Xiaoling Xu
Yiliang Li ✉
Gaowei Lei

https://doi.org/10.21278/TOF.493070924
ISSN 1333-1124
eISSN 1849-1391

ROBOTIC ARM GRASPING OF INVISIBLE OBJECTS IN CLUTTERED ENVIRONMENTS

Summary

Grasping invisible targets using a robotic arm in cluttered environments remains a significant challenge. Many existing methods depend on redundant operations, which can compromise efficiency. In this paper, we introduce a novel framework called Deep Reinforcement Attention DenseNet (DRA-DenseNet), which utilises self-attention mechanisms for Q-value prediction. This framework excels at capturing global features of the objects, significantly enhancing the accuracy of action prediction in cluttered environments. We develop a search strategy that leverages the capabilities of DRA-DenseNet along with historical action data to locate invisible targets. Further, we propose an action coordination strategy to achieve more accurate grasping by training both the push network and the grasp network simultaneously. Experiments in both simulated and real-world environments demonstrate that the proposed approach outperforms many existing methods in terms of motion efficiency and success rate.

Key words: *grasping, invisible targets, action coordination, domain knowledge, self-attention mechanisms*

1. Introduction

Grasping is a fundamental operation in robotics, and is widely employed in industrial production [1] as well as in human-machine interactions [2]. Although robot grasping technology has been researched for several decades, most of the research efforts have concentrated on grasping a single object within a simple environment [3]. However, in practical applications, robots often operate in cluttered environments where objects are occluded. This complexity poses challenges in accurately identifying target objects and their optimal grasping positions, which directly affects the success rate of robotic grasping.

To address the challenges, researchers have developed strategies for the grasping task in cluttered environments. For instance, Tan et al. [4] introduced the stochastic continuous time recurrent neural network to achieve flexible grasping motion. However, the model lacks sufficient generalisation ability in cluttered environments. Novkovic et al. [5] constructed an active and interactive perception system based on reinforcement learning for object grasping. Nevertheless, the method assumes that the object has a specific colour, thereby restricting the universality of the algorithm. Fang et al. [6] proposed a task-oriented grasping network based on large-scale simulated self-supervision for grasping. Unfortunately, the model also exhibits poor generalisation when dealing with unseen objects.

In recent years, deep reinforcement learning (DRL) has significantly advanced the field of robotic grasp control [7]. The DRL method excels in learning complex, high-dimensional mappings even under sparse reward, highlighting the potential of robot grasp tasks. For example, in [8], a visual pushing for grasping method using the DRL method is proposed. This approach evaluates target states under cluttered conditions using predicted confidence levels and guides the robot to perform appropriate manoeuvres. However, the pushing is limited, serving mainly to separate objects and to create optimal grasping spaces. Xu et al. [9] developed a three-stage prediction network that trains the robot to proficiently handle grasping tasks in diverse environments. In scenarios where targets are severely obscured or invisible, Yang et al. [10] proposed a Bayesian-based policy that leverages historical action to statistically estimate pushing actions for target localisation. However, their method is often inefficient due to redundant actions. It is also impractical to use the estimates of pushing motion learned from visible target instances to predict motion execution positions. In [11], a graph-based DRL model is presented to detect hidden objects. Nevertheless, this method still suffers from excessive redundancy and lacks accuracy in predicting pushing and grasping motions. Some learning strategies have been developed to improve the synergy between pushing and grasping. Among these strategies, Kiatos et al. [12] learned direct mapping from visual observations to actions, and utilised a fully convolutional network to predict the best push-grasp action. Tekden et al. [13] proposed a graph neural network-based framework to predict the pushing task. Al-Shanoon et al. [14] introduced a self-learning strategy to grasp unfamiliar-adjacent objects. Yu et al. [15] combined visual transformers with convolutional techniques to enhance robot performance in push-grasp tasks. The method improves the processing of visual information and operation planning. However, the preceding discussion assumes clear visibility of the target.

To further elaborate on the challenges of grasping obscured targets, several studies have been performed. In [16], a probability-driven active learning strategy is introduced for searching for occluded objects based on a Gaussian process model. This strategy enhances search efficiency by intelligently prioritising tasks. However, the high computational complexity of this model affects its real-time performance. In [17], an algorithm that learns occupancy distribution is developed to improve success rates in grasping invisible objects. Nevertheless, the performance of the algorithm declines when dealing with objects whose shapes significantly differ from the shapes in the training data. Wu et al. [18] proposed a push-grasp method based on reinforcement learning for multiple objects in clutter. Unfortunately, the generalisation ability of this algorithm in relation to unseen target objects is still not high.

To address the challenge of obscured target objects in cluttered environments, the paper explores target grasping strategies for robotic arms. Compared with other methods, this algorithm has more powerful search and grasp abilities, and shows good generalisation ability in complex scenarios. The main contributions of this work are addressed as follows:

(1) A DRA-DenseNet (Deep Reinforcement Attention-DenseNet) model is proposed. The action Q-value prediction deep network that integrates self-attention mechanisms with the DenseNet model is constructed. This model can effectively capture global object features, significantly enhancing the localisation accuracy of invisible targets in cluttered environments.

(2) To tackle the challenge of searching for severely occluded, invisible targets, a search strategy based on the DRA-DenseNet and the search experience mask constructed by the past action experience is explored. It can minimise redundancy and ineffective searches.

(3) A pixel-level push-grasp action collaborative strategy is introduced for grasping invisible targets. The strategy constructs pixel-level target gap mask maps to accurately evaluate the grasp space. Besides, the push network and the grasp network are trained simultaneously by using the DRA-DenseNet. It allows the robotic arm to flexibly execute grasp and push actions in various situations, thereby performing the grasping action more precisely.

The remainder of the paper is organised as follows. Section 2 describes the robotic arm system. Section 3 analyses the grasping strategy based on the Deep Q-Network(DQN) and

hindsight experience replay(HER). Section 4 introduces the DRA-DenseNet while Section 5 presents the searching strategy. Section 6 proposes the push-grasp collaborative strategy and Section 7 shows the simulation and experimental results. Finally, section 8 concludes our work and highlights potential research directions.

2. Description of the robotic arm system

To improve the overall efficiency of target acquisition, we focus on two key components, namely searching and grasping. Fig. 1 describes the robotic arm system and illustrates the architecture of our proposed grasping strategy.

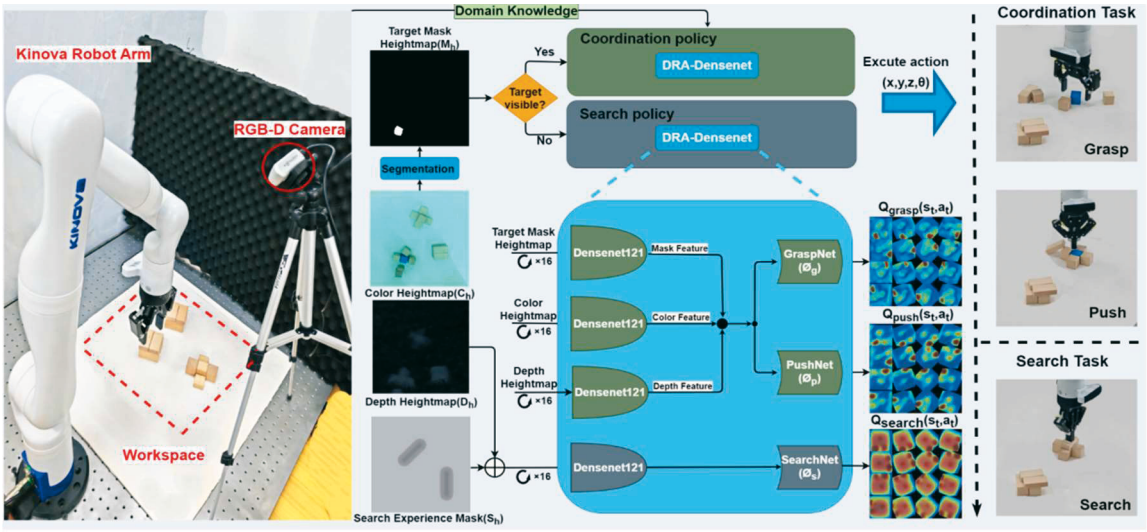


Fig. 1 The architecture of the proposed grasping approach

In Fig.1, both colour and depth images are captured through the RGB-D camera in the workspace. Each pixel in the colour image corresponds precisely to the depth information of the corresponding pixel in the depth map. We can generate the target masks by performing colour semantic segmentation on these colour images using the lightweight RefineNet [19]. The target mask can be used to evaluate the visibility of the target and guide the implementation of the appropriate strategies. Moreover, the colour and depth maps can be utilised to create the heightmaps by projecting 3D point clouds on a vertical plane. The sizes of depth heightmap D_h, colour heightmap C_h, and target mask heightmap M_h are defined as 224×224.

Each heightmap (224×224) corresponds to the two-dimensional coordinate system of the workplace (0.448m×0.448m), so each pixel on the heightmap corresponds to a 2mm×2mm area in the workspace. The heightmaps are rotated in different directions to achieve different motion orientations, and are input into the motion critic to predict pixel-wise Q-values for pushing and grasping actions. This rotation of the heightmaps is significant as it enables the robot to better adapt to various scenarios. By analysing the rotated heightmaps, the robotic arm can formulate appropriate action strategies through flexible processing of the heightmaps to achieve efficient pushing and grasping operations.

