Resilient Navigation Among Dynamic Agents with Hierarchical Reinforcement Learning

Sijia Wang[1,2], Hao Jiang[1,2(✉)], and Zhaoqi Wang[1,2]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
jianghao@ict.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. Behaving safe and efficient navigation policy without knowing surrounding agents' intent is a hard problem. This problem is challenging for two reasons: the agent need to face high environment uncertainty for it can't control other agents in the environment. Moreover, the navigation algorithm need to be resilient to various scenes. Recently reinforcement learning based navigation has attracted researchers interest. We present a hierarchical reinforcement learning based navigation algorithm. The two-level structure decouples the navigation task into target driven and collision avoidance, leading to a faster and more stable model to be trained. Compared with the reinforcement learning based navigation methods in recent years, we verified our model on navigation ability and the resilience on different scenes.

Keywords: Reinforcement learning · Navigation

1 Introduction

Navigating in crowded space with high efficiency and safety is a challenging task. Traditional approaches often need to adjust theirs parameters manually for different scenes, which restricts the application environment [10,18]. In order to perform adaptive and resilient behaviors for diverse scenes, navigation algorithm needs to understand different environment semantic.

To address the above issues, some work attempt to learn navigation behaviors from data. According to the source of data, the approaches can be divided into imitation learning and reinforcement learning (RL). Imitation learning [6,12,14,17,20] learns navigation policy based on human walking trajectory data from real word, resulting in that the agent can perform similar behaviors to humans. However, the application of imitation learning restrict to the scenes of collected data.

In recent years, RL shows the level of human beings in go and games [15,16]. More and more researchers try to apply RL to navigation. Reinforcement learning does not depend on real data sets, it can continuously obtain training data

N. Magnenat-Thalmann et al. (Eds.): CGI 2021, LNCS 13002, pp. 504–516, 2021.
https://doi.org/10.1007/978-3-030-89029-2_39

through the virtual environment. Moreover, the optimization goal of RL is the cumulative environmental feedback signal. Compared with imitation learning, instead of simply imitating the real data, RL thinks about what strategies will make the cumulative environmental feedback higher. In recent years, the RL based navigation methods have achieved good results [3–5,13], which verify the effectiveness of RL.

To address this, we propose to use hierarchical RL to generate plausible and collision free trajectories. The main contributions of our work include:

1) We built an effective and novel HRL framework to guide the agent to reach the destination efficiently. By using hierarchical RL framework, the navigation task is decoupled into target driving and collision avoidance, so that a stable and robust model can be trained, which can quickly adapt to a new environment.
2) Through comparisons with state-of-the-art RL-based methods, our model achieves superior performance, especially in various challenging resilient experiments.

2 Related Work

In this section, we review related works on navigation and hierarchical RL which our work refers to.

2.1 Conventional Methods for Navigation

Helbling et al. [9,10] proposed social forces model to describe interactions among pedestrians. The model is based on a potential field in which attractive forces lead agents to the destination and repulsive forces block the surrounding obstacles. Reactive model predicts the collision time based on the current velocity. The representing work is ORCA [21], which is based on RVO [2]. ORCA seeks joint obstacle avoidance velocities under reciprocal assumptions. These models which are designed elaborately by researchers behave well in the specific application scene, while they rely on hand-craft functions and can not generalize well to various scenes.

2.2 Deep Reinforcement Learning Methods for Navigation

Earlier works [8,23] was limited by calculate capability, thus researches tried to simplify problems when applying RL to the navigation problem. The combination of RL and deep learning enables processing data with higher dimensions and larger state space. Recent work of navigation by deep reinforcement learning can be divided into two categories according to the algorithm.

One is value based reinforcement learning, which decomposes the action space into discrete velocity set V according to speed and direction. The method CADRL [5] first applied DRL to navigation, which adapted two-agent to the

multi-agent case through maximin operation to pick up the best action. Chen et al. [4] proposed SARL which rethinks human-robot pairwise interactions with a self-attention mechanism.

One is policy-based reinforcement learning, whose action space is continuous. Long et al. [13] directly mapped raw sensor measurements to desired collision avoidance policy and presented a multi-scene multi-stage training framework for adapting to different scenes. Based on [7,13] learned safer and more resilient behaviors for navigation by integrating uncertainty estimation.

