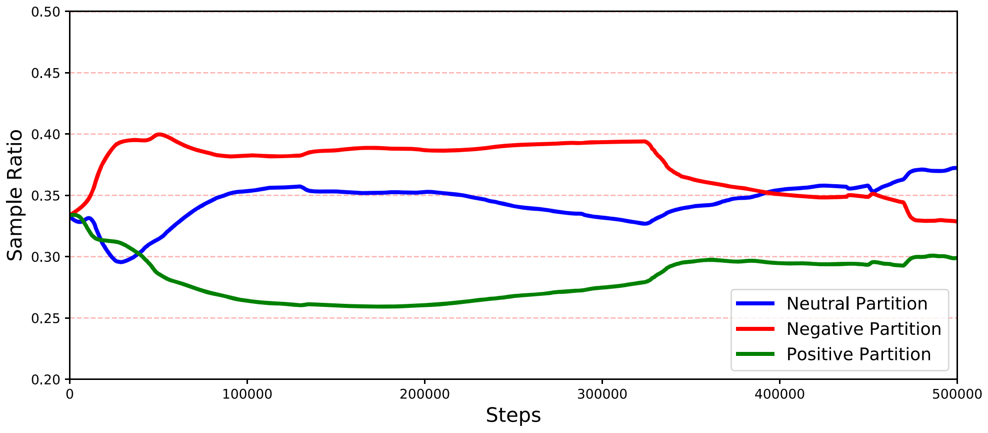


Fig. 12 The changes in sampling ratios of different partitions during training of AsDDQN-Reg in the vertical landing phase when dynamic partitioned experience replay is employed

marker alignment and vertical landing, adding the dueling framework does not improve the model, and there is no need to introduce the dueling framework.

Test on sampling In the vertical landing phase, the quadrotor moves in 3D space. The vast state space makes the problem of sparse and delayed rewards deteriorate. To alleviate this problem, we use the dynamic partitioned experience replay method to perform transition experience sampling. We also compare it with the partitioned buffer replay method and prioritized buffer replay method to highlight the advantages of our dynamic partitioned experience replay method. Both the prioritized buffer replay and our method update the priorities of the sampled experiences at each time step.

Figure 11 shows the landing performances of AsDDQN-Cla and AsDDQN-Reg using different sampling strategies, including partitioned buffer replay, prioritized buffer replay, and our dynamic partitioned experience replay sampling methods. Although the success rate of the model on the asphalt background textures is reduced after using dynamic partitioned experience replay compared with partitioned buffer replay, the success rate on all other different types of background textures that are not involved in the training is improved by a large margin. By using the dynamic partitioned experience replay sampling method, the average success rate of the generalization capability test of AsDDQN-Cla and AsDDQN-Reg is increased by 27% and 7% compared with partitioned buffer replay, and 53% and 31% compared with prioritized buffer replay. This observation demonstrates the effectiveness of our proposed dynamic partitioned experience replay method in improving the landing accuracy and generalization of the learned policy.

To gain a deeper insight into our dynamic partitioned experience replay mechanism, for each partition we show its changes in samples ratio as training proceeds in Fig. 12. Here, X and Y axes represent the index of time steps and the proportion of experiences sampled from the partition, respectively. We make the following observations from the Figure. In the initial training period, the negative partition's sampling ratio keeps increasing, and negative partition occupies a large proportion of the selected samples. The dominance of the negative partition lasts for an extended period until the late part of the training. At this point, the neutral partition's sampling ratio starts to increase and surpass that of the negative partition. Throughout the entire training procedure, the sample ratio of the positive partition is relatively small. This observation reveals that the learning initially focuses on avoiding negative states with a large

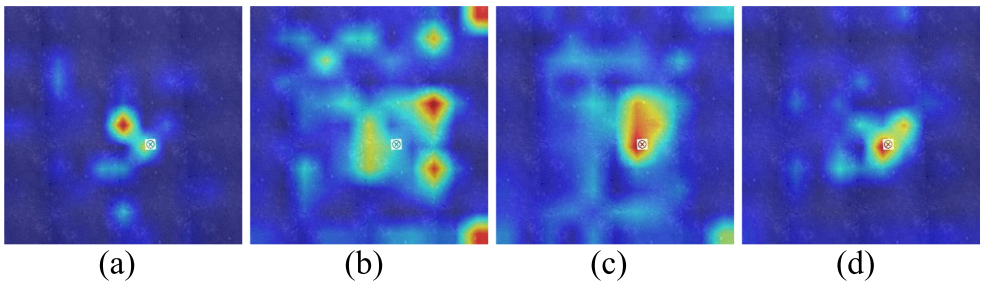


Fig. 13 Visualization of extracted features from different models in the marker alignment phase: (a) SDQN, (b) As-DDQN-Cla, (c) As-DDQN-Reg, (d) As-D3QN-Reg