The action space of the robotic arm can be parameterised by the tuple (x, y, z, ψ_i), where (x, y, z) represents the spatial coordinates of the robotic arm, and ψ_i represents the orientation of the gripper around the z-axis [20]. During a search action, the gripper positions itself above the target. In a push action, the gripper closes first. Then, the arm aligns the gripper at the predefined rotation angle ψ_i and descends to the exact 3D coordinate (x, y, z). Finally, the gripper pushes a fixed distance in the angle ψ_i. For accurate control, the rotation angle ψ_i can be set to any of the discrete values given by $\psi_i = \pi i/8$, $1 \leq i \leq 16$, which represent the possible action angles. After completing the push, the arm returns to its initial position. The grasping

action is executed by adjusting the gripper to a set width before descending to grasp at the given coordinate. The gripper then closes and grasps the target. We set a desired closure distance based on the predetermined size of the target object. If the distance matches or is close to the preset value, the grasp is deemed successful.

The gripper remains perpendicular to the workspace in the system. The vertical position z is derived from depth information, which can determine the height of the action. Considering these factors, x, y, and ψ_i are sufficient to define the action space of the robotic arm [21]. Once the image information of the workspace is captured, it is fed into a deep neural network to obtain the action heatmaps (Q-maps). The pixel coordinates (u, v) corresponding to the point (x, y) can be obtained under the robotic arm's coordinate system. The process can be described as follows. Based on the depth information d and the scale parameter s of the depth camera, the z-axis coordinate z_c in the camera coordinate system is calculated as $z_c = d/s$. Next, x_c and y_c can be calculated using the intrinsic focal lengths f_x and f_y of the camera and the principal point coordinates (u_0, v_0), $x_c = (u - u_0)\, z_c / f_x$, $y_c = (v - v_0)\, z_c / f_y$. The coordinates ($x_c$, y_c, z_c) can be converted into the robotic arm's coordinates (x, y, z) using the transformation matrix T which can be defined as:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = T \times \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}. \tag{1}$$

This conversion enables us to relate the positions observed by the camera to the actual positions in the robotic arm's coordinate system. Building on this foundation, the robotic arm's grasping capability is explored within cluttered environments.

3. Grasping strategy based on DQN and HER

To grasp the target object, a robotic arm grasping strategy based on DQN and HER [22] is developed. It aims to learn the value of state-action pairs and help the robotic arm to control its movement. The robotic arm's actions are modelled within a Markov Decision Process (MDP). Let us assume that the state s_t of the MDP model for the robotic arm is represented by the depth heightmap D_h^t, the colour heightmap C_h^t, and the target mask heightmap M_h^t at time t. In this model, the robotic arm selects an action a_t based on the environmental state s_t and the policy π at time t. After performing the action, the robotic arm transitions to the next state s_{t+1} according to the state transition probability $P(s_{t+1} | s_t, a_t)$. Then, it receives an immediate reward $r(s_t, a_t)$ based on the reward function $R(s_t, a_t, s_{t+1})$. The reward function can be a search reward R_s, a grasping reward R_g,, or a pushing reward R_p, depending on the type of action executed. The objective of the robotic arm is to learn an optimal policy π^* that selects the optimal action based on the environmental state to maximise the expected accumulated discounted reward from searching, grasping or pushing, as follows:

$$\begin{aligned} &\pi^*(s) \leftarrow \text{argmax}_a\, Q_\pi(s,a) \\ &Q_\pi(s,a) = \mathbb{E}_\pi[\textstyle\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_t = s, a_t = a\,], \end{aligned} \tag{2}$$

where the function $Q_\pi(s,a)$ is the expected accumulated reward after taking an action a starting with state s under the strategy π, and γ is a discount factor. The process ensures that the robotic arm gradually learns from experiences and optimises its grasping strategy.

To enhance the exploration of the behaviour space and foster a more comprehensive understanding of its environment, the robotic arm employs an ε-greedy strategy to select an action based on the current state. As training progresses, the probability on random exploration diminishes, allowing the robotic arm to increasingly depend on its learned value function to guide its behaviour. The ε-greedy strategy can be defined as:

$$\pi(a_t|s_t)=\begin{cases}\varepsilon & a_t \neq \arg\max_{a\in A} Q(s_t,a)\\ 1-\varepsilon & a_t = \arg\max_{a\in A} Q(s_t,a)\end{cases}. \tag{3}$$

The robotic arm interacts with the environment based on the selected action and then observes a new state and reward. This process produces a transition that is stored in the experience replay buffer. In practical training, a random set of samples is taken from the experience replay buffer. For DQN, these samples are then used to train the network, where the goal is to minimise the loss function $L(\theta)$ to train the network parameters θ:

$$\begin{aligned} &L(\theta)=\mathbb{E}\left[f_l(e_t(\theta))\right]\\ &f_l(e_t(\theta))=\begin{cases}0.5e_t(\theta)^2 & |e_t(\theta)|<1\\ |e_t(\theta)|-0.5 & \text{otherwise}\end{cases}\\ &e_t(\theta)=Q(s_t,a_t;\theta)-\left(r+\gamma\max_{a'}Q(s_{t+1},a';\theta^-)\right), \end{aligned} \tag{4}$$

where $f_l(e_t(\theta))$ is the smooth L1 loss function, with this function being smoothed if $|e_t(\theta)|<1$. The performance of the strategy can be determined by evaluating the loss function, following which the strategy optimisation direction can be determined. The error $e_t(\theta)$ is the difference between the predicted Q-value and the optimal Q-value at state s_t. r is a reward for taking an action in state s_t. $\max_{a'} Q(s_{t+1},a',\theta^-)$ is the optimal Q-value for all possible actions a' at the next state s_{t+1}, which is calculated using the network parameters θ^- to increase the stability of the training process. By minimising the loss function $L(\theta)$, the robotic arm learns how to adjust its parameter θ to better predict the long-term cumulative reward for taking a particular action in a given state.

Besides, we face a major challenge in the sparse rewards associated with actions during the training process. The occasions of acquiring effective rewards are extremely rare, which greatly reduces the efficiency of action learning. To address these challenges, we introduce the integration of DQN and HER. The mapping from states to actions is learned through DQN. The HER strategy is employed to reuse failed experiences. This integration enables the original experiences that did not reach the predefined goals to be reevaluated by introducing new objectives, thus enhancing the learning outcomes. That is, if the robotic arm fails to achieve the original goal g, an alternative goal g' is selected. The experience quintuple (s_t, a_t, r_t, s_{t+1}, g) generated by the action is replaced by (s_t, a_t, r'_t, s'_{t+1}, g'), and is stored in the replay buffer. The reward function r'_t is recalculated to reflect the new targets g'. The state s'_{t+1} corresponds to the successful state of target g'. In this way, the robotic arm's action is reinterpreted as a beneficial attempt to reach the new goal g'. This approach expands the sample experiences in the replay buffer, accelerating the efficiency of the robotic arm's motion training. The process is iterative, with the robotic arm learning to maximise the objective function $Q_\pi(s,a)$ by constantly interacting with the environment and updating its strategy.

4. DRA-DenseNet

In this section, we integrate an attention mechanism into the DenseNet [23] framework to propose the DRA-DenseNet. It can extract features from heightmaps to enhance the robotic arm's grasping abilities.

As illustrated in Fig.2, the DRA-DenseNet is composed of an encoder segment and a decoder segment. In the encoder segment, the pre-trained DenseNet-121 network is employed to extract features from the heightmap. Initially, the RGB image, the depth image, and the target mask map are preprocessed to standardise their dimensions to $1\times3\times H\times W$. These images are then individually fed into the pre-trained DenseNet-121 model to extract distinct feature maps, each with dimensions of $1\times C\times20\times20$. To facilitate subsequent feature processing, a feature

concatenation strategy is employed to achieve the integration of feature maps from the three types of images. By concatenating the channel dimensions, a new combined feature map with dimensions of 1×3*C*×20×20 is obtained. The method not only preserves the essential information from each image but also provides a comprehensive representation of feature information, thereby enhancing the utilisation of diverse image features.

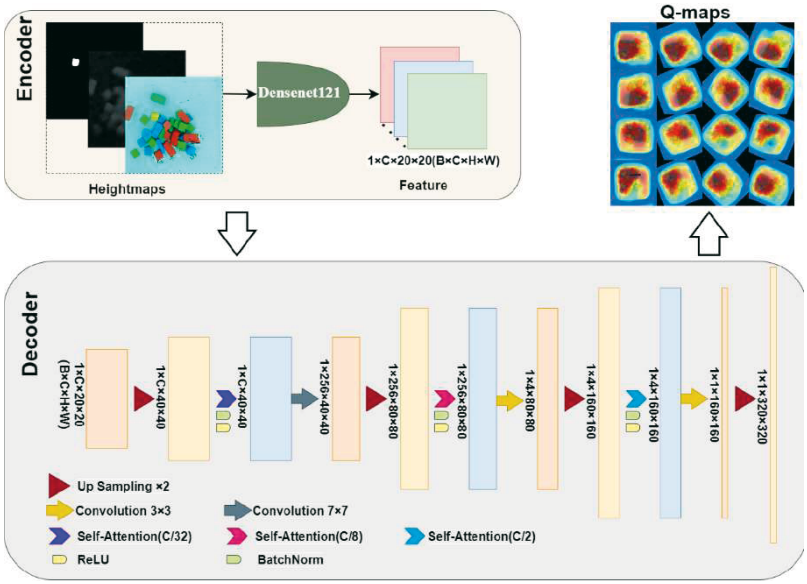


Fig. 2 The DRA-DenseNet structure

In the decoder segment, a symmetrical upsampling architecture is employed. The initial module employs bilinear interpolation for 1/8 upsampling on the original image. Then, a series of processing steps are performed. These modules include a self-attention layer, a 7×7 convolutional layer, a Batch Normalisation(BatchNorm) layer, and a Rectified Linear Unit (ReLU) activation function. The subsequent module performs 1/4 upsampling, with further processing conducted through a 3×3 convolutional layer, BatchNorm, and ReLU. The third module then upsamples the image to half its original scale and processes it to generate a single-channel output via a 3×3 convolutional layer. Finally, additional upsampling operations are performed to restore the image to its original resolution.