2.3 Hierarchical Reinforcement Learning

Hierarchical reinforcement learning (HRL) is inspired by divide-and-conquer, decoupling task to reduce training difficulty. Bacon et al. [1] proposed Option-Critic Framework which decouples the problem into two levels. The high policy is responsible for choosing an option. The low level do the low-level policy following the option until meet the option's termination condition.

Vezhnevets [22] proposed FuUdal Networks where two levels of hierarchy within an agent communicate via explicit goals. Both the high level and low level are deep learning model and no gradients are propagated between two levels. The Manager receives its learning signal from the environment alone. Our model takes inspiration from the design of FeUdal Networks.

3 Approach

3.1 Overview

In this work, we consider the problem that an agent navigates towards a goal on the ground where N dynamic obstacles exists. Both the agent and N dynamic obstacles are modeled as discs with the same radius. The agent can not communicate with other dynamic obstacles. Therefore at each time t the agent's observation is N obstacles' positions $p_i^t = [p_x^t, p_y^t] \in P, P^t = \{p_1^t, p_2^t, \cdots, p_n^t\}$ and velocities $v_i^t = [v_x^t, v_y^t] \in V, V^t = \{v_1^t, v_2^t, \cdots, v_n^t\}$.

Figure 1 shows the overview of method, our model takes inspiration from feudal reinforcement learning [22] where levels of hierarchy communicate via explicit goals. The high-level module aims to optimize long term interest, whose output is the sub-goal $g_t = (p_x^{t+c}, p_y^{t+c})$ for the future. The low-level module aims to safely and efficiently navigate to the sub-goal, whose output is primitive actions which is 2-dimensional velocity $a_{t+i} = (v_x^{t+i}, v_y^{t+i})$. The low-level is the module that actually interacts with the environment.

Both of high-level and low-level are constructed by deep RL models and optimize their policy respectively on the reward received from the environment. Similar to the FeUdal Networks [22], there are no gradient between two levels. The information that agent can get from the environment, x_t contains last six consecutive observations: $x_t = \left\{P^{t-5}, V^{t-5}, \cdots, P^t, V^t\right\}$. As Fig. 2 shows, our model converts the change of obstacles' positions into the 360-degree laser form

o_z^t, the shape of which is 6×360. The high-level and low-level modules don't use the same neural network but share the same neural network architecture as the Fig. 3 shows. Based on the current velocities of the obstacles, we predict the obstacles' positions in next 3 time steps by linear interpolation and convert the future positions to the laser form o_p^t. θ_t is the current agent's orientation and o_{z+p}^t is a slice of o_z^t and o_p^t chosen by θ_t: $o_{z+p}^t = o_z^t[\theta_t] + o_p^t[\theta_t]$. p_g^t is the agent's final goal (for high-level) or sub-goal (for low-level). v_t is the agent's current velocity. Both high and low level take the agent's position as the origin for the local coordinate every time step, high-level's y-axis points toward the final goal and low-level's y-axis points toward the sub-goal.

Here we introduce the transition between the high-level and low-level. Every c time steps, the high-level module calculates the sub-goal g_t for the future c time steps and conveys it to the low-level module. c is a hyper-parameter which we set 20. Low-level receives the sub-goal and starts the loop of calculating the primitive action, two-dimensional velocity v_t. The low-level module will not stop until meeting the terminal conditions. The terminal conditions contain three situations, the first one is the agent arrives the sub-goal, the second one is the agent collides with other obstacle, the third one is the agent have performed low-level action c times.

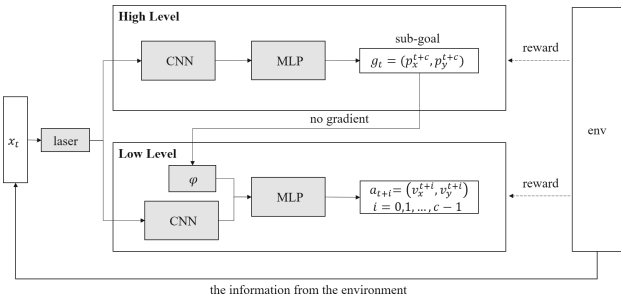

Fig. 1. The overview of our model. There are two levels in our model. The high-level module aims to optimize long term interest, whose output is the sub-goal $g_t = (p_x^{t+c}, p_y^{t+c})$ for the future. The low-level module aims to safely and efficiently navigate to the sub-goal, whose output is primitive actions which is 2-dimensional velocity $a_{t+i} = (v_x^{t+i}, v_y^{t+i})$.

