# CAPABILITY ITERATION NETWORK FOR ROBOT PATH PLANNING

Buqing Nie,* Yue Gao,** Yidong Mei,* and Feng Gao***


## Abstract

Path planning is an important topic in robotics. Recently, value iteration-based deep learning models have achieved good performance such as value iteration network (VIN). However, existing value iteration based methods suffer from slow convergence and low accuracy on large maps, hence restricted in path planning for agents with complex kinematics such as legged robots. Therefore, we propose a new value iteration-based path planning method called capability iteration network (CIN). CIN utilizes sparse reward maps and encodes the capability of the agent with state-action transition probability, rather than a convolution kernel in previous models. Furthermore, two training methods including end-to-end training and training capability module alone are proposed, both of which speed up convergence greatly. Several path planning experiments in various scenarios, including on 2D, 3D grid world and real robot with different map sizes, are conducted. The results demonstrate that CIN has higher accuracy, faster convergence and lower sensitivity to random seed compared to previous VI-based models, hence more applicable for real robot path planning.



## Key Words

Path planning, value iteration, value iteration network, capability iteration network


## 1. Introduction

Path planning is an important direction for autonomous robot, whose objective is to compute a path to the goal region that not only avoids collisions but also satisfies kinematic constraints of the robot [1], [2]. Various classic methods including $A^*$ [3] and potential fields [4] are proposed. Recently, deep reinforcement learning (DRL) has been utilized to solve path planning problems without handcraft parameters tuning according to the robot topological structures [5], for example on robotic manipulator [6], [7], drilling robot [8] and planetary rover [9].

DRL has the benefits of representing complex state space and learning a network, which enables function approximation of the features and future reward values [10]. However, most existing DRL-based path planning models lack planning modules, which make them sample inefficient and difficult to generalize for unseen environments [11]. This makes DRL difficult to be applied in real scenarios, like planning paths for real robots. A recent work, *value iteration network* (VIN) [11], combines recurrent convolution neural networks and max-pooling to emulate value iteration (VI) [12] algorithm, which enables VIN to plan paths for unseen mazes. However, VIN is particularly challenging on training when given large maps. In addition, how to utilize VIN to solve path planning for robots with complex kinematics remains a challenging task.

For RL models, sparse reward is an important property required for convergence and generalization [13]. VIN utilizes convolution layers to compute the reward map, which is not sparse and may decrease the accuracy of VIN because of its defects, especially on large maps. Furthermore, VIN utilizes concatenation and a convolution kernel $P^a_{s'}$ (probability distribution for the next state based on the action to take) instead of the state conditioned transition probability $P^a_{ss'}$ in the value iteration algorithm.

In this paper, a new VI-based model, *capability iteration network* (CIN), is proposed. In CIN, sparse reward maps are utilized, which limit reward maps influencing the accuracy of the model. Furthermore, as illustrated in Fig. 1, a capability module is utilized to encode each state $s$ with a state-conditioned transition probability $P^a_{ss'}$. These state-conditioned transition probabilities in CIN represent the capability of the agent, which differentiates CIN from previous models that utilize $P^a_{s'}$ without the knowledge of the current state. CIN is trained to learn the capability of the agent, which is relevant to the local region and independent of the global map. Therefore, CIN is able to be trained on local maps with a faster learning speed. Several experiments demonstrate that CIN can be applied to scenarios including 2D mazes, 3D terrain maps and path planning for real hexapod robots.


* Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China; e-mail: {niebuqing, meiyidong}@sjtu.edu.cn

** MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai 200240, China; e-mail: yuegao@sjtu.edu.cn

*** State Key Laboratory of Mechanical System and Vibration, Shanghai Jiao Tong University, Shanghai 200240, China; e-mail: fengg@sjtu.edu.cn

Corresponding author: Yue Gao



*Recommended by Dr. Ponnambalam S.G.*
*(DOI: 10.2316/J.2022.206-0598)*

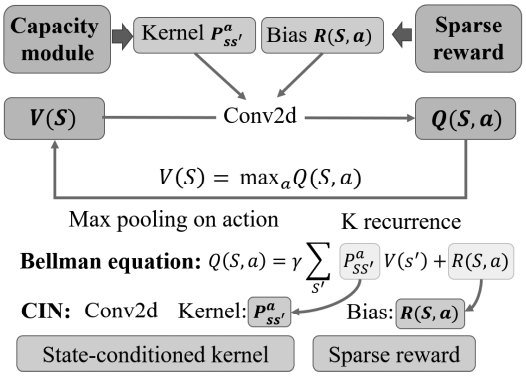



Figure 1. Structure of the *capability iteration network* (CIN).

The main contributions of this paper are as follows:

- A new VI-based model called CIN is proposed, which greatly improves performance on accuracy and learning speed compared with previous VI-based algorithms on various scenarios.
- Two training methods are provided: End-to-end training and capability module training alone on small maps. Both of these methods greatly improve learning speed compared with the previous models.
- Experiments on 2D, 3D maps and hexapod robot demonstrate that CIN outperforms previous models on accuracy and learning speed and can be utilized for path planning with real robots.

## 2. Related Work

### 2.1 Path Planning

Path planning is one of the essential tasks in the robotics research. It has been widely applied to lots of scenarios including auto driving, autonomous underwater vehicle control and video games [14], [15], [16].

$A^*$ algorithm developed by Hart *et al.* [3] is widely used in path planning for its optimality and simplicity. Many various variants of $A^*$, such as $D^*$ Lite [17] and any-angle $A^*$ [16], are proposed. Artificial potential field (APF) introduced by Oussama Khatib [4] is another important method, which creates an artificial potential field to guide the agent to the goal [18]. However, the agent may stay in local minimum regions, which is the main drawback of the APF method. Probabilistic path planning algorithms, such as rapidly-exploring random trees [19], are also effective methods, which do not require any environment modeling and outperform previous algorithms in terms of computation cost. Genetic algorithm is another effective path planning problem in various scenarios such as robot manipulators and unmanned surface vehicle [20], [21].

Recently, deep reinforcement learning (DRL) has been applied to solve path planning problems in many scenarios such as robotic manipulator [6], [7], drilling robot [8] and planetary rover [9]. DRL-based methods try to summarize patterns through numerous attempts to generate a suitable path in a new environment, which does not require modeling of environment and robot before planning paths. However, existing DRL methods rely heavily on training data, which limits its application scenarios.

### 2.2 Reinforcement Learning

Reinforcement learning (RL) is a powerful paradigm to solve the sequential decision making problem, whose objective is to find an optimal policy through interaction with the environment. RL methods can be categorized into model-free and model-based methods, depending on whether the policy has access to the underlying model of the environment.

Model-free methods learn the policy directly from interactions with the environment [10]. Mnih *et al.* proposed deep Q network [5], which approximates $Q(s,a)$ with a neural network to find the best policy; Lillicrap *et al.* proposed deep deterministic policy gradient [22], which utilizes policy networks to generate actions while using Q-networks to criticize the policy. Model-free methods have achieved great success in many scenarios such as video games [23]. However, low sample efficiency limits the application of model-free methods.

Model-based methods simulate the transition probability of the environment, resulting in higher sample efficiency and faster convergence speed [10]. PILCO [24] utilizes Gaussian process to learn the environment dynamics and achieves good performance in nonlinear control tasks [25]. World model [26] utilizes mixture density network-recurrent neural network (MDN-RNN) to simulate the environment with raw images and outperforms previous methods in gym tasks.

### 2.3 Value Iteration-based Methods

For a Markov decision process (MDP) problem with known transition probability $P^a_{ss'}$, a dynamic programming method called value iteration [12] can give an optimal policy $\pi^*$. The following (1) and (2) are the basis of VI algorithm called the Bellman equations:

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P^a_{s,s'} V(s') \tag{1}$$

$$V'(s) = \max_a Q(s,a) \tag{2}$$

The optimal value map $V^*(s)$ and Q value map $Q^*(s,a)$ can be obtained by calculating between $V(s)$ and $Q(s)$ iteratively until convergence, after which the optimal policy $\pi^*$ can be obtained utilizing $\pi^* = \arg\max_a Q^*(s,a)$.