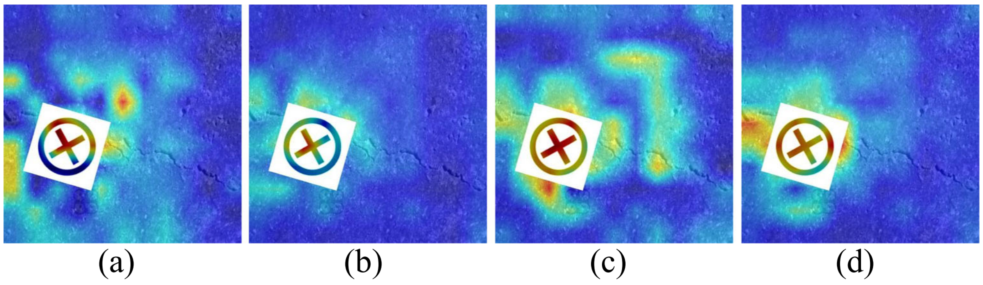


Fig. 14 Visualization of extracted features from different models in the vertical landing phase: (a) SDQN, (b) As-DDQN-Cla, (c) As-DDQN-Reg, (d) As-D3QN-Cla

penalty and then shifts to reaching the marker and landing precisely.

5.3 Feature visualization

Experiment results have demonstrated that adding auxiliary localization tasks improves the learned policy model's generalization capability. In this section, we provide an in-depth explanation of the observation by visualizing the features extracted by the convolutional layers of difference models.

Figure 13 shows the visualization of the features extracted by the convolutional layers of different models in the marker alignment phase using heat maps. The red area represents apparent features, while the blue area is the opposite. Due to the limitations of the feature extraction capability, the most apparent features extracted by the SDQN model deviate from the marker. The area covered by the features extracted by AsDDQN-Cla is too large, and the deviation is significant. The region corresponding to the classification label is too large, and the features extracted by the trained model are not accurate enough, degrading the robustness of the learned policy. The model cannot accurately locate the marker under diverse background textures, and the alignment success rate is low. The features extracted by the regression agent are concentrated. As shown in Fig. 13(c)-(d), the region where the maker is located has the most apparent features, and the features extracted by the model are accurate. The policy that is further learned by using accurate features is more robust, and the alignment success rate is higher under diverse background textures.

Figure 14 is the visualization of the features extracted from different models during the vertical landing phase. The features extracted by the SDQN model are too scattered and do not focus on the marker in the image. The features extracted by the AsDDQN-Cla model are more concentrated and distributed around the marker. However, adding the dueling structure does not improve the feature extraction capability. The features extracted by AsD3QN-Cla are still scattered. The features extracted by the AsDDQN-Reg model are most concentrated, and the most apparent feature is from the marker. Due to the quadrotor's altitude change, the marker's size in its field of view changes significantly, which increases the complexity of feature extraction. Compared with the features extracted by the

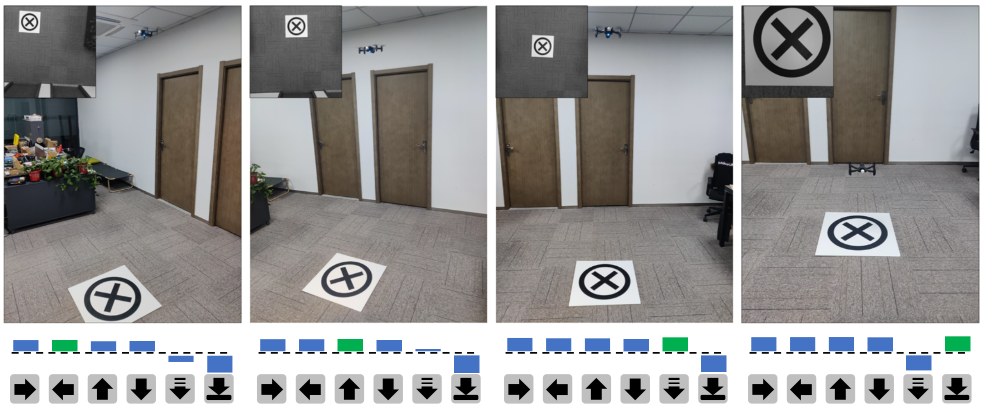


Fig. 15 Snapshots of the indoor vertical landing phase flight. The upper left corner is the picture taken by the quadrotor. At the bottom is the Q values of each action predicted based on the input image (the maximum Q value is green). From left to right, each action icon represents "turn right", "turn left", "go forward", "go backward", "descend" and "landing"