We introduce a channel self-attention module in the decoder segment. The module is designed to enhance the model's recognition and emphasis on key feature channels, thereby improving its performance in expressing image features. As shown in Fig. 3, the feature maps are input into the adaptive average pooling layer and downsampled to a smaller scale, resulting in a 1×1 scale. The operation aims to extract global feature information. Then, these global features are fed into a fully connected attention module, which is composed of two linear transformation layers and a ReLU activation function for further processing:

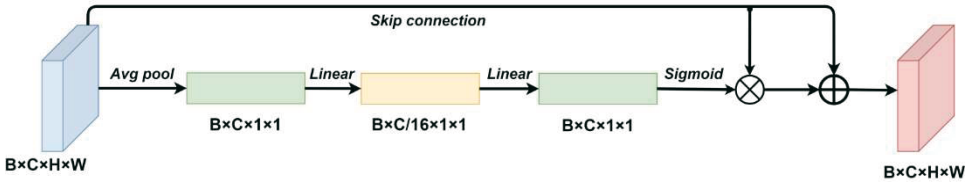


Fig. 3 Structure of the channel self-attention module

In this process, the first linear transformation layer reduces the number of feature channels to c/r_d of the original number, where c represents the number of feature channels and r_d is the ratio of dimension reduction. After the features pass through the ReLU activation function, the second linear transformation layer restores the number of channels to the original size. Then,

the channel attention weights, obtained by applying the Sigmoid function, are element-wise multiplied with the original feature maps, and the final weighted feature maps are formed through the combination of residual connections. This residual learning mechanism based on channel attention can enhance the decoder's emphasis on key channel information. It enables the model to identify and utilise important features more effectively, thereby exhibiting better generalisation when processing different data.

To further enhance the decoder's performance in image reconstruction, we integrate a quartic bilinear upsampling technique into the decoder's architecture. The technique ensures a smoother transition from low-resolution to high-resolution feature maps, effectively reducing information loss during the upsampling process. Moreover, the multi-stage upsampling approach aligns with the classical encoder-decoder framework used in deep learning for image segmentation. This integration can improve the network's ability to recover fine details and reconstruct features. The search and grasp networks are all built upon the foundation of the DRA-DenseNet.

5. Search strategy

To address the issue of targets being heavily obscured by obstacles, we propose a DRA-DenseNet-based search strategy. The strategy can effectively capture the comprehensive features of the image and integrate the past action experiences to improve the search efficiency.

5.1 Framework of the search strategy

The framework of the search strategy is illustrated in Fig. 4. It is designed to generate a search heatmap, which guides the robot's searching actions in cluttered environments.



Fig. 4 The framework of the search strategy

As depicted in Fig. 4, the search network ϕ_s at time t receives inputs that include the depth heightmap D_h^t and the search experience mask S_m^t. Furthermore, we integrate two depth heightmaps (2×224×224) with a search experience mask (1×224×224). This integration results in a data structure (3×224×224), enhancing the integration of search experiences and depth information. This fusion facilitates feature extraction by the search network's encoder. The fused images are then processed by the search network ϕ_s to generate 16 Q-maps, each with a size of 224×224 in sixteen distinct rotational directions. The search network ϕ_s is constructed based on the DRA-DenseNet framework, and the state s_t is described by $\phi_s(S_m^t, D_h^t)$ at time t. The Q-value of each pixel in each Q-map represents the expected future reward for performing the corresponding action. As described above, a search strategy based on DQN and HER, which is described in Section 3, can be employed to learn the optimal control policies, so that the robotic arm can perform the search based on the search Q-map.

We select pushing as the primary action in the search strategy. After each interaction, the search experience mask S_m^t is updated accordingly. This updated mask guides future search actions, enabling the robotic arm to more effectively explore the environment and locate targets. Prior to each search, the search experience mask is reset to an all-ones matrix. The push path that did not lead to a successful search is marked as the centre of a low peak after each search. ς is the pixel

value at the centre of the low peak, and β is the slope of the low peak. Additionally, points within a certain proximity to the push path undergo blurring treatment, which can be expressed as:

$$v(x,y)=\frac{\beta((x_e-x_s)(y-y_s)-(y_e-y_s)(x-x_s))}{r\sqrt{(x_e-x_s)^2+(y_e-y_s)^2}}+\varsigma\,, \tag{5}$$

where r is a hyperparameter, representing the operational range of the action. (x_s,y_s) and (x_e,y_e) denote the start and end points of the search action trajectory, respectively. Algorithm 1 details the generation of the search experience mask S_m^{t+1}:

Algorithm 1 Generating search experience mask S_m^{t+1}

Require: Start point (x_s,y_s) and end point (x_e,y_e). Previous search mask S_m^t

Ensure: The next search experience mask S_m^{t+1}

1: Initialisation: $S_m^{t+1} \leftarrow$ All-ones mask; % Set the value of all pixels to 1, and the size of S_m^{t+1} is the same as that of the depth heightmap;

2: **while** (x, y) in S_m^{t+1} **do**

3: $S_m^{t+1}(x,y) \leftarrow v(x,y)$; % Update the value of the pixel at point (x,y) according to the algorithm (3);

4: **end while**

5: $S_m^{t+1} \leftarrow S_m^{t+1} \cdot S_m^t$; % Generate the search experience mask by synthesising the previous search experience;

5.2 Search reward function

To enhance the convergence of the search network, the search reward is designed. After the robotic arm performs the search action, it receives different rewards based on the observed environmental changes. The search reward R_s is described as:

$$R_s(s_t,a,s_{t+1})=\begin{cases}\kappa\cdot\delta_m & \text{state } s_t \text{ is transformed to state } s_{t+1}\\ \delta_m & \text{target is searched}\\ 0 & \text{otherwise}\end{cases}, \tag{6}$$

where δ_m is defined as the value corresponding to the start point of each search action in S_m^t, $\delta_m=S_m^t(x_s,y_s)$. κ is a scaling factor, $0<\kappa<1$. Before each search, S_m^t is initialised to a default state where every pixel has a value 1, and $\delta_m=1$. If the same coordinates are subsequently searched, the pixel value at that location is updated, resulting in a decrease in δ_m. It can effectively diminish the reward for repeated searches at the same location, thereby encouraging the robotic arm to adopt a more effective search strategy.

6. Push-grasp collaborative strategy

The push-grasp collaborative strategy is designed to secure a successful grasp of the target object. To achieve this, accurately assessing the target's available grasping space is crucial, as it enables the seamless integration of pushing and grasping actions. In addition, the coordination of push and grasp actions is essential for improving the grasp efficiency.

6.1 Framework of a collaborative strategy

The framework of a push-grasp collaborative strategy is shown in Fig. 5. An RGB-D camera captures a colour heightmap and a depth heightmap, which provide the necessary environmental information for the robotic arm. The state is represented based on the depth heightmap, the colour heightmap, and the generated target mask heightmap. Additionally, these heightmaps are rotated at 16 different angles and fed into the pushing network, and the grasping network is constructed by a DRA-DenseNet to generate the grasp prediction Q-maps and the push prediction Q-maps, respectively. The target gap mask is generated by using the target mask heightmap and the depth heightmap, and is analysed to determine the presence of viable

grasping spaces around the target in the push-grasp collaborative task. If a viable space is identified, a grasping action is performed; otherwise, a pushing action is executed. Finally, the corresponding action is performed at the pixel coordinate with the largest Q-value of the Q-map. This process is repeated until the object is successfully captured. As described above, a collaborative strategy based on DQN and HER is employed to learn the optimal control policies, so that the robotic arm can perform grasping actions efficiently.

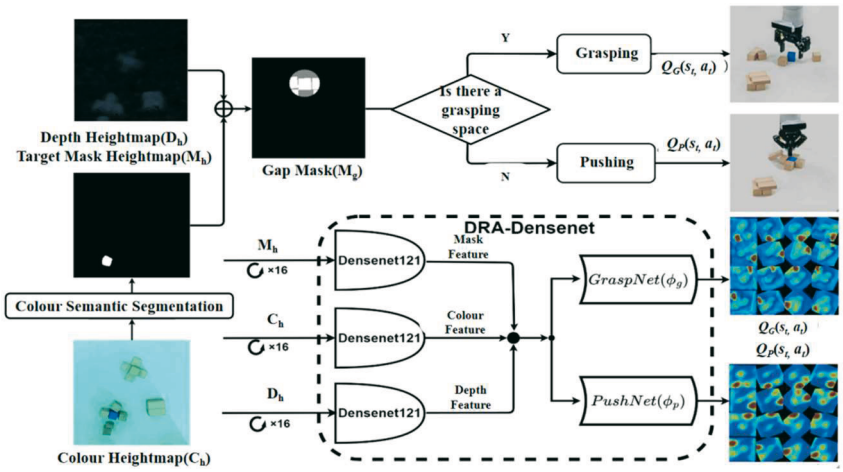
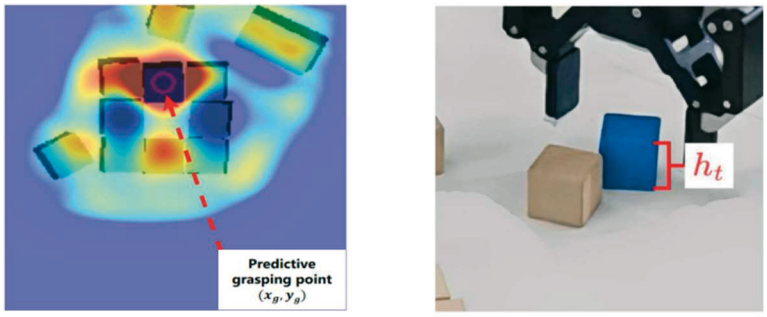


Fig. 5 The framework of the push-grasp collaborative strategy

6.2 Target gap mask

Pushing is implemented if no viable grasping space is present. It is necessary to judge whether the target can be grasped by employing a graph that can accurately describe the surrounding area. We use the depth heightmap and the target mask heightmap as the prior knowledge to construct the target gap mask map. The map promotes the execution of the grasping and pushing collaborative actions.



