Supplementary Information of "Cognitive Escape Reinforcement Learning for Complex Decision Making"

Qihui Wu, *Senior Member, IEEE*, Shijin Zhao, Fuhui Zhou, Hao Zhang, Yang Huang, and Kai-Kuang Ma, *Life Fellow, IEEE*

I. SUPPLEMENTARY METHODS

The process of constructing the similarity map We divide a satellite map into a grid map with 21×21 grids, from which we randomly select several different target locations to obtain the corresponding target images. Then, we construct the corresponding similarity map for each target image (that is, the similarity distribution of the target image on the satellite map). Specifically, the pre-trained siamese network extracts the features of the target image o_g and a grid image o_p, where o_p represents the satellite image corresponding to the position p on the grid map. Then, the similarity calculation module computes their feature distance d_p^f (that is, the similarity between the grid image and the target image). Subsequently, we calculate the similarity between each grid image on the navigation map and the target image. Thus, we can obtain the similarity map for the target image. For the CERL and the RL agents, they directly interact to similarity maps with 21×21 grids for training the policy network. To avoid the learned policy network overfitting training scenarios, we enrich the training scenarios with transformations such as transpose, flip and rotation on these similarity maps, as shown in Fig. 4.

Application of CERL on the Robot Operating System platform We present two videos to further demonstrate the superiority of the proposed CERL under the environment of Robot Operating System (ROS), which is a well-known platform. In the first video, CERL, CRL, Baseline DRL, and Levy Walk are

Q. Wu, S. Zhao, F. Zhou, H. Zhang, and Y. Huang are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, P. R. China (e-mail: wuqihui2014@sina.com, shijin_zhao@nuaa.edu.cn, zhoufuhui@ieee.org, haozhangcn@nuaa.edu.cn, yang.huang.ceie@nuaa.edu.cn).

K.-K. Ma is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: ekkma@ntu.edu.sg).

tested in the same area with the same start points and goal points, which are randomly selected within the area with pseudo random numbers. The proposed CERL and CRL can find the goal successfully compared to the Baseline DRL, and Levy Walk. However, CRL is easier to stick in a loop within a small area. In contrast, with the help of the escape module, CERL can escape the local optimal area. In the second video, we compare CERL with CRL in a new map area without pretraining. CERL exceeds CRL in all episodes and succusses in reaching the goal with fewer steps and time. The two videos validate the superiority of the proposed CERL in a more complicated and practical scenario. The supplementary videos can be accessed at the following link: https://www.youtube.com/playlist?list=PLikB0rE_abTddTVyY8yXhbcwi1EzQOOp5

II. SUPPLEMENTARY FIGURES

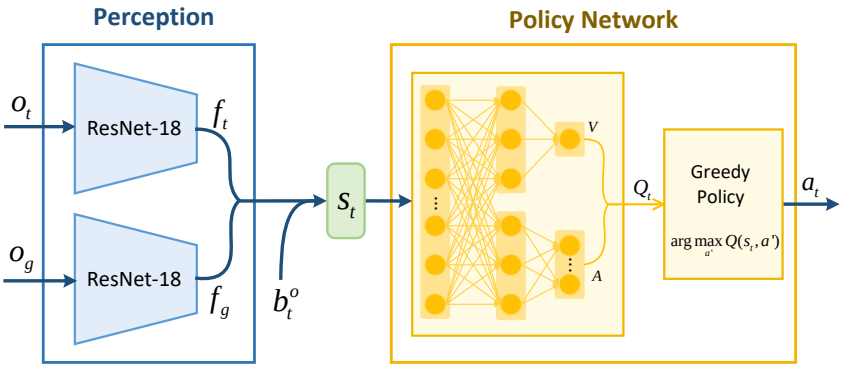


Fig. 1. Model architecture of the Baseline-DRL method. The Baseline-DRL framework consists of a perception module and a policy network. The perception module exploits the same siamese architecture and pre-trained parameters φ as those of the CERL framework and the parameters φ are constant during the training of the policy network. It extracts key features of the observation image o_t and the target image o_g and outputs feature vectors f_t and f_g. Those two features integrate with the boundary information b_t^o to form the state s_t. Then, s_t is input into the policy network, which also has the same network architecture as that of the CERL framework. The policy network (that is, the Q-network) with parameters θ produces a 8-d state action value $Q_t = Q(s_t, a')$. Finally, the action is performed according to the greedy policy $a_t = \arg\max_{\mathbf{a}'} Q(s_t, a')$.