model in the marker alignment phase, the accuracy of the features extracted at this phase is still insufficient, resulting in the landing success rate is not high enough.

5.4 Real-world experiments

We tested the trained model in the real-world environment, using the Parrot Anafi quadrotor as the test platform. As the AsDDQN-Reg model has the highest success rate in the simulation environment, we used it both in the marker alignment phase and vertical landing phase. We conducted landing experiments in a variety of indoor and outdoor environments with different background textures. The quadrotor starts to land at a random location. The quadrotor has seven executable actions like simulation: “go forward”, “go backward”, “turn left”, “turn right”, “trigger”, “descend”, and “landing”. When testing in an outdoor environment, the “go forward”, “go backward”, “turn left”, “turn right”, and “descend” actions respectively produce a displacement of 1m in the corresponding direction; when testing in an indoor environment, due to the limitation of the movable space, we only test the vertical landing phase and reduce the displacement generated by the actions to 0.25m.

The model trained in Gazebo can be seamlessly transferred to a real quadrotor for testing, and the performance is remarkable. Figure 15 shows some snapshots during the test. The prediction of Q values conforms to the reward design: the Q value of the action that can reach the marker as soon as possible is the largest. Since we do not penalize the non-shortest path too much, the Q values of some other actions are also very close to the maximum Q value. Since the “landing” action should only be taken within the target zone, the Q value of the “landing” action is negative in most cases due to the smaller area of the target zone.

6 Discussion

Based on the above experiment results, we conclude that adding marker localization as an auxiliary task enhances the convergence speed and helps to find a better policy. The learned policy has proved to perform better in both seen and unseen environments compared with SDQN. In particular, the regression agent achieves the best performance, with an average success rate over 0.9 and 0.62 in 78 unseen environments in the marker alignment phase and vertical landing phase, respectively. Feature visualization has been performed to provide an in-depth explanation of the experiment results. The visualization shows that adding auxiliary tasks can help the agent better capture the features related to the landing marker compared with SDQN. With better features, the agent can locate the marker more accurately thereby improving the robustness of the learned policy.

Besides, the dynamic partitioned experience replay method further improves the model’s generalization capability by making learning from experiences more efficient. The changes in sampling ratios of different partitions show the model can focus on different types of samples during training. With the more efficient sampling method, the regression agent achieves the best performance with an average success rate above 0.66 in 78 unseen environments in the vertical landing phase.

Despite the fact that the auxiliary localization tasks and dynamic partitioned experience replay method have greatly improved the model’s performance, our approach still has limitations. The model is influenced by the environment background when extracting marker features. As seen in the feature visualization, the model will notice regions where the color is close to the marker color. This may limit the performance of the model in complex environments. In the future, we hope to eliminate the influence of environment background on model feature extraction, allowing the model to work stably in complex environments.

7 Conclusions

In this paper, we use DDQN with auxiliary tasks to achieve the quadrotor autonomous landing. Adding the auxiliary regression task in the marker alignment and vertical landing phases steadily improves the success rate of the quadrotor landing under diverse background textures. Adding auxiliary tasks also solves the unstable convergence problem in SDQN and SAC-Discrete, allowing the model to generalize better and perform stably in unseen environments. Besides, we propose the dynamic partitioned experience replay sampling method, which stabilizes the training procedure and improves the efficiency of learning from experiences. Future work will focus on eliminating the effect of environment background on feature extraction so that the model can work in more complex environments.

Appendix

To test the trained model’s generalization capability, the ground background texture is replaced with textures not involved in training for testing. As shown in Fig. 16, there are 6 types of textures, each of which contains 13 different instances.