(a) Predictive grasping point (b) The height of target object

Fig. 6 Judging target graspability

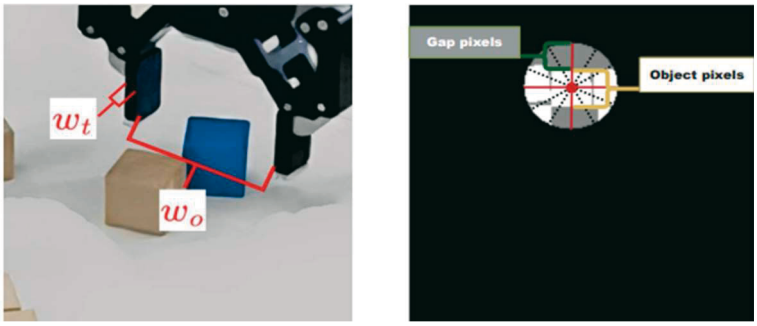
The predicted Q-map allows us to determine the grasping point, denoted as (x_g, y_g), as shown in Fig. 6(a). Each pixel value in the heightmap represents the height of this grasping point above the workspace's reference plane. By integrating the grasping point (x_g, y_g) with the depth information, we can obtain the exact height h_t of the target object, as shown in Fig. 6(b). Leveraging both the grasp point (x_g, y_g) and the height h_t, we can then construct the target mask gap map M_g, as shown in Algorithm 2. The threshold for the grasping height is determined as the product of the object's height h_t and the scaling factor ω, i.e., $h_{th} = \omega \cdot h_t$, $0 < \omega < 1$. Within the workspace, any object exceeding this height threshold is classified as an obstacle. A circle is created with its centre at (x_g, y_g) and a radius of r in the region mask. r is the hyperparameter, which is determined by the opening width of the gripper of the robotic arm. In the workspace, a pixel point in the depth heightmap is considered an obstacle if its value exceeds the height threshold

z_{min}; otherwise, it is deemed not to interfere with the target grasping. This approach enables a clear and efficient representation of the feasible grasping spaces with respect to the target object.

Algorithm 2 Constructing the target gap mask M_g

Require: Predictive grasping Q-value Q_g, depth heightmap D_h, scaling factor ω, a radius r, the minimum threshold on the z-axis z_{min}
Ensure: Target gap mask M_g;
1: $(x_g, y_g) \leftarrow Q_g$; % Obtain the pixel coordinates where the grasp action is performed;
2: $h_t \leftarrow D_h(x_g, y_g)$; % Obtain the height of the target object;
3: $h_{th} = \omega \cdot h_t$; % Set the height threshold of grasping;
4: $O_m \leftarrow$ all zeros mask; % Initialise an all-zero mask to the region mask O_m;
5: $O_m[\mathrm{d} \leq r] \leftarrow 1$;% Set the value of the pixel to 1 if the distance between the pixel and point (x_g, y_g) is less than r
6: $O_d \leftarrow O_m \cdot D_h$; % Construct a grasping region heightmap O_d entered on the grasping point;
7: $M_g \leftarrow O_d$; % Construct the target gap mask;
8: $M_g\,[O_d < z_{min}] \leftarrow 0$; % Set the pixel value of M_g to 0 if the value of a pixel is below the threshold z_{min} in O_d
9: $M_g\,[O_d < h_{th}] \leftarrow 0$; % Set the pixel value of M_g to 0 if the value of a pixel is less than h_{th} in O_d
10: $M_g\,[O_d \geq h_{th}] \leftarrow 1$;% Set the pixel value of M_g to 1 if the value of a pixel is greater than h_{th} in O_d
11: M_g [out of O_m] $\leftarrow -1$; % Set the pixel value of M_g to -1 if the point is outside the O_m
12: return M_g;

Subsequently, we conduct an analysis of gaps in 16 distinct directions to determine the availability of suitable grasping spaces. The judgment of the gripper's grasping action is shown in Fig. 7. The open width w_0 of the gripper during grasping and the width w_t of the gripper's finger end are described in Fig. 7(a). In Fig. 7(b), 16 rays are evenly distributed around the grasping point (x_g, y_g). These rays form eight pairs in opposite directions, creating eight straight lines, each corresponding to a specific rotation angle. By analysing the pixel values at these angles on M_g, the algorithm identifies the gaps and the portion of pixels occupied by the target object across various rotation angles. The decision on whether a gap can be grasped at a particular angle is based on a comparison between the gap size and the target, considering the geometric characteristics of the gripper. If the gap on either side of the target is larger than the width w_t, and the target is smaller than the open width w_0 in the corresponding direction, the algorithm identifies this direction as suitable for grasping. Algorithm 3 evaluates the graspability of the target and then selects the appropriate action accordingly.



(a) Grasping action (b) Target gap mask analysis

Fig. 7 Grasp action judging of the gripper

Algorithm 3 Pushing and grasping judgment strategy

Require: Target gap mask M_g; predictive grasping point (x_g, y_g); width w_t; open width w_0;
Ensure: Pushing or Grasping;
1: Gap=0; Occup=0; % Initialise the number of pixels of the grasping gap and the target object
2: **for** $i = 1, \cdots, 16$ **do**
3: $gap_i = 0$; $occup_i = 0$; % Initialise the number of pixels of the grasping gap and the target object in the given direction i
4: $x_i \leftarrow x_g + (w_0/2)\cos(\pi i/8)$, $y_i \leftarrow y_g + (w_0/2)\sin(\pi i/8)$; % Set the grasping position on the given direction i

5: **for** pixel in traceLine(M_g, (x_g, y_g), (x_i, y_i)) **do** % Determine the number of pixels of the grasping gap and the target object in the given direction i
6: **if** pixel← Ø **then**
7: $gap_i = gap_i + 1$
8: **else**
9: $occup_i = occup_i + 1$
10: **end if**
11 **end for**
12: **end for**
13: **for** $j \leftarrow 1,\ldots, 8$ **do** % Evaluate and judge actions
14: **if** $occup_j \leq w_0$ and $gap_j \geq w_t$ and $gap_{j+8} \geq w_t$ **then**
15: robotic arm can grasp at angle $\pi j/8$
16: **else**
17: robotic arm can push at angle $\pi j/8$
18: **end if**
19: **end for**

6.3 Push and grasp reward functions

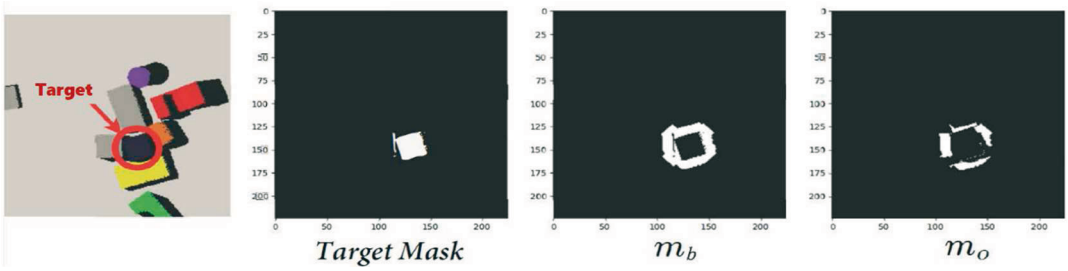


Fig. 8 Construction of target boundary masks

The push rewards are assigned based on the state information observed both before and after the push action. After completing the grasp action, if the target is successfully captured, the reward R_g for the grasp action is set to 1; otherwise, $R_g = -0.1$. To describe the grasp rewards more precisely, the target boundary map m_b is generated utilising the binary target mask map, as shown in Fig. 8. In this map, the white region denotes the target area with a pixel value of 1, while the black region corresponds to a pixel value of 0. Next, by combining the map m_b with the depth heightmap, pixels within the target boundary that exceed the workspace height are designated as 1, and those that are less than or equal to the workspace height are set to 0. Through this process, the obstacle boundary map m_0 is established. The reward R_p for the push action is defined as

$$R_p = \begin{cases} 0.25 & 0 < \Delta o_b < 0.1 \\ 1 & \Delta o_b \geq 0.1 \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

where Δo_b is defined as the difference between the target boundary occupancy rate o_b^t at the pre-action stage, and the target boundary occupancy rate o_b^{t+1} at the post-action stage, e.g., $\Delta o_b = o_b^{t+1} - o_b^t$, $o_b^t = n_0/n_b$. Here, n_0 represents the sum of pixel values within the map m_0, $n_0 = \sum_{m_0} p_v$, where p_v is the pixel value, and n_b represents the sum of pixel values within the map m_b, $n_b = \sum_{m_b} p_v$. If o_b^t is larger, it means that there are more obstacles around the target at the pre-action stage. If $0<\Delta o_b<0.1$, it means that there are fewer obstacles around the target, which is advantageous for grasping, leading to a positive reward. If $\Delta o_b \geq 0.1$, it indicates that there is a significant change in the environment before and after the push action, facilitating future grasping and warranting a higher reward. This reward guides the optimisation of the action strategies, accelerating the training speed of the push actions.

7. Experiments

This section presents a series of simulations and experiments to evaluate the grasp efficiency of the robotic arm. CoppeliaSim is utilised for analysing the grasping strategy of the robotic arm [24]. It is a physics-based simulation environment that integrates the Bullet 2.8.3 physics engine to provide accurate dynamic simulations. The integration of the control architecture and general computing modules can provide powerful support for system modelling, simulation and analysis. The simulation environment is shown in Figure 9. To simulate the visual feedback in the real environment, we utilise the built-in image sensor of CoppeliaSim and OpenGL technology to render the image data. It ensures that the visually guided grasping process of the robotic arm closely resembles the actual operations.