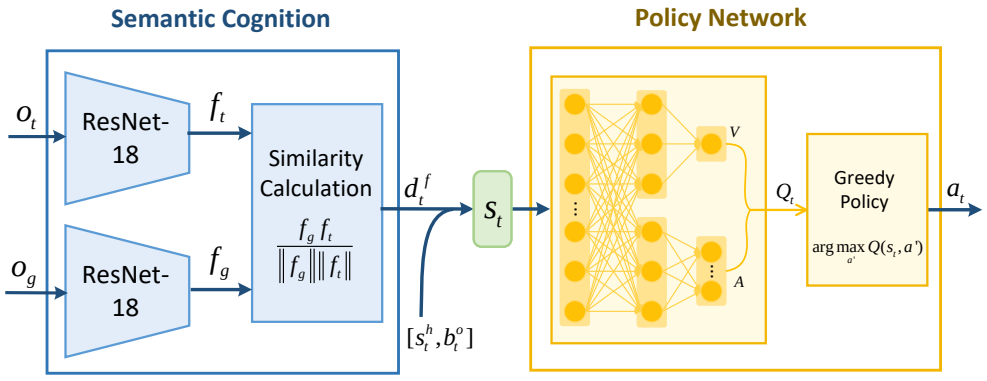
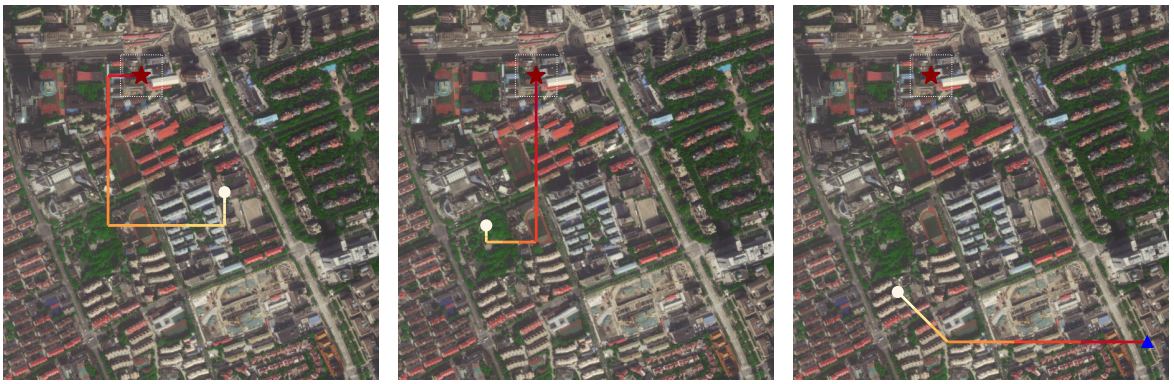


Fig. 2. Model architecture of CRL method. The CRL framework consists of a semantic cognition module and a policy network. The two modules adopt same network architecture as the CERL framework. The semantic cognition module adopts pre-trained parameters φ, which are fixed during the training of the policy network. It first extracts key features of observation image o_t and target image o_g to produce f_t and f_g. Then it calculates Euclidean distance of two feature vectors f_t and f_g to produce semantic information, i.e., the current feature distance d_t^f, which indicates the similarity between observation and target images. d_t^f is combined with the historical trajectory information s_t^h and boundary distance b_t^o as the state s_t. Then, the policy network (that is, the Q-network) receives s_t and produces a 8-d state action value $Q_t = Q(s_t, a')$. Finally, the action is performed according to the greedy policy $a_t = \arg\max_{\mathbf{a}'} Q(s_t, a')$.



a Baseline-DRL navigation **b** Semantic-DRL navigation **c** Semantic-DRL navigation

Fig. 3. Navigation trajectories of different methods. The white circle represents the starting position and the red star represents the goal position. The color of the trajectory corresponds to the navigation time, and the darker colors indicating the final flying position. **a** shows the navigation trajectories of the Baseline-DRL agent that successfully finds the target. **b** shows the navigation trajectories of the Semantic-DRL agent that successfully finds the target. **c** shows the navigation trajectories of the Semantic-DRL agent that fails to find the target. The blue triangle represents the final position of the Semantic-DRL agent at the episode where the agent fails to find the target.