3.2 High-Level Module

Above all, low-level module actually interacts with the environment while high-level module interacts with environment by directing low-level module. The responsibility for high-level module is giving low-level module a good sub-goal which can balance the need of the reaching final goal and safety. Therefore, we estimate high-level policy by the trajectory that the low-level module actually performed in loop after receiving the sub-goal from high-level.

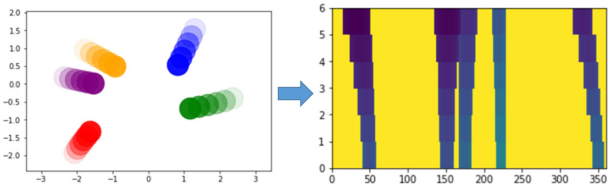

Fig. 2. In our model, the agent processes the environment observation to the form of laser.

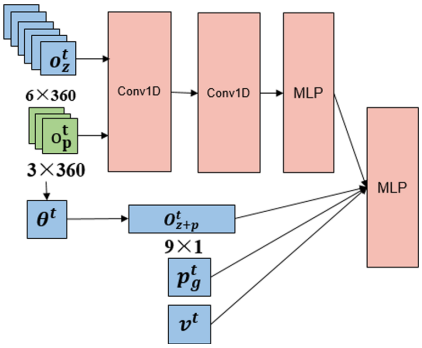

Fig. 3. The neural network shared by high-level and low-level module.

We first introduce the notations of the trajectory that the low-level module generated, then introduce the estimation criteria on the trajectory. After the low-level received sub-goal, assume that low-level performed c^L times before meeting the terminal condition, $c^L \leq c$. Then the agent's trajectory is $\{p_t, p_{t+1}, \cdots, p_{t+c^L}\}$, the minimal distances between the agent and obstacles are $\{d_t, d_{t+1}, \cdots, d_{t+c^L}\}$.

Let the displacement during c^L times be $p_{t+c^L} - p_t$ and the travelled distance be $\sum_{k=1}^{k=c^L} \|p_{t+k} - p_{t+k-1}\|$. Let the agent's final goal be g, then $g - p^t$ is the relative vector from the final goal to the agent's position at the beginning of the loop.

We define d_{goal} to represent the relative distance that the agent navigate to the final goal during the c^L times and the relative distance the agent move per unit time is $UnitGoal$.

$$d_{goal} = \frac{(p^{t+c^L} - p^t)(g - p^t)}{\| g - p^t \|} \tag{1}$$

$$UnitGoal = \frac{d_{goal}}{c^L} \tag{2}$$

If the agent's minimal distance to the obstacles is less than 0.25 m, then the reward function will give a penalty signal for being too close to obstacles, which is -0.25+d. Formula 3 is the sum of the distance penalty $CloseDist$, $1_{d<0.25}$(d) is an indicator function.

$$CloseDist = \sum_{t=1}^{c^L} 1_{d<0.25}(d) * (-0.25 + d) \tag{3}$$

We define $ExtraPath$ to represent the difference between the travelled distance and displacement.

$$ExtraPath = \sum_{k=1}^{k=c^L} \|p_{t+k} - p_{t+k-1}\| - \|p^{t+c^L} - p^t\| \tag{4}$$

Formula 5 is the high-level reward function, which takes $UnitGoal$, $CloseDist$, and $ExtraPath$ into account. w_1, w_2, w_3 is weighting parameters which are set 0.5, -0.5 and 0.5 at the beginning for dimensional homogeneity. Then these parameters were gradually adjusted by persistent attempts to $w_1 = 0.8$, $w_2 = -0.5$, $w_3 = 0.4$.