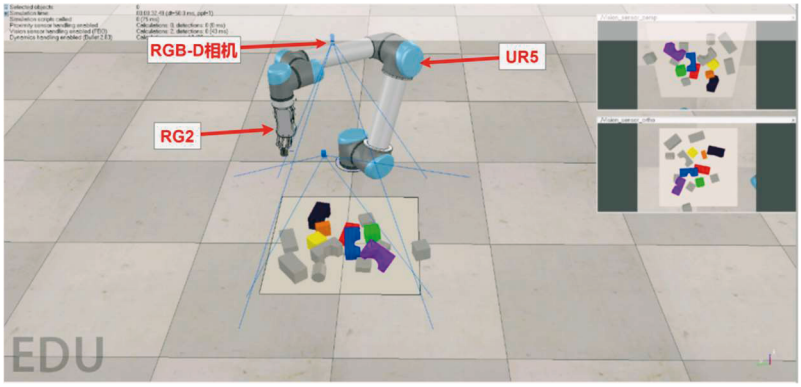


Fig. 9 Simulation environment

In the simulation, the gripper of the robotic arm remains oriented downward throughout, adjusting only the wrist joint according to the position and orientation of the target object during movement. This approach simplifies the complexity of the control algorithm, allowing us to focus on optimising the grasping strategy. We use the AdamW optimiser to train the two networks. In these networks, the fixed learning rate is set to 0.0001, the weight attenuation coefficient is 2E-5, the momentum is 0.95, and the reward discount factor $\kappa = 0.25$. The centre pixel value of the low peak is set to 0.8, and the slope of the low peak is 0.2. The search range r is determined based on the size of the gripper, $r = 12$. The scaling factor ω in the grasping height is 0.7. The ε-greedy algorithm is implemented to guide the training process. Initially, ε is set to 0.5 to balance exploration and exploitation. As training progresses, ε is gradually reduced to 0.1. This decrement strategy aims to gradually reduce the behaviour of random exploration, while increasing the decision frequency based on existing knowledge.

7.1 Simulation of the search strategy

In Fig. 10, we present four training scenarios in which the targets are obscured and invisible. During the training process, there is collision uncertainty between the robotic arm's actions and the objects. This setup allows a limited initial scene to generate a rich set of training data through numerous random movements of the robotic arm. At the beginning of each training iteration, a randomly selected training case is generated on the workbench to simulate the dynamics and unpredictability of the environment, thereby enhancing the generalisation and adaptability of the model.

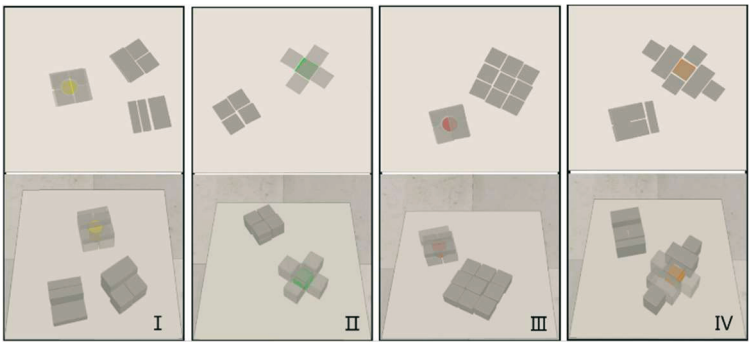


Fig. 10 Search training scenarios

The search network undergoes 800 iterations within these scenarios. During each training round, the robotic arm executes a search action, and the vision system obtains an image of the workspace. The system utilises colour segmentation to determine whether the target colour block appears within the field of view and then verifies whether the target is found. If the target is found, the search action is successful and a corresponding reward is given. If the robotic arm performs five invalid actions, that is, if the position of the object in the workspace remains unchanged, the current training round is terminated. The workspace is updated, and a new round of training commences.

This paper introduces two sets of ablation experiments to prove that the proposed search experience mask can effectively enhance the success rate of target search and improve the generalisation ability in a cluttered environment. In the first set of experiments, the search strategy used only depth (OD) for predicting search actions. In the second set, depth with an experience mask was used as the foundation of search action prediction. To ensure a fair comparison, both sets of experiments were trained in the same environment, with a scenario randomly selected from four preset training scenarios at the beginning of each iteration. We limited the number of searches to a maximum of 10 to prevent useless searches caused by targets falling outside the workspace. Each method executed 800 rounds of search training. Figure 11 shows the variation in the number of actions required to successfully execute two different search strategies during training.

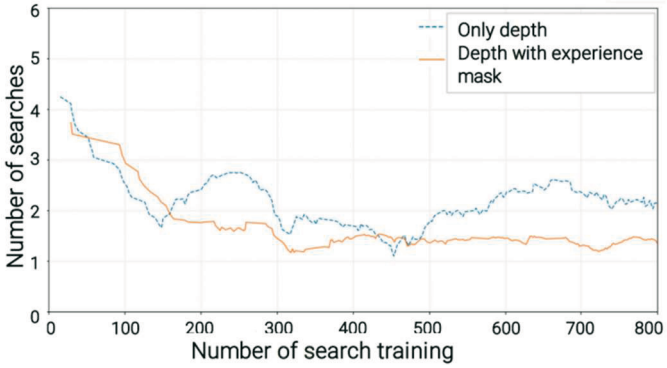


Fig. 11 Search training results

In Fig. 11, the result indicates that the OD method can quickly learn and improve its performance initially, but the rate of improvement slows down as training progresses. In contrast, the search method based on depth and experience mask has a slow convergence at the initial stage. However, its performance steadily improves over the training cycle. The number of searches for the strategy quickly stabilises at around 1.5. This trend suggests that the strategy has a shorter convergence time and better long-term performance stability.

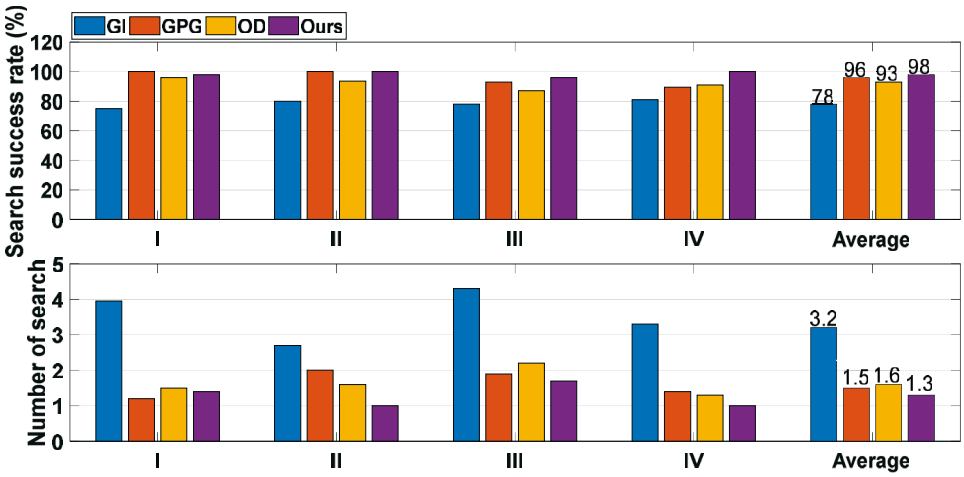


Fig. 12 Experimental results under search training scenarios in Fig.10

We performed comparisons between our method and the GI model [10], the GPG model [11], and the OD model in Figure 12. Each test scenario involved 50 rounds of independent tests. A test is considered successful if the target is found within the maximum number of times. The required number of actions was recorded. Figure 12 shows the experimental results. The proposed method shows certain advantages in search learning ability, search success rate, and stability.

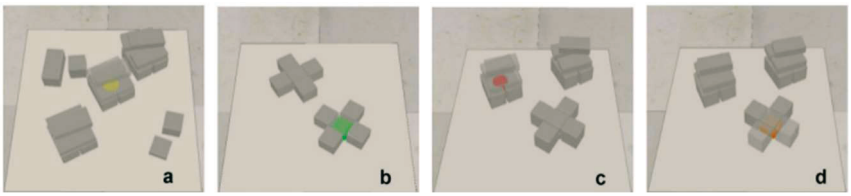


Fig. 13 Search testing scenarios

To further validate our search model, four testing scenarios were designed for search evaluation, as shown in Fig. 13. Fig. 14 shows the performance comparison. The proposed method achieves a 100% average search success rate across all test examples, which is better than the 91% success rate of the GI model and the 99% success rate of the OD model. It shows that the proposed method has good stability and high performance in various complex environments. Moreover, in terms of the number of actions, the average number of actions required to achieve a successful search is only 1.3 for the proposed method, 1.4 for the GPG model, and 2.7 for the GI model. The proposed method requires fewer actions across all test cases. The model thus shows significant improvement in action efficiency.