As is shown in Fig. 2, Tamar *et al.* [11] proposed *value iteration network* (VIN). VIN combines value iteration with convolution operators to plan paths with unknown environment dynamics. Niu *et al.* proposed generalized value iteration network (GVIN) [27], which is able to learn and plan paths on irregular spatial graphs. Lee *et al.* proposed gated path planning network (GPPN) [28], which replaces the unconventional recurrent update in VIN with a gated long short-term memory (LSTM) recurrent

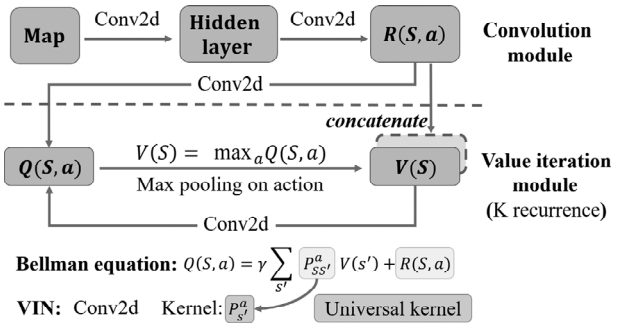



Figure 2. Structure of the *value iteration network* (VIN) [11].

operator to alleviate optimization issues such as random seed sensitivity.

## 3. Capability Iteration Network

### 3.1 Framework

The framework of CIN is illustrated in Fig. 3, which consists of two important modules: the capability module and the value iteration module. The input of CIN is a grid map $\mathcal{M}(m \times m)$, current state $s_t$ and the goal $s_g$. A sparse reward map $R(s)$ is generated with positive reward $r_p$ at goal state and small negative living cost $r_n$ otherwise.

The core component of CIN is a capability module where environment states are processed as state-conditioned transition probabilities. The value iteration module computes the Q-value map $Q(s, a)$ and value map $V(s)$ iteratively utilizing convolution with kernels given by the capability module and max-pooling, respectively. The optimal Q-value map $Q^*(s, a)$ obtained after $K$ iterations is utilized to find the best action $a^*$ for current state $s_t$ by maximization through the action space.

### 3.2 Capability Module

Capability module is the key component of CIN, which utilizes environment information to predict probability distribution of the next state $s_{t+1}$, *i.e.* $P(s_{t+1}|s_t, a)$, often denoted as $P^a_{ss'}$. The input of the capability module is an $F \times F$ local map, which contains information of state $s_{i,j}$ at the center, and states can be reached in one step from $s_{i,j}$. $F$ is a hyper-parameter for CIN, which is related to the agent. The output is the probability distribution of the next state from the $s_{i,j}$ with all the available actions, *i.e.*

$$f(\mathcal{M}_{[i,j,F]})_{k,l,a} = P\Big(s_{i-\frac{F-1}{2}+k, j-\frac{F-1}{2}+l} | s_{i,j,a}\Big), \quad 1 \le k, l \le F \tag{3}$$

where $f(\cdot)$ is the capability module, $X_{[i,j,F]}$ denotes the image patch centered at position $(i, j)$ with kernel size $F$. The capability module learns the capability of the agent, which determines the performance of the whole model. For path planning tasks in 2D and 3D mazes, a multilayer perceptron is enough; more complex classifiers such as xgboost [29] are necessary for more difficult tasks.

### 3.3 Value Iteration Module

The value iteration module utilizes convolution to implement the value iteration algorithm, which computes the Q-value map $Q(s, a)$ and value map $V(s)$ iteratively.