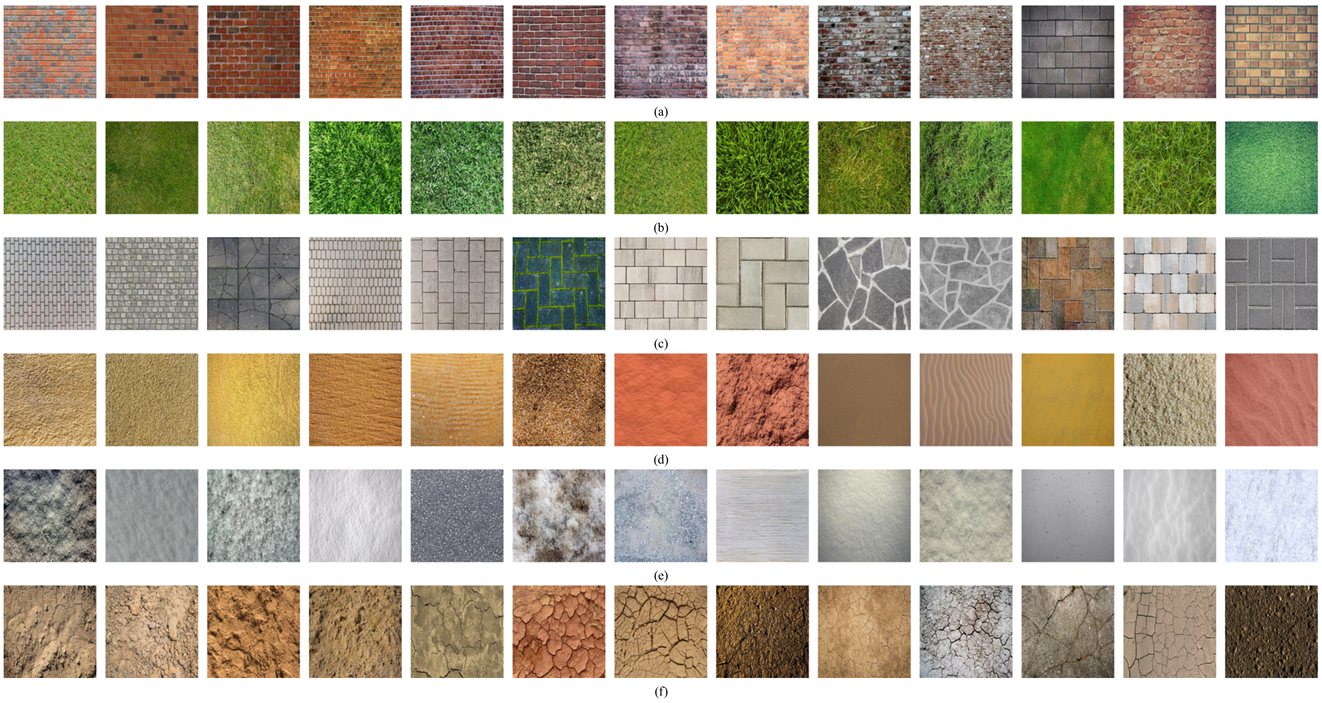


Fig. 16 The background textures used in the model generalization test: (a) brick, (b) grass, (c) pavement, (d) sand, (e) snow, (f) soil