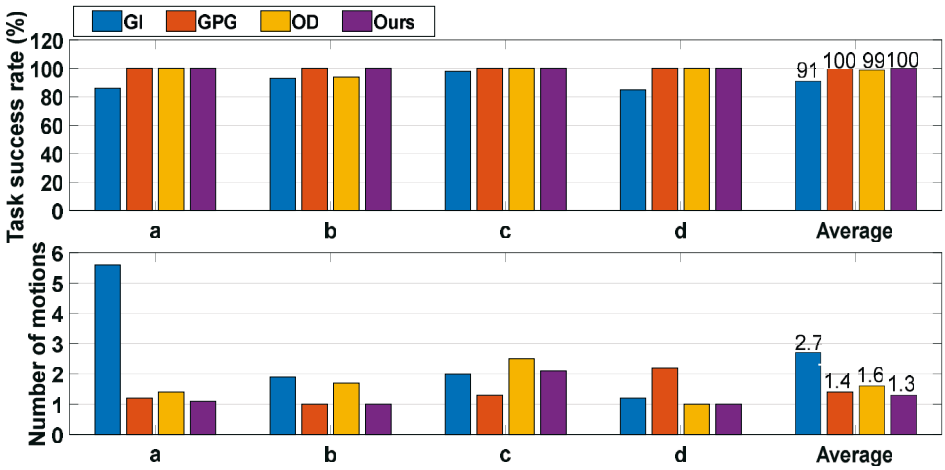


Fig. 14 Experimental results under search testing scenarios in Fig. 13

7.2 Simulation of push-grasp collaborative strategy

In this strategy, the robotic arm learns accurate pushing and grasping actions through self-supervised interaction with the environment. The two networks were trained simultaneously in the collaborative strategy. As shown in Fig.15, the training process was divided into two stages. During the initial training stage (Stage 1), which comprised the first 1000 rounds, the simulation environment randomly generated 10 blocks for each grasping task: seven coloured target blocks and three grey obstacle blocks. In Stage 2, starting from round 1001, the number of obstacle blocks increased to eight, totalling 15 blocks to enhance training complexity. Moreover, the grasping network was trained separately for the first 300 rounds. Starting from round 301, both the grasping and pushing networks were trained together. This method can improve the prediction accuracy of the grasping network before it collaborates with the pushing network. After completing 1000 iterations of the initial phase, the training progressed to the more challenging second stage, which consisted of 3250 training rounds.

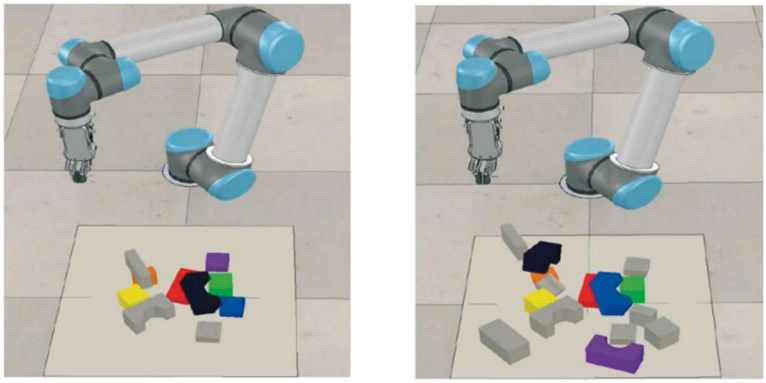


Fig. 15 Push-grasp collaborative training environment

We compared the performance of three different models in grasping tasks: the proposed model, the EPG model [9], and the GI model [10]. Before training, all models were set with the same simulation environment. Throughout the training process, we recorded the grasping success rate of each model and plotted these rates against the training rounds, as shown in Fig. 16. According to the experimental results, the proposed model exhibits superior stability compared to both the GI and EPG models, despite experiencing slight fluctuations during the training process. The grasping success rate of the proposed model reaches nearly 90% after 2500 iterations and maintains this level throughout subsequent training. In contrast, the EPG model shows some fluctuation but generally maintains a success rate of around 80%. The GI model shows limited improvement in the grasping success rate during training, and finally stabilises at about 70%. After comprehensive analysis, the proposed model outperforms both the EPG and GI models in terms of success rate and stability, indicating its strong adaptability to these tasks.



Fig. 16 Training performance

To verify the effectiveness of the push-grasp collaborative strategy, our model was evaluated using the test cases of reference [10], as shown in Figure 17. Target grasping experiments in various scenarios were conducted to verify whether the robotic arm had learned the motion collaboration strategy between pushing and grasping. We compared the success rate of the EPG model [9], the GI model [10], and the proposed method during the training process. To evaluate the performance of the push-grasp collaborative strategy, some baseline methods were introduced, including the Mask-VPG model[8] and the GPG model [25].

A task is considered successful if it is completed within 10 movements. Each test case is conducted for 50 rounds. Before each test, the simulation environment places the block in a specified position. If the total number of push and grasp actions exceeds 10 times or a grasp is successful, the next round of testing begins. The experimental results are shown in Fig. 18 which indicates that five different methods can perform the push-grasp collaborative task in a cluttered environment. From the overall performance perspective, the proposed method shows certain advantages in grasping accuracy, action coordination efficiency and task success rate.

Specifically, the proposed method achieved a task success rate of 97%, compared with 70% for Mask-VPG, 87% for GI, 95% for EPG, and 91% for GPG, respectively. The proposed method performed well in different test cases, especially in cases 2, 3, 4, 5, and 7, where the task success rate reaches 100%. This shows that the model has good stability and generalisation when facing different grasping tasks. Besides, the proposed method achieves an average grasping success rate of 96%, while the performance of other models remains below 90%, which verifies the superior grasping ability of the proposed model. In addition, the proposed method requires an average of 2.4 actions per grasping process. These results indicate that the proposed method not only improves the success rate, but also significantly optimises the action efficiency.

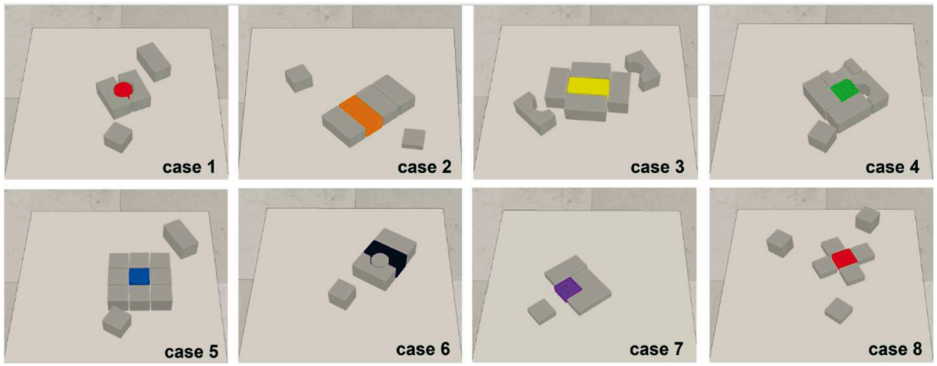


Fig. 17 Testing scenarios of push-grasp coordination tasks

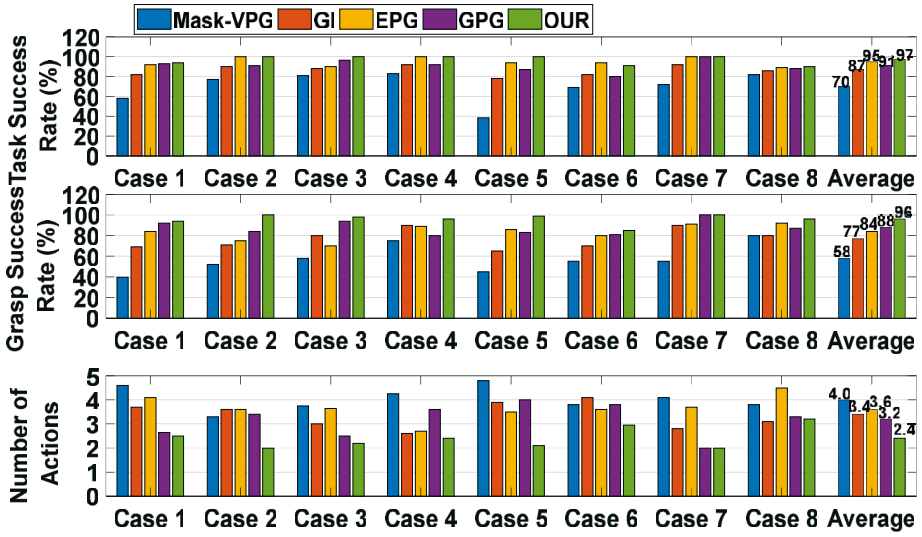


Fig. 18 Performance of push-grasp collaborative strategies

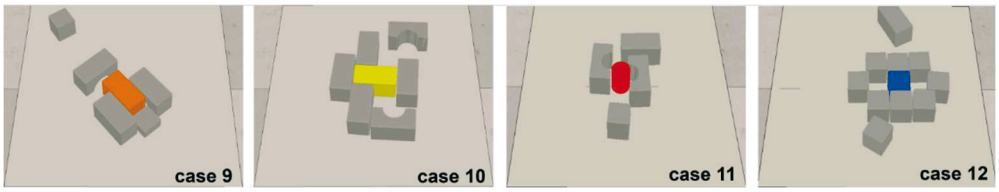


Fig. 19 Special test case of action collaborative tasks

To further verify the proposed collaborative strategy and action prediction network model, we constructed four special test cases, as shown in Fig.19. In these test cases, the object can be grasped without the need for pushing. The purpose of these tests is to prove the accuracy of the collaborative strategy in assessing the grasping space and the high performance of the grasping action prediction network. In the experiments, each scenario is executed for 50 rounds, with a maximum of 10 actions per round. If the task exceeds the threshold, it is deemed a failure, and the next round of tests is initiated and the simulation environment regenerates the work scene.