$$R_t = w_1 UnitGoal + w_2 ExtraPath + w_3 CloseDist \tag{5}$$

High-level module is responsible for giving a good sub-goal, therefore we train the module to maximize the one step interest. We refer to the training design of DDPG [11], a policy based RL algorithm, and transform the design to optimize one step interest. There are three neural networks, a policy network $Actor$ for giving out the sub-goal, two value networks $Critic$ and $Critic_{target}$ for describing the state value. The network structure of $Critic$ and $Critic_{target}$ is the same. The task of $Critic_{target}$ is to provide policy's value without gradient, thus it won't be optimized when training, the parameters of $Critic_{target}$ will periodically copy from $Critic$. The loss for $Critic$ is shown as Formula 6, whose aim is minimizing the value estimated by $Critic$ and the real reward. The loss for $Actor$ is shown as Formula 7, whose aim is maximizing the state value that the Actor can bring.

$$critic\ loss = (Critic(s_t, g_t) - r_t)^2 \tag{6}$$

$$actor\ loss = -Critic_{target}(s_t, Actor(s_t)) \tag{7}$$

3.3 Low-Level Module

The responsibility for low-level module is navigating to the sub-goal safely, this task is similar to the mono-layer RL work [4,13]. The termination condition of low-level is reaching sub-goal or colliding with obstacles or the executions is over c. Formula 8 shows the low-level reward function which awards navigating to the sub-goal and penalizes collisions.

If the agent collides with other obstacles, we will give penalty $r_{collision} = -3$. If the agent is too close to other obstacles (the distance to other obstacles is less than 0.2 m), we will give penalty on the uncomfortable distance: $-0.6 + d_{min}/2$. Otherwise, we will give the award for navigating to the sub-goal: $-w_g(\|p^{t-1} - g\| - \|p^t - g\|)$, where $w_g = 2.5$. Our reward function doesn't award for reaching the final goal specially like the mono-layer RL method [4,13], because low-level

can end up with reaching the final goal or reaching the sub-goal or timeout so that awarding the final goal will induce the inequality. Previous discount factor γ in [4,13] is over 0.9. However the collision influence doesn't need be so far, thus the discount factor γ in our model is 0.6.

$$r_t = \begin{cases} r_{collision} & p^t - p_j^t < 2R \\ -0.6 + d_{min}/2 & d_{min} < 0.2 \\ -w_g(\|p^{t-1} - g\| - \|p^t - g\|) & otherwise \end{cases} \tag{8}$$

Low-level module is trained using Deep Deterministic Policy Grading (DDPG) [11], a policy based method. Compare to the stochastic policy search of Proximal Policy Optimization (PPO) [19], the deterministic policy of DDPG accelerates convergence on navigation problem. When navigating in the crowded environment, low-level module will frequently collide with other obstacles, which possibly leading to the training lies in local minimum. For improving this problem, low-level module uses two tricks. The first is controlling the ratio of success and fail trajectories in the data set, which we set 0.6:0.4 in our paper. The second is adding protect for the low-level policy, the 2-d velocity. The conventional method ORCA has robust collision avoidance ability, whose input is the prefer velocity. After the low-level policy network outputs the velocity v_t, we use ORCA to reduce the collision probability by taking v_t as the prefer velocity. In other words, we treat the low-level model's output as the prefer velocity for ORCA to increase security.

4 Experiment

4.1 Scene Design

The scenes should be able to verify the model's navigation ability, we design the scenes from two aspect: First, the agent should be able to maintain navigation ability when the scene's size changed. Second, the agent should be able to maintain navigation ability when the scene change to dissimilar scene.

Therefore, this paper train and test on following three scenes which are easy to change size. Figure 4 shows the diagram of these three scenes, the red disc represents the agent and the blue disc represents the human, the stars with the corresponding color represents the agent's or humans' goals. Fig(a) is scene *Squeeze*, where an agent and a human randomly positioned on a circle of radius of rm and their goal positions are on the opposite side of the same circle. (b) is scene *Circle*, its design is similar to *Squeeze*, the only difference is *Circle* has one agent and five humans. (c) is scene *Square*, an agent and five humans randomly positioned on a square whose side length is w m.

We design two kinds of experiment. Experiment 1 compares the models' navigation, where the train and test scene is the same. Experiment 2 compares the models' resilience from two aspect: compare the resilience explicitly on scene size and scene type.