As is shown in Fig. 3, CIN utilizes convolution to simulate (1). The transition probabilities $P^a_{ss'}$ provided by

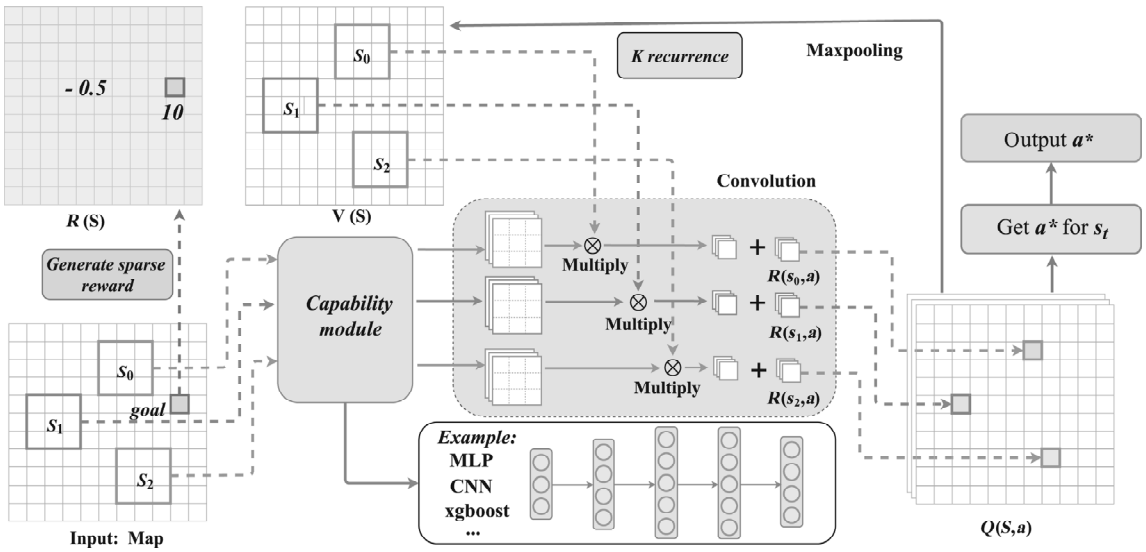



Figure 3. Structure of *capability iteration network* (CIN). The input of CIN is the map $\mathcal{M}$, current state $s_t$ and the goal position $s_g$, which is used to generate a sparse reward map. For each state $s_i$, CIN obtains the transition probability $P^a_{ss'}$ utilizing the capability module, after which CIN computes the Q-value map $Q(s, a)$ and $V(s)$ utilizing convolution and max-pooling iteratively with the Bellman equations until convergence. The best action $a^*$ is obtained utilizing max-pooling on the Q-value map.

the capability module are the convolution kernels, and the sparse reward $R(s)$ is the bias, *i.e.*

$$Q^{(k+1)}(s_{i,j},a) = \gamma f(\mathcal{M}_{[i,j,F]})_a \cdot V^{(k)}(s_{[i,j,F]}) + R(s_{i,j}), \quad 1 \le i,\ j \le m \tag{4}$$

Then the new value map $V(s)$ is obtained utilizing max-pooling illustrated in (2):

$$V^{(k+1)}(s_{i,j}) = \max_{a \in A} Q^{(k)}(s_{i,j},a), \quad 1 \le i,j \le m \tag{5}$$

The VI algorithm demonstrates that the value iteration module will output the optimal Q-value map $Q^*(s,a)$ after convergence:

$$\lim_{k \to \infty} Q^{(k)}(s,a) = Q^*(s,a) \tag{6}$$

In practical, the number of iteration $K$ is a hyper-parameter, which needs to be large enough to ensure convergence. Then the best action $a^*$ is obtained utilizing maximization through the action space:

$$a^* = \arg\max_{a \in A} Q^*(s_t,a) \tag{7}$$

Compared to VIN described in Fig. 2, CIN utilizes $P^a_{ss'}$ instead of $P^a_{s'}$ in VIN, which makes CIN completely consistent with the Bellman equation. This makes what CIN learns closer to the nature of the task compared with VIN. Furthermore, the sparse reward map used in CIN is simple to generate and avoids the accuracy of CIN being influenced by the defects in reward maps generated by the neural network, especially on large maps.