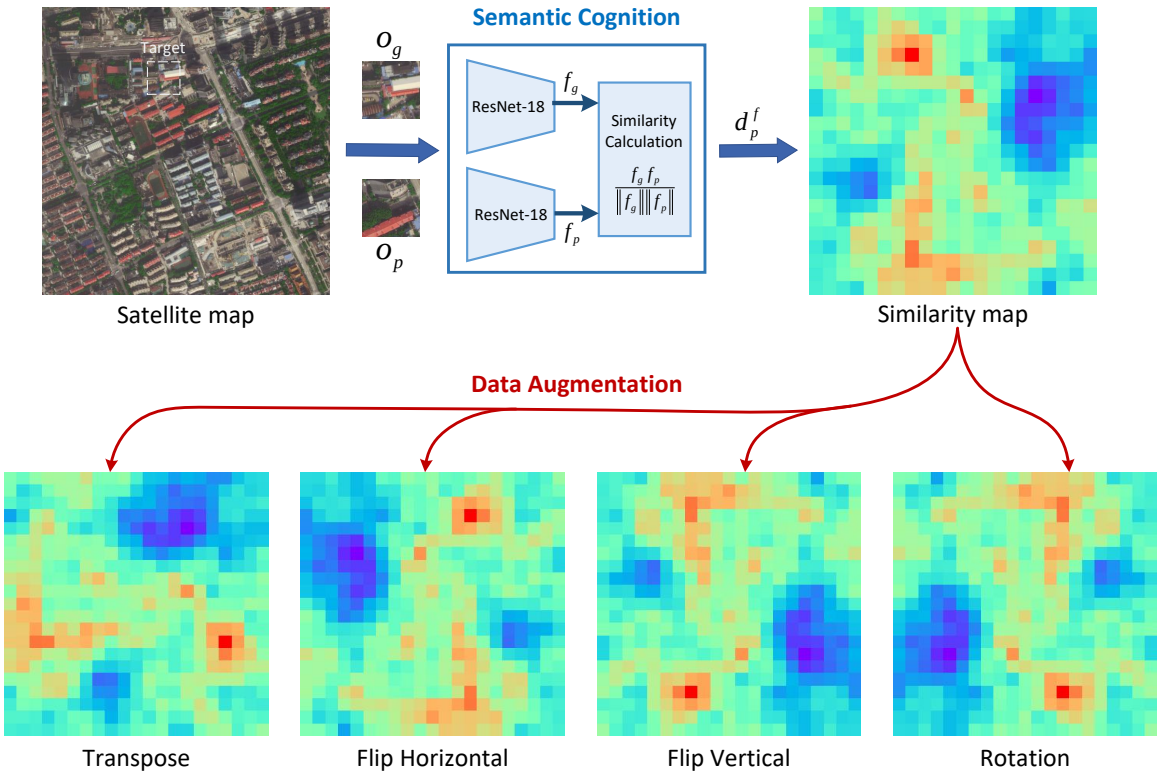


Fig. 4. The process of constructing the similarity map. The semantic cognition module extracts the semantics of the original satellite map to obtain the similarity map of the target image, i.e., the similarity distribution of the target image on the original map. To avoid the learned policy overfitting training scenarios (that is, similarity maps), we enhance training scenarios with transformations such as the transpose, flip and rotation on the original similarity maps.

III. SUPPLEMENTARY TABLES

Algorithm 1 Baseline-DRL algorithm based D3QN

Input: Siamese network Ψ parameters φ; initial Q network parameters θ; target Q network parameters $\theta^- = \theta$; empty replay buffer D; minibatch size N_b; target network update frequence N_u; maximum number of steps per Episode N_m.

Output: Action-value function Q_θ.

for episode $e \in \{1, 2, ..., M\}$ **do**
  Initialize state $s_1 = \{\Psi_\varphi(x_1), \Psi_\varphi(x_g), (0,0)_N, d_1^b, b_1\}$, time step $t = 1$ and state sequence $S_1 = \{s_1\}$
  while not done or $t \leq N_m$ **do**
    With probability $(1 - \epsilon)$ select a random action a_t, otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$
    Execute action a_t, obtain reward r_t and next state s_{t+1}
    Store transition $(s_t, a_t, r_t, s_t + 1)$ in D
    Sample random minibatch of N_b transitions (s_i, a_i, r_i, s_{i+1}) from D
    For each of the N_b transitions, compute the target:
    $$y_i = \begin{cases} r_i, \text{ if } s_{i+1} \text{ is terminal} \\ r_i + \gamma \cdot Q(s_{i+1}, \arg\max_a(s_{i+1}, a; \theta); \theta^-), \text{ else.} \end{cases}$$
    Perform a gradient descent step on $(y_i - Q(s_i, a_i; \theta))^2$ with respect to the network parameter θ
    Replace target parameters $\theta^- = \theta$ every N_u steps
    set $s_t \leftarrow s_{t+1}$ and $t \leftarrow t + 1$.
  end while
end for

Algorithm 2 CERL algorithm based D3QN

Input: Siamese network Ψ parameters φ; initial Q network parameters θ; target Q network parameters $\theta^- = \theta$; empty replay buffer D; minibatch size N_b; target network update frequence N_u; maximum number of steps per Episode N_m.