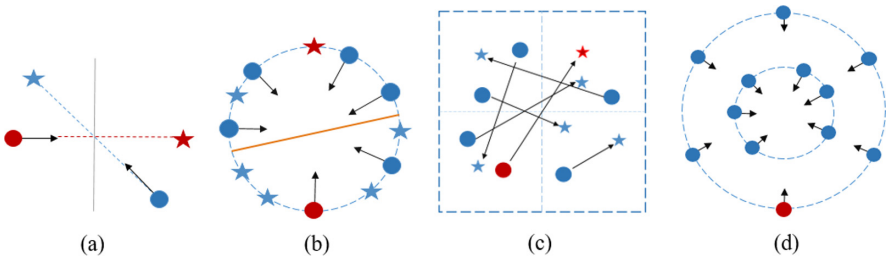

Fig. 4. The scenes that we use in this paper to verify the effectiveness of our model.

4.2 Perform Metrics

To fairly compare our model with other models, every test scene is evaluated for 100 repeats. We use the three performance metrics. Success rate represents the ratio that the agent successfully arrived the destination without collision. Collision rate represents the ratio that the agent collide with other obstacles. Average navigation time represents the average navigation time of the successful trajectories.

4.3 Navigation Ability Comparison

We compared with the representative models which are based on RL. As expected, the CADRL and the SARL have low collision rate due to their training algorithm is value based, which cautiously enumerates all the discrete actions. Long et al. [13] uses policy-based algorithm which action space is continuous, which exploration space is further higher than the value based. With the scene's complexity increase, Long et al. fails to avoid the obstacles. The collision rates of *Squeeze*, *Circle* and *Square* are 0.24, 0.3, 0.47 respectively. Our model's hierarchical structure decouples the navigation task into target-driven task and collision avoidance task, each layer concentrates on the its own responsibility and our low-level module add ORCA policy to protect, thus our model has the highest success rate among three scenes.

As for the average navigation time, the value-based methods' has longer time than the policy-based methods on *Circle* and *Square*. Because the value-based methods' action space is discrete, which means its trajectories are less smooth than the policy-based trajectories. We also make a statistic on the trajectories' kinetic energy, we compute the minimum, mean value and maximum of 100 trajectories' energy. As the fifth row in Table 1 shows, our model has the lowest value among *Circle* and *Square*.

4.4 Resilience Comparison

Table 1. Testing the learning ability of the model: train and test on the same scene.

Metrics	Method	Squeeze	Circle	Square
SuccessRate	CADRL	0.95	–	–
	SARL	0.93	0.93	0.96
	Long et al.	0.76	0.7	0.53
	Our model	**0.96**	**0.96**	**1.0**
CollisionRate	CADRL	0.05	–	–
	SARL	0.07	0.01	0.0
	Long et al.	0.24	0.3	0.47
	Our model	**0.04**	0.04	**0.0**
AvgNavTime	CADRL	9.96	–	–
	SARL	9.77	10.93	8.23
	Long et al.	10.25	8.8	5.61
	Our model	**8.5**	**9.66**	6.76
Energy	CADRL	(31.2,36.8,51.6)	–	–
	SARL	(32.5,38.0,49.0)	(32.7,37.9,46.4)	(6.0,29.6,68.3)
	Long et al.	(37.3,37.3,37.3)	(30.6,30.6,30.6)	(4.9,16.3,43.0)
	Our model	(29.5,32.5,41.0)	**(23.3,28.8,38.0)**	**(2.8,22.4,41.0)**

Resilience on Scene Size. In this experiment, we train the model on *Squeeze* with diameter 8 m, then test the model on *Squeeze* with diameter 16 m. The same configuration for *Circle* and *Square*. The test results are shown in Table 2 (3rd, 4th and 5th column). Above all, the value-based model CADRL and SARL behaved bad on scene *Squeeze* and *Circle*, the success rates of which are 0.03 and 0 respectively. Because *Square* changes little with its side length increases, the SARL still retains 0.49 success rate. This is because, the value-based method chooses policy by the formula $a_t \leftarrow \arg\max_{a_t \in A} R(s_t, a_t) + \gamma V(\hat{s}_{t+1})$ which highly depends on the accurate estimation on the state value, $V(\hat{s}_{t+1})$. If the value network hasn't seen the state s_{t+1} before, it is hard for value-based method to choose a reasonable action.