### 3.4 Training Methods

**End-to-end:** The whole network of CIN is differentiable, which means we can train CIN end-to-end using any RL or imitation learning (IL) [30] algorithms. The cross entropy loss is utilized in this experiment:

$$\mathcal{L}_{CE} = -\sum_{i=1}^{|A|} p_i \log \hat{p}_i, \quad p_i = \begin{cases} 1, & a_i = a^*, \\ 0, & a_i \ne a^*. \end{cases} \tag{8}$$

where $\hat{p}_i$ is the probability of the model to choose the action $a_i$.

**Train Capability Module Alone:** There is no parameter to train in the value iteration module. Thus, we can train the capability module alone by supervised learning. Take grid world as an example: trajectories of the agent are obtained through exploration. For each state $s_t$ on the trajectories, a $F \times F$ local map with center $s_t$ is sampled, which is utilized as the training data combined with the action $a_t$. The position of the next state $s_{t+1}$ is the label. The capability module is trained utilizing supervised learning with the mean squared error loss:

$$\mathcal{L}_{MSE} = \frac{1}{F^2} \sum_{k=1}^{F} \sum_{l=1}^{F} (f(\mathcal{M}_{[i,j,F]})_{k,l,a} - P(s_{i+k-\frac{2F-1}{2},\ j+l-\frac{2F-1}{2}} | s_{i,j}, a))^2 \tag{9}$$

## 4. Experiment and Discussion

In this section, we evaluate the performance of CIN in three types of experiment environments (2D, 3D grid world and real hexapod robot) and make a comparison with existing methods including VIN, GPPN and ResNet-based reactive policy to demonstrate the effectiveness and improvement of CIN.

### 4.1 2D Grid-world Domain

Grid world path planning is one of the most common environments for path planning problem. In 2D grid-world, the input map contains 0 (obstacles) and 1 (free-space) only. The size of the training dataset $N_{tr}$ is 10K; validation dataset size $N_{val}$ is 1K; test dataset size $N_{te}$ is 1K.

**CIN:** In 2D domain, we train the capability module alone with supervised learning. The sparse reward map consists of $+10$ at $s_g$ and $-0.5$ at other states. A multilayer perceptron is utilized for the capability module. Two baselines are utilized in the experiment.

**VIN** [11]: VIN is a classic VI-based model combined with VI algorithm and CNN. VIN achieves good performance in path planning problems for its planning computation.

**ResNet18** [31]: The problem setting for this task is similar to the image segmentation, in which problem each pixel in the given image needs to be assigned a label (the best action choice in our case). Thus, ResNet18 can be seen as a reactive path planning model in our experiments.

Experiments are conducted on mazes with four sizes ($m \times m$): $8 \times 8$, $15 \times 15$, $28 \times 28$ and $45 \times 45$. Two metrics are utilized to evaluate the performance, including **%Optimal**(%Opt)—the percentage of states whose predicted paths under the policy estimated have optimal length and **%Success**(%Suc)—the percentage of states whose predicted paths under the policy estimated reach the goal state.

Table 1 shows the performance of models on each metric. The bold values means the corresponding method achieves the best performance in the experiment. Two conclusions can be concluded: (1) CIN outperforms other models in 2D domain, especially on big maps. (2) Models with planning computation have better performance on big maps, compared to reactive policies (ResNet18).

Table 1
Experimental Results in 2D Grid World with Varying Map Size ($m \times m$)

| $m$ | VIN | | GPPN | | CIN | | ResNet18 | |
|---|---|---|---|---|---|---|---|---|
| | %Opt | %Suc | %Opt | %Suc | %Opt | %Suc | %Opt | %Suc |
| 8 | 98.9 | 99.3 | 98.6 | 99.2 | **99.9** | **99.9** | 99.7 | **99.9** |
| 15 | 89.1 | 90.5 | 97.6 | 98.4 | **99.7** | **99.8** | 97.4 | 98.2 |
| 28 | 80.9 | 81.0 | 96.3 | 96.7 | **99.5** | **99.5** | 86.5 | 91.8 |
| 45 | 63.3 | 72.1 | 92.3 | 95.2 | **99.0** | **99.1** | 54.0 | 66.3 |