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Silvagni M, Tonoli A, Zenerino E, Chiaberge M (2017) Multipurpose uav for search and rescue operations in mountain avalanche events. Geomatics, Natural Hazards and Risk 8(1):18–33
2. Whitehead K, Hugenholtz CH (2014) Remote sensing of the environment with small unmanned aircraft systems (uass), part 1: A review of progress and challenges. Journal of Unmanned Vehicle Systems 2(3):69–85
3. Yang S, Yang X, Mo J (2018) The application of unmanned aircraft systems to plant protection in china. Precision agriculture 19(2):278–292
4. Yang T, Li Z, Zhang F, Xie B, Li J, Liu L (2019) Panoramic uav surveillance and recycling system based on structure-free camera array. IEEE Access 7:25763–25778
5. Tanaka S, Senoo T, Ishikawa M (2019) High-speed uav delivery system with non-stop parcel handover using high-speed visual control. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE, pp 4449–4455
6. Dai Z, Yi J, Zhang Y, Zhou B, He L (2020) Fast and accurate cable detection using cnn. Appl Intell 50(12):4688–4707
7. Tian G, Liu J, Zhao H, Yang W (2021) Small object detection via dual inspection mechanism for uav visual images. Appl Intell, pp 1–14
8. Lee S, Shim T, Kim S, Park J, Hong K, Bang H (2018) Vision-based autonomous landing of a multi-copter unmanned aerial vehicle using reinforcement learning. In: 2018 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, pp 108–114
9. Al-Sharman MK, Emran BJ, Jaradat MA, Najjaran H, Al-Husari R, Zweiri Y (2018) Precision landing using an adaptive fuzzy multi-sensor data fusion architecture. Applied soft computing 69:149–164
10. Talha M, Asghar F, Rohan A, Rabah M, Kim SH (2019) Fuzzy logic-based robust and autonomous safe landing for uav quadcopter. Arab J Sci Eng 44(3):2627–2639
11. Gui Y, Guo P, Zhang H, Lei Z, Zhou X, Du J, Yu Q (2013) Airborne vision-based navigation method for uav accuracy landing using infrared lamps. J Intelligent & Robotic Systems 72(2):197–218
12. Tang D, Hu T, Shen L, Zhang D, Kong W, Low KH (2016) Ground stereo vision-based navigation for autonomous take-off and landing of uavs: a chan-vese model approach. Int J Adv Robot Syst 13(2):67
13. Kalinov I, Petrovsky A, Agishev R, Karpyshev P, Tsetserukou D (2021) Impedance-based control for soft uav landing on a ground robot in heterogeneous robotic system. In: 2021 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, pp 1653–1658
14. Almeshal AM, Alenezi MR (2018) A vision-based neural network controller for the autonomous landing of a quadrotor on moving targets. Robotics 7(4):71
15. Khazetdinov A, Zakiev A, Tsoy T, Svinin M, Magid E (2021) Embedded aruco: a novel approach for high precision uav landing. In: 2021 International Siberian Conference on Control and Communications (SIBCON), IEEE, pp 1–6
16. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing atari with deep reinforcement learning. arXiv:1312.5602
17. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. nature 518(7540):529–533