The policy-based method uses the policy network to learn the relation of environment state and the action. Therefore, even if the method meets the unfamiliar environment state, the policy network still knows the general direction of the action. As the second row in Table 2 shows, the success rates of the Long et al. are 0.76, 0.7 and 0.53 explicitly, which are apparently higher than value-based method. However, it has higher collision rate with the complexity of the scene increases. Our model retains high success rate when the scene size increased,

where the success rates are 0.96, 0.95, and 0.92 explicitly. Our high-level module helps to avoid the relative crowded area by choosing temporal destination.

Resilience on Scene Type. For testing the resilience on scene type, we train the model on *Squeeze* with diameter 8 m, then test on *Circle* with diameter 8 m and *Square* with side length 8 m. Thus the scene sizes of train and test are the same. The test results are shown in Table 2 (6th and 7th column). Face to the unfamiliar scene with the same size, the CADRL and SARL retain some navigation ability, whose success rates are over 0.65. The SARL behaves better than CADRL for its value network can process crowd while the CADRL can only process pair-wise relationship. The success rate of our model on *Circle* and *Square* are 0.99 and 0.92, apparently behaves better than other models.

Table 2. Testing the resilience of the model: train and test on different scenes.

Metrics	Method	Resilience on scene size			Resilience on scene type	
		Squeeze (8 m→16 m)	Circle (8 m→16 m)	Square (8 m→16 m)	Squeeze→Circle	Squeeze→Square
SuccessRate	CADRL	0.03	–	–	0.65	0.67
	SARL	0	0	0.49	0.83	0.89
	Long et al.	0.88	0.71	0.54	0.82	0.79
	Our model	**0.96**	**0.95**	**0.92**	**0.99**	**0.92**
CollisionRate	CADRL	0.02	–	–	0.03	0.03
	SARL	0.05	0.05	0.00	0.17	0.11
	Long et al.	0.12	0.29	0.31	0.18	0.21
	Our model	0.04	**0.00**	**0.00**	**0.00**	**0.00**
AvgNavTime	CADRL	20.08	–	–	12.94	7.40
	SARL	–	–	10.73	8.48	6.60
	Long et al.	21.50	17.00	11.88	8.25	6.40
	Our model	**18.07**	18.07	12.62	9.54	8.92
Energy	CADRL	(79.3,82.9,86.3)	–	–	(25.7,103.4,165.9)	(6.0,25.2,61.5)
	SARL	–	–	(9.0,57.2,77.0)	(31.5,33.4,43.5)	(6.0,24.8,47.6)
	Long et al.	(82.4,82.4,82.4)	(64.6,64.6,64.6)	(2.9,43.2,76.0)	(30.4,30.4,30.4)	(3.0,21.0,41.6)
	Our model	(66.3,70.5,74.9)	(48.7,64.2,70.8)	(11.1,49.6,76.2)	(25.3,29.7,39.3)	(1.1,23.8,57.1)

Table 3. Test the navigation ability of the model: train and test in Concentric scene.

	Success	Collision	AvgNavTime
Mono-layer	0.18	0.82	19.05
Our model	0.99	0.00	20.21

4.5 Ablation Experiment

To verify the effectiveness of our hierarchical architecture, we compare our model with mono-layer RL model. The reward function and train configuration of mono-layer RL is same to our model's low-level module, except the low-level module calculate velocity policy based on sub-goal while our model based on final goal. We compare the two models under the scene Concentric (the Fig(d) in Fig. 4). The radius of two circles are 8 m and 16 m. The test result is shown

in Table 3. The success rate of our model is 0.99 which are much higher than the mono-layer's 0.18. As the Fig. 5 shows, mono-layer model's policy (the left trajectory) is aggressive that the agent walks straight to the final goal. Although the agent tried to avoid the nearby obstacles, the high-density led to collision at last. Our policy (the right trajectory) avoid the high-density by sub-goals which are represented as red stars in Fig. 5.