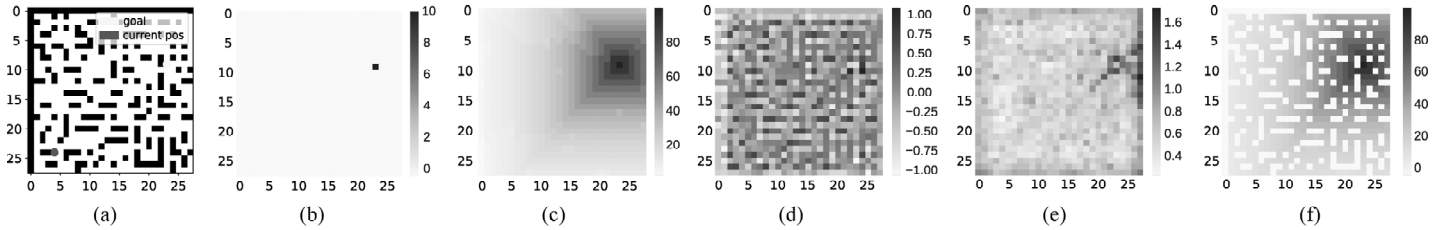



Figure 4. Visualization of the maze, reward maps and value maps: (a) maze; (b) CIN $R(S)$; (c) CIN $V(S)$; (d) VIN $R(S)$; (e) VIN $V(S)$; and (f) VI $V(S)$.

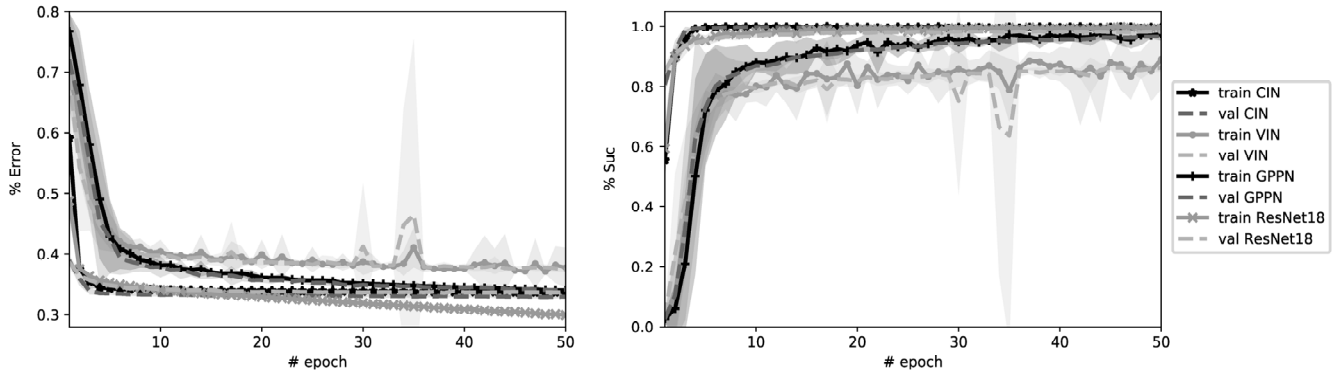



Figure 5. The learning curve of each model on the $15 \times 15$ 2D mazes.

### 4.2 Model Result Visualization

The visualization of the model results is depicted in Fig. 4. The reward map used in CIN (Fig. 4(b)) is sparse, which is consistent with the VI algorithm. Reward map of VIN (Fig. 4(d)) is generated by CNN, which is more messy, sensitive to the map settings and may influence the accuracy of VIN on unseen mazes. Compared to VIN (Fig. 4(e)), value map of CIN (Fig. 4(c)) is sharp at the edges of obstacles with obvious tendency toward goal point and is generally consistent with the optimal value map (Fig. 4(f)).