Output: Action-value function Q_θ.

for episode $e \in \{1, 2, ..., M\}$ **do**
  Initialize state $s_1 = \{dist(\Psi_\varphi(x_1), \Psi_\varphi(x_g)), (0,0)_N, d_1^b, b_1\}$, time step $t = 1$ and state sequence $S_1 = \{s_1\}$
  while not done or $t \leq N_m$ **do**
    if $S_{N1} \neq S_{Nk}, k = 1, 2, ..., m$ **then**
      With probability $(1-\epsilon)$ select a random action a_t, otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$
      Execute action a_t, obtain reward r_t and next state s_{t+1}
      Store transition (s_t, a_t, r_t, s_{t+1}) in D
      Sample random minibatch of N_b transitions (s_i, a_i, r_i, s_{i+1}) from D
      For each of the N_b transitions, compute the target:

$$y_i = \begin{cases} r_i & \text{if } s_{i+1} \text{ is terminal} \\ r_i + \gamma \cdot Q(s_{i+1}, \arg\max_a(s_{i+1}, a; \theta); \theta^-), & \text{otherwise.} \end{cases}$$

      Perform a gradient descent step on $(y_i - Q(s_i, a_i; \theta))^2$ with respect to network parameter θ
      Update target parameters $\theta^- = \theta$ every N_u steps
      Increase time step $t \leftarrow t + 1$.
    else
      Compute escape policy $A_t^e = \{a_t^e, a_{t+1}^e, ...a_{t+m}^e\}$
      Perform escape action sequence $\{a_t^e, a_{t+1}^e, ...a_{t+m}^e\}$ and obtain next state s_{t+1}
      Increase time step $t \leftarrow t + m$.
    end if
    set $s_t \leftarrow s_{t+1}$ and add new s_t to state sequence S_t
  end while
end for

Hyperparameter	Value	Description
minibatchsize N_b	128	Number of training cases over which each stochastic gradient descent (SGD) update is computed
replay memory size	1000000	SGD updates are sampled from this number of most recent frames.
target network update frequency N_u	100	The frequency (measured in the number of parameter updates) with which the target network is updated.
discount factor γ	0.98	Discount factor gamma used in the Q-learning update.
learning rate	0.001	The learning rate used by RMSProp.
initial exploration	1	Initial value of ϵ in ϵ-greedy exploration.
final exploration	0.1	Final value of ϵ in ϵ-greedy exploration.
final exploration episode	1000	The number of episodes over which the initial value of ϵ is linearly annealed to its final value.
state history length N_h	8	The length of history trajectory information in the state.

TABLE I. Summary of hyperparameter

Notation	Description	Notation	Description
s_t	The state at time step t	s_t^h	History trajectory information
a_t	The action at time step t	b_t^o	Boundary information
r_t	The reward at time step t	s_t^o	Overall states from time step 1 to t
o_g	Target image	c_t	The control signal
o_t	Observation image at time step t	g_t^p	The gate network in the policy network
f_g	Target image feature	g_t^e	The gate network in the escape module
f_t	Image feature at time step t	p_t	The position of agent at time step t
d_t^f	Feature distance at time step t	p_t^s	Sub-goal postion at time step t
α_t	Flight direction at time step t	A_t^e	Escape action sequence at time step t
d_t^b	Boundary distance at time step t	θ	Parameter vector of Q network
b_t	Boundary index at time step t	Ψ	Siamese network
r_t^g	Goal-reaching reward at time step t	φ	Parameter vector of siamese network
r_t^s	Similarity reward at time step t	D	Experience replay buffer
r_t^b	Boundary penalty at time step t	S_i	Indicator of success in the i-th episode
r_t^p	Time-step penalty at time step t	l_i	Actual path length in the i-th episode
Δd_t^f	The change in feature distance	d_i	Optimal path length in the i-th episode

TABLE II. Summary of notation