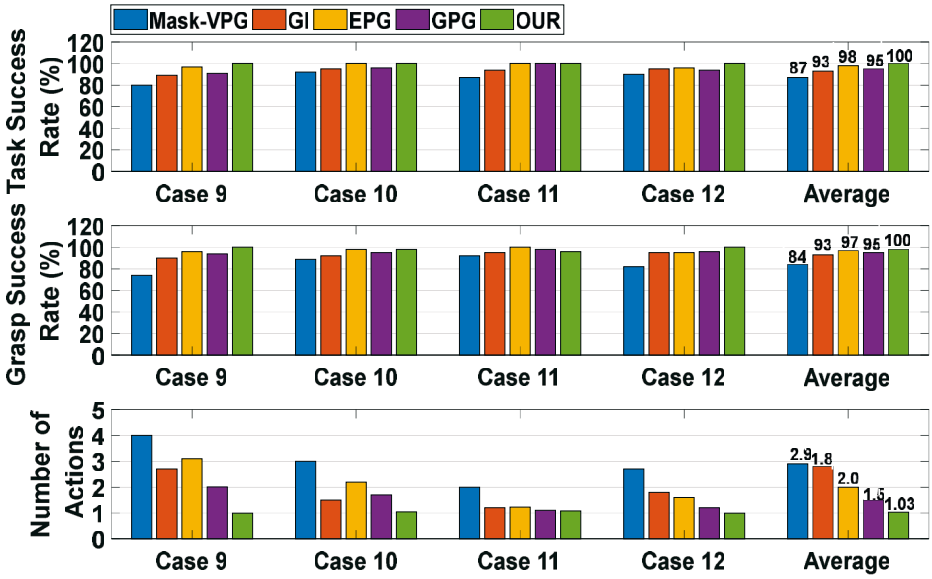


Fig. 20 Experimental results for special test cases

Figure 20 presents the experimental results for each test case. The simulation results demonstrate that the proposed strategy performs well in special test cases, achieving a task success rate of 100% and a grasp success rate of 100%. The performance of the EPG algorithm is close to that of the proposed algorithm in terms of the grasping success rate and the task success rate. However, the EPG algorithm needs to increase the number of actions. The proposed strategy demonstrates superior performance in terms of the number of actions compared to other baseline methods, with an average of 1.03. It indicates that the model successfully completes the task with a single action. These results show that the push-grasp collaborative strategy and the proposed network are effective and can accurately identify and grasp targets in cluttered environments while minimising the number of actions performed.

Fig. 21 illustrates the performance of the proposed method in specific task cases. The first row of the figure displays the grasp simulation map, clearly demonstrating the method's capability to accurately grasp the object in the simulated environment. The second row shows the Q-maps, highlighting the model's ability to accurately predict the target location. The third row presents the target gap mask M_g, providing a clear view of the gaps and obstacles around the target. The experiments show that the method can complete tasks efficiently in dense scenes. Especially in complex test cases, the average capture success rate exceeds 96%, proving the method's generalisation ability and adaptability.

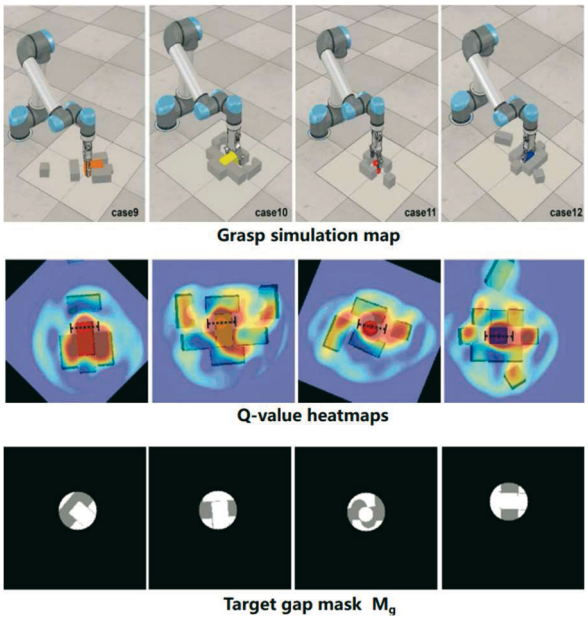


Fig. 21 Simulation for specific task cases

7.3 Real-robot experiments

To verify the performance of the proposed method in practical scenarios, we built a robotic arm experiment platform. The workspace measures 0.448 m×0.448 m. It comprises a Kinova gen3 robotic arm equipped with a Robotiq 2F-85 two-finger gripper, and an Intel RealSense D435i RGB-D camera, as shown in Figure 22. The PC controller is a high-performance computer equipped with an Intel i7-10700 CPU and a NVIDIA GTX 2080Ti GPU. The camera is positioned diagonally above the workspace to avoid any interference with the robotic arm's movements. We obtained the coordinate conversion matrix between the camera and the robotic arm using hand-eye calibration technology. Then, the end-effector reaches the specified position, grasps the object and moves it to the given area, or pushes it a certain distance. The motion model trained in the simulation environment is transferred to the practical robotic arm system, and the grasp experiments for invisible targets are carried out.



Fig. 22 Robotic arm system in real environments

During the experiment, the camera captures images of the workspace and transmits them to the PC controller. These images are processed to generate an action heightmap, which outputs the action execution coordinate information to the robotic arm. The robotic arm interacts with the environment, and the performance of these actions is fed back to the PC controller, which

records the interaction details. The search model and the push-grasp coordination model trained in simulation are transferred to the real environment.

7.3.1 Hidden target object grasping

Five test cases were constructed in the real environment, as shown in Figure 23. In each test case, the coloured target object is hidden behind an obstacle block. The robotic arm needs to push the obstacle block to search for the target using the proposed search strategy. After locating the target, the robotic arm employs a coordinated strategy to grasp the target. In the experiments, the proposed method was tested along with three other methods: Mask-VPG [8], GI [10] and GPG [25]. Each test case consists of 10 rounds, with a maximum of 15 movements permitted per round. If the robotic arm exceeds 15 actions without successfully capturing the hidden target, it is deemed a failure, and the next round of experiment begins.

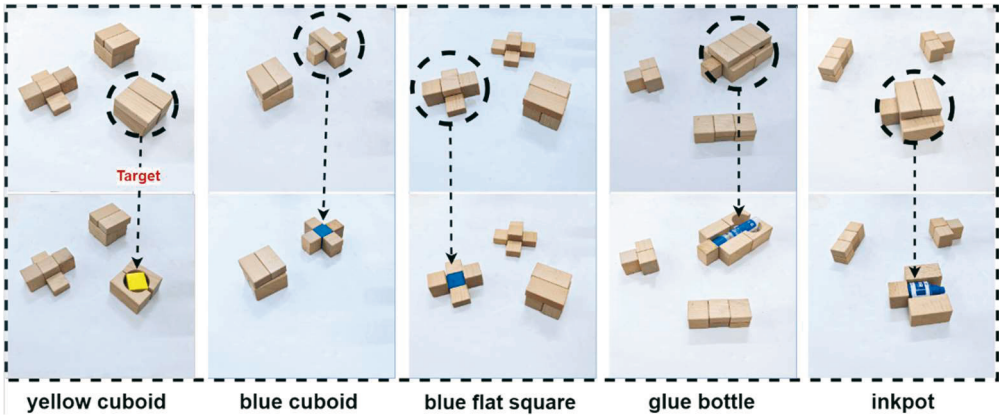


Fig. 23 Test cases in a real environment

The experimental results are presented in Table 1. The proposed method shows significant advantages in both the task success rate and the number of actions required. Specifically, the proposed method achieves a task success rate of 92%, which indicates that the robotic arm can successfully complete the grasping task in most cases. In addition, it requires only 6.9 actions on average, meaning that the proposed method not only enhances the grasping success rate but also improves the action efficiency.

Table 1 Experimental results

	Task success rate	Average number of actions
MASK-VPG	66%	12.2
GI	84%	10.4
GPG	90%	7.3
Ours	92%	6.9

7.4 Irregular target object grasping

To verify the generalisation of the proposed method, we conducted grasping experiments using irregular objects from daily life. The experiment is divided into three scenarios, with five, seven, and nine irregular objects respectively. Some scenarios are shown in Fig. 24. Each scenario is tested in 10 rounds, with objects randomly placed in the workspace. Each round has a maximum limit of 15 movements. If all objects in the workspace are successfully grasped, the test for that round is finished. If the number of movements exceeds 15, the round ends and the next round begins. Fig. 25 shows the heatmap of irregular objects.

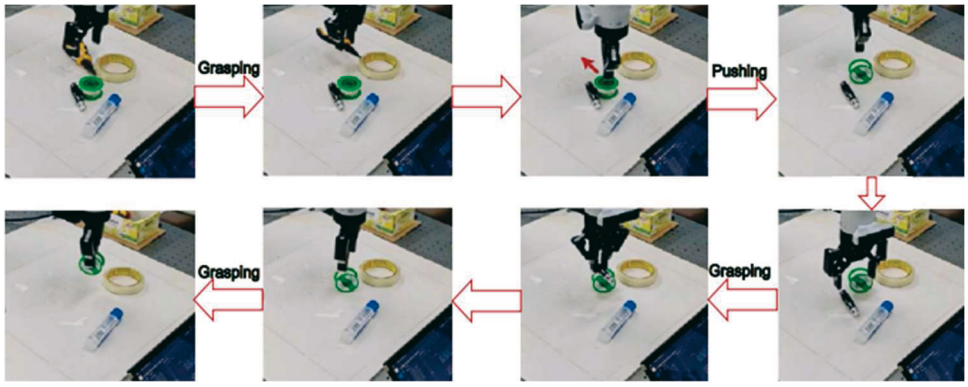


Fig. 24 Grasping of irregular objects

The experimental results for grasping irregular objects are shown in Table 2. As the number of objects increases in the experiment, the average grasping success rate decreases. The average number of movements required per block also increases with the growing number of blocks. This shows that as the number of objects increases, the robot requires more attempts to locate and grasp the target object. The complexity of grasping increases because of the mutual occlusion between the objects and the diversity of their spatial positions.