### 4.3 Learning Speed and Stability

The learning curve of each model on the $15 \times 15$ 2D mazes is depicted in Fig. 5. CIN is trained end-to-end in this experiment for fairness. All the models are trained utilizing at least five different random seeds with training dataset size $N_{tr} = 10$K.

The **%Error** in Fig. 5 is the percentage of states whose predicted action is different with the action given by the expert. As shown in the figure, CIN and ResNet18 have faster learning speed compared to GPPN and VIN, because CIN utilizes sparse reward immediately while VIN and GPPN use CNN with more parameters to train. However, ResNet18 appears overfitted because it belongs to reactive policy without planning computation. Furthermore, GPPN utilizes gated LSTM recurrent operator, which alleviates optimization issues compared to VIN. Above all, CIN has better random seed sensitivity and converges more stably than other models.

Table 2
Experimental Results in 3D Grid World

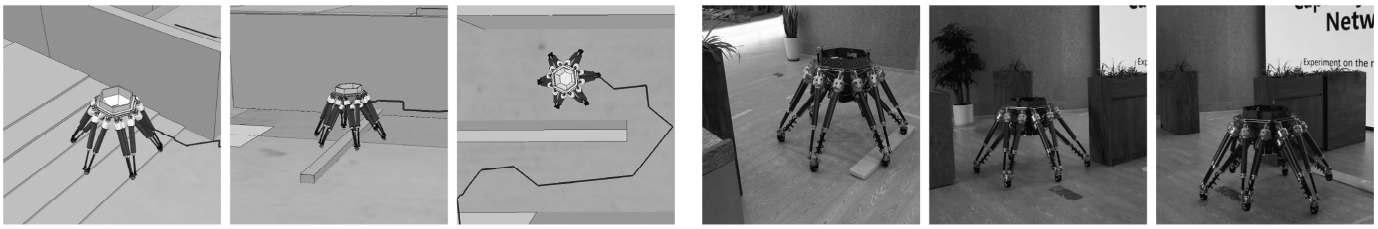
| $m$ | VIN | | GPPN | | CIN | | ResNet18 | |
|---|---|---|---|---|---|---|---|---|
| | %Opt | %Suc | %Opt | %Suc | %Opt | %Suc | %Opt | %Suc |
| 8 | 92.0 | 93.7 | 93.6 | 94.0 | **95.2** | **96.5** | 94.0 | 94.2 |
| 15 | 83.7 | 86.8 | 89.3 | 89.7 | **93.7** | **94.3** | 79.1 | 81.3 |
| 28 | 74.1 | 76.5 | 85.3 | 87.5 | **89.6** | **89.9** | 56.2 | 60.0 |
| 45 | 60.9 | 63.1 | 80.3 | 81.5 | **85.3** | **87.6** | 47.3 | 51.3 |

### 4.4 3D Terrain Map

Unlike maps in 2D domain, matrix elements in 3D terrain maps can be any float value, representing height in a specific position. The agent can only walk to its neighborhoods if the height difference is smaller than a certain value. Compared to 2D mazes, 3D terrain maps are more complex and challenging for path planning models. As shown in the Table 2, CIN still achieves better performance than VIN and GPPN on various metrics and map settings, especially on big maps.

### 4.5 Real Hexapod Robot

**Hexapod robot:** The hexapod robot has the characteristics of good stability, large carrying capability and flexibility. The robot used in this experiment is a parallel structure robot, also known as a parallel–parallel (PP) structure six-legged robot.



(a) (b)

Figure 6. Experimental results with the hexapod robot: (a) Simulation in CoppeliaSim and (b) experiment with the real robot.

**Adaption to hexapod robot:** Currently, VI-based algorithms are only effective for discrete state and action space, which means discretization is needed. Given a real environment, we construct a 3D map with SLAM algorithm, after which the 2D map with $0.2\,m \times 0.2\,m$ squares is obtained by projection. The coordinate of the square in which the robot mass center lies is utilized as the current state $s_t$.

To simplify the large