18. Zhang F, Leitner J, Milford M, Upcroft B, Corke P (2015) Towards vision-based deep reinforcement learning for robotic motion control. arXiv:1511.03791
19. Tai L, Paolo G, Liu M (2017) Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In: 2017 IEEE/RSJ International conference on intelligent robots and systems (IROS), IEEE, pp 31–36
20. Polvara R, Patacchiola M, Hanheide M, Neumann G (2020) Sim-to-real quadrotor landing via sequential deep q-networks and domain randomization. Robotics 9(1):8
21. Xu Y, Liu Z, Wang X (2018) Monocular vision based autonomous landing of quadrotor through deep reinforcement learning. In: 2018 37th Chinese control conference (CCC), IEEE, pp 10014–10019
22. Le L, Patterson A, White M (2018) Supervised autoencoders: Improving generalization performance with unsupervised regularizers. Adv Neural Info Process Systems 31:107–117
23. Sun Y, Wang X, Liu Z, Miller J, Efros A, Hardt M (2020) Test-time training with self-supervision for generalization under distribution shifts. In: International conference on machine learning, PMLR, pp 9229–9248
24. Schaul T, Quan J, Antonoglou I, Silver D (2015) Prioritized experience replay. arXiv:1511.05952
25. Sutton RS, Barto AG (2018) Reinforcement learning: An introduction. MIT press
26. Kavuk EM, Tosun A, Cevik M, Bozanta A, Sonuç SB, Tutuncu M, Kosucu B, Basar A (2021) Order dispatching for an ultra-fast delivery service via deep reinforcement learning. Appl Intell, pp 1–26
27. Hui TS, Ishak MK, Mohamed MFP, Fadzil LM, Ahmarofi AA (2021) Balancing excitation and inhibition of spike neuron using deep q network (dqn). In: Journal of physics: Conference series, vol 1755, IOP Publishing, p 012004
28. Al-Gablawy M (2021) Optimal peak shifting of a domestic load connected to utility grid using storage battery based on deep q-learning network. Int J Energy Res 45(2):3269–3287
29. Van Hasselt H, Guez A, Silver D (2016) Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI conference on artificial intelligence, vol 30
30. Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning, PMLR, pp 1861–1870
31. Olivares-Méndez MA, Mondragón IF, Campoy P, Martínez C (2010) Fuzzy controller for uav-landing task using 3d-position visual estimation. In: International conference on fuzzy systems, Ieee, pp 1–8
32. Keipour A, Pereira GAS, Bonatti R, Garg R, Rastogi P, Dubey G, Scherer S (2021) Visual servoing approach for autonomous uav landing on a moving vehicle. arXiv:2104.01272
33. Saavedra-Ruiz M, Pinto-Vargas AM, Romero-Cano V (2021) Monocular visual autonomous landing system for quadcopter drones using software in the loop. IEEE Aerosp Electron Syst Mag
34. Lange S, Sunderhauf N, Protzel P (2009) A vision based onboard approach for landing and position control of an autonomous multirotor uav in gps-denied environments. In: 2009 International conference on advanced robotics, IEEE, pp 1–6
35. Huang X, Xu Q, Wang J (2019) Vision-based autonomous landing of uav on moving platform using a new marker. In: IOP Conference series: Materials science and engineering, vol 646, IOP Publishing, p 012062
36. Lebedev I, Erashov A, Shabanova A (2020) Accurate autonomous uav landing using vision-based detection of aruco-marker. In: International conference on interactive collaborative robotics, Springer, pp 179–188
37. Niu G, Yang Q, Gao Y, Pun M-O (2021) Vision-based autonomous landing for unmanned aerial and mobile ground vehicles cooperative systems. IEEE robotics and automation letters
38. Vankadari MB, Das K, Shinde C, Kumar S (2018) A reinforcement learning approach for autonomous control and landing of a quadrotor. In: 2018 International conference on unmanned aircraft systems (ICUAS), IEEE, pp 676–683
39. Shim T, Bang H (2018) Autonomous landing of uav using vision based approach and pid controller based outer loop. In: 2018 18th International conference on control, automation and systems (ICCAS), IEEE, pp 876–879
40. Kim C, Lee EM, Choi J, Jeon J, Kim S, Myung H (2021) Roland: Robust landing of uav on moving platform using object detection and uwb based extended kalman filter. In: 2021 21st International conference on control, automation and systems (ICCAS), IEEE, pp 249–254
41. Zhang H-T, Hu B-B, Xu Z, Cai Z, Liu B, Wang X, Geng T, Zhong S, Zhao J (2021) Visual navigation and landing control of an unmanned aerial vehicle on a moving autonomous surface vehicle via adaptive learning. IEEE Trans Neural Networks and Learning Systems
42. Guo K, Tang P, Wang H, Lin D, Cui X (2022) Autonomous landing of a quadrotor on a moving platform via model predictive control, vol 9
43. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Information Processing Systems 25:1097–1105
44. Narasimhan K, Kulkarni T, Barzilay R (2015) Language understanding for text-based games using deep reinforcement learning. arXiv:1506.08941
45. Christodoulou P (2019) Soft actor-critic for discrete action settings. arXiv:1910.07207
46. Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N (2016) Dueling network architectures for deep reinforcement learning. In: International conference on machine learning, PMLR, pp 1995–2003

Jiawei Wang received the B.S. degree from Southeast University, Nanjing, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Control Science and Engineering, Tongji University, Shanghai, China. His research interests include pattern recognition, reinforcement learning, and visual navigation.

Teng Wang received Ph.D. degree from Iowa State University, Ames, USA, in 2016. She is currently an assistant professor with the School of Automation at Southeast University, Nanjing, China. Her research interests including pattern recognition, machine vision, and visual navigation.

Wenzhe Cai received the B.S. degree from Southeast University, Nanjing, China, in 2019. He is currently a PhD candidate in department of automation at Southeast University. His research interests include machine learning, pattern recognition, reinforcement learning and visual navigation.

Zichen He received the B.S. and M.S. degrees in mechanical engineering from the China University of Petroleum, Beijing, China, in 2019. He is currently pursuing the Ph.D. degree in control science and engineering with Tongji University, Shanghai, China. His research interests include reinforcement learning, multi-robot collaborative navigation, and motion planning.

Changyin Sun received the B.S. degree in applied mathematics from the College of Mathematics, Sichuan University, Chengdu, China, in 1996, and the M.S. and the Ph.D. degrees in electrical engineering from Southeast University, Nanjing, China, in 2001 and 2004, respectively. He is currently a Professor with the School of Automation, Southeast University, Nanjing, China. His current research interests include intelligent control, flight control, and optimal theory. Dr. Sun is an Associate Editor of the IEEE Transactions on Neural Networks and Learning Systems, Neural Processing Letters, and the IEEE/CAA Journal of Automatica Sinica.