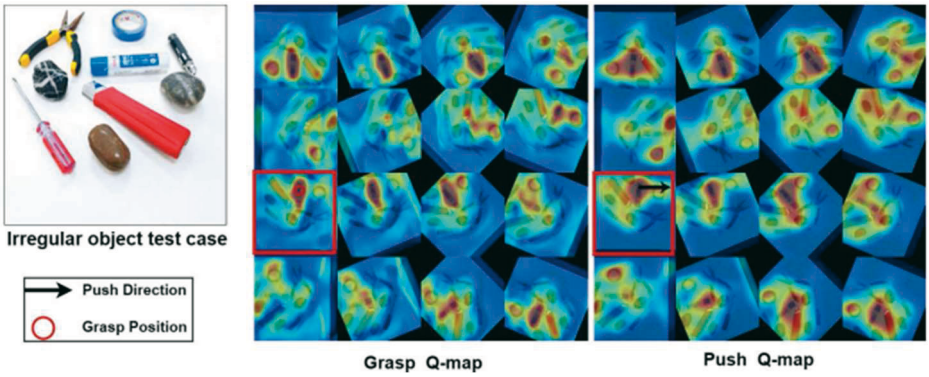


Fig. 25 Heatmap of irregular object grasping

Table 2 Real robot experimental results

Number of objects		TGSR(%)	Average number of actions
5		78.1	6.4
7		73.6	9.5
9		67.6	13.3

8. Conclusion

This paper presents a DRL-based approach for robotic arm grasping in cluttered environments. A search strategy based on the DRA-DenseNet and the experiment mask is proposed to solve the search problem of invisible targets. Additionally, a pixel-level push-grasp action collaboration strategy based on the target gap mask is introduced to improve the success rate of grasping tasks. The experimental results show that this method reduces the number of actions required during the search process, and achieves high grasp success rates in both simulated and real environments. However, in the real environment, the colour images captured by the camera are prone to distortion, making it difficult to generate the target mask map using semantic segmentation. Therefore, future research could focus on exploring semantic segmentation techniques to improve the push-grasp collaborative strategy. However, in this strategy, the gripper of the robotic arm is positioned perpendicular to the workspace and can only perform top-down actions. It is necessary to explore the motion capabilities of the robotic arm with more degrees of freedom to achieve multi-angle grasping tasks.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 62273109, the Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515010168, the Guangdong General University Innovation Project under Grant 2024KTSCX181, the Key Special Foundation for General Universities in Guangdong Province under Grant 2022ZDZX1018, the Maoming Science and Technology Plan Foundation under Grant 2024036, and Projects of Talents Recruitment of GDUPT under Grant XJ2023000402.

REFERENCES

[1] Fu, X.; Liu, Y.; Wang, Z. Active Learning-Based Grasp for Accurate Industrial Manipulation, *IEEE transactions on automation science and engineering* **2019**, 16(4), 1610-1618. https://doi.org/10.1109/TASE.2019.2897791

[2] Baig, R. U. Identifying and Prioritising Future Robot Control Research with Multi-Criteria Decision-Making. *Transactions of FAMENA*, **2020**, 44(3): 23-34. https://doi.org/10.21278/TOF.44302

[3] Liu, W.; Hu, J.; Wang, W. A Novel Camera Fusion Method Based on Switching Scheme and Occlusion-Aware Object Detection for Real-Time Robotic Grasping, *Journal of Intelligent & Robotic Systems* **2020**, 100(3), 791–808. https://doi.org/10.1007/s10846-020-01236-7

[4] Tan, B. H.; Tang, H.; Yan, R. Flexible and robust robotic arm design and skill learning by using recurrent neural networks, *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* **2014**, 522-529. https://doi.org/10.1109/IROS.2014.6942609

[5] Novkovic, T.; Pautrat, R.; Furrer, F.; Breyer, M.; Siegwart, R.; Nieto, J. Object finding in cluttered scenes using interactive perception, *2020 IEEE International Conference on Robotics and Automation (ICRA)* **2020**, 8338-8344. https://doi.org/10.1109/ICRA40945.2020.9197101

[6] Fang, K.; Zhu, Y.; Garg, A.; Kurenkov, A.; Mehta, V.; Fei-Fei, L.; Savarese, S. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research* **2020**, 39(2-3): 202-216. https://doi.org/10.1177/0278364919872545

[7] Newbury, R.; Gu, M.; Chumbley, L.; Mousavian, A.; Eppner, C.; Leitner, J.; Cosgun, A. Deep learning approaches to grasp synthesis: A review, *IEEE Transactions on Robotics* **2023**, 39(5), 3994-4015. https://doi.org/10.1109/TRO.2023.3280597

[8] Zeng, A.; Song, S.; Welker, S.; Lee, J.; Rodriguez, A.; Funkhouser, T. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning, *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* **2018**, 4238-4245. https://doi.org/10.1109/IROS.2018.8593986

[9] Xu, K.; Yu, H.; Lai, Q.; Wang, Y.; Xiong, R. Efficient learning of goal-oriented push-grasping synergy in clutter, *IEEE Robotics and Automation Letters* **2021**, 6(4), 6337-6344. https://doi.org/10.1109/LRA.2021.3092640

[10] Yang, Y.; Liang, H.; Choi, C. A deep learning approach to grasping the invisible, *IEEE Robotics and Automation Letters* **2020**, 5(2), 2232-2239. https://doi.org/10.1109/LRA.2020.2970622

[11] Zuo, G.; Tong, J.; Wang, Z.; Gong, D. A graph-based deep reinforcement learning approach to grasping fully occluded objects, *Cognitive Computation* **2023**, 15(1), 36-49. https://doi.org/10.1007/s12559-022-10047-x

[12] Kiatos, M.; Sarantopoulos, I.; Koutras, L.; Malassiotis, S.; Doulgeri, Z. Learning push-grasping in dense clutter, *IEEE Robotics and Automation Letters* **2022**, 7(4), 8783-8790. https://doi.org/10.1109/LRA.2022.3188437

[13] Tekden, A. E.; Erdem, A.; Erdem, E.; Asfour, T.; Ugur, E. Object and relation centric representations for push effect prediction, *Robotics and Autonomous Systems* **2024**, 174, 1-17. https://doi.org/10.1016/j.robot.2024.104632

[14] Al-Shanoon, A.; Lang, H. Learn to grasp unknown-adjacent objects for sequential robotic manipulation, Journal of Intelligent & Robotic Systems **2022**, 105(4), 1-14. https://doi.org/10.1007/s10846-022-01702-4

[15] Yu, S.; Zhai, D. H.; Xia, Y. A novel robotic pushing and grasping method based on vision transformer and convolution, *IEEE Transactions on Neural Networks and Learning Systems* **2023**, 35(8), 1-14. https://doi.org/10.1109/TNNLS.2023.3244186

[16] Cui, Y.; Ooga, J. I.; Ogawa, A.; Matsubara, T. Probabilistic active filtering with gaussian processes for occluded object search in clutter, *Applied Intelligence* **2020**, 50(12), 4310-4324. https://doi.org/10.1007/s10489-020-01789-y

[17] Danielczuk, M.; Angelova, A.; Vanhoucke, V.; Goldberg, K. X-ray: Mechanical search for an occluded object by minimizing support of learned occupancy distributions, *2020 IEEE/RSJ International Conference on Intelligent R*obots and Systems (IROS) **2020**, 9577-9584. https://doi.org/10.1109/IROS45743.2020.9340984

[18] Wu, L.; Chen, Y.; Li, Z.; Liu, Z. Efficient push-grasping for multiple target objects in clutter environments, *Frontiers in Neurorobotics* **2023**, 17(12): 1-12. https://doi.org/10.3389/fnbot.2023.1188468

[19] Lin, G.; Liu, F.; Milan, A.; Shen, C.; Reid, I. Refinenet: multi-path refinement networks for dense prediction, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2019**, 42(5), 1228-1242. https://doi.org/10.1109/TPAMI.2019.2893630

[20] Chou, W. X.; Liu, Y. W. An analytical inverse kinematics solution with the avoidance of joint limits, singularity and the simulation of 7-DoF anthropomorphic manipulators. *Transactions of FAMENA*, **2024**, 48(1), 117-132. https://doi.org/10.21278/TOF.461033121

[21] Selvamarilakshmi, D.; Prasanna, J. Experimental study on the effects of crossbeam parameters in a fin ray robotic gripper in the field of horticulture applications. *Transactions of FAMENA*, **2024**, 48(1), 27-45. https://doi.org/10.21278/TOF.481050923

[22] Huang, G.; Liu, Z.; Laurens, V. D. M.; Weinberger, K.Q. Densely Connected Convolutional Networks, *2017 IEEE Conference on Computer Vision and Pattern Recognition* **2017**, 2261-2269. https://doi.org/10.1109/CVPR.2017.243

[23] Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P. Hindsight experience replay, *Advances in neural information processing systems* **2017**, 1-11. https://doi.org/10.48550/arXiv.1707.01495

[24] Rohmer, E.; Singh, S. P.; Freese, M. V-REP: A versatile and scalable robot simulation framework, *2013 IEEE/RSJ international conference on intelligent robots and systems (IROS)* **2013**, 1321-1326. https://doi.org/10.1109/IROS.2013.6696520

[25] Zuo, G.; Tong, J.; Wang, Z.; Gong, D. A graph-based deep reinforcement learning approach to grasping fully occluded objects, *Cognitive Computation* **2023**, 15(1), 36-49. https://doi.org/10.1007/s12559-022-10047-x

Submitted: 22.8.2024

Accepted: 18.02.2025

Xiaoling Xu
Guangdong University of Petrochemical Technology
School of Automation
525000, Maoming, Guangdong, China
xiaolingandyou@163.com

Yiliang Li*
Guangzhou Public Utilities Technician College
510006, Guangzhou, Guangdong, China
Liyiliang@163.com

Gaowei Lei
Guangdong University of Petrochemical Technology
School of Automation
525000, Maoming, Guangdong, China
906474904@qq.com

*Corresponding author:
18319209527@163.com