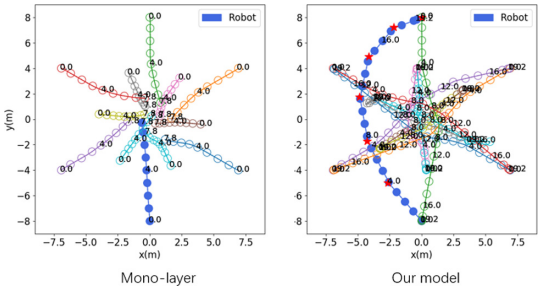

Fig. 5. Train and test the models on Concentric scene. (Color figure online)

5 Conclusion

In this paper, we propose a hierarchical reinforcement learning based navigation algorithm, which decouple navigation task into target-driven and collision avoidance. We evaluated our approach by comparing the trajectories taken by the agent with previous methods. Our experimental results suggest that our approach produces motion that is more resilience in different scenes.

Acknowledgments. This work was supported by National Key Research and Development Program of China (No. 2018AAA0103002 and 2017YFB1002600) and National Natural Science Foundation of China (No. 61702482 and 62002345).

References

1. Bacon, P., Harb, J., Precup, D.: The option-critic architecture. CoRR abs/1609.05140 (2016)
2. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: 2008 IEEE International Conference on Robotics and Automation, pp. 1928–1935. IEEE (2008)
3. Chen, C., Hu, S., Nikdel, P., Mori, G., Savva, M.: Relational graph learning for crowd navigation. arXiv preprint arXiv:1909.13165 (2019)
4. Chen, C., Liu, Y., Kreiss, S., Alahi, A.: Crowd-robot interaction: crowd-aware robot navigation with attention-based deep reinforcement learning. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 6015–6022. IEEE (2019)

5. Chen, Y.F., Liu, M., Everett, M., How, J.P.: Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 285–292. IEEE (2017)
6. Fahad, M., Chen, Z., Guo, Y.: Learning how pedestrians navigate: A deep inverse reinforcement learning approach. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 819–826. IEEE (2018)
7. Fan, T., Long, P., Liu, W., Pan, J., Yang, R., Manocha, D.: Learning resilient behaviors for navigation under uncertainty. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 5299–5305. IEEE (2020)
8. Godoy, J., Chen, T., Guy, S.J., Karamouzas, I., Gini, M.: ALAN: adaptive learning for multi-agent navigation. Autonomous Robots **42**(8), 1543–1562 (2018)
9. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic. Nature **407**(6803), 487–490 (2000)
10. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**(5), 4282 (1995)
11. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
12. Liu, Y., Xu, A., Chen, Z.: Map-based deep imitation learning for obstacle avoidance. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8644–8649. IEEE (2018)
13. Long, P., Fan, T., Liao, X., Liu, W., Zhang, H., Pan, J.: Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 6252–6259. IEEE (2018)
14. Long, P., Liu, W., Pan, J.: Deep-learned collision avoidance policy for distributed multiagent navigation. IEEE Robot. Autom. Lett. **2**(2), 656–663 (2017)
15. Mnih, V., et al.: Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
16. Peng, X.B., Abbeel, P., Levine, S., van de Panne, M.: DeepMimic: example-guided deep reinforcement learning of physics-based character skills. ACM Trans. Graph. (TOG) **37**(4), 1–14 (2018)
17. Pfeiffer, M., et al.: Reinforced imitation: sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. IEEE Robot. Autom. Lett. **3**(4), 4423–4430 (2018)
18. Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pp. 25–34 (1987)
19. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
20. Tai, L., Zhang, J., Liu, M., Burgard, W.: Socially compliant navigation through raw depth inputs with generative adversarial imitation learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1111–1117. IEEE (2018)
21. Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics Research, pp. 3–19. Springer (2011). https://doi.org/10.1007/978-3-642-19457-3_1

22. Vezhnevets, A.S., et al.: Feudal networks for hierarchical reinforcement learning. In: International Conference on Machine Learning, pp. 3540–3549. PMLR (2017)
23. Zhang, C., Lesser, V.: Coordinating multi-agent reinforcement learning with limited communication. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, pp. 1101–1108 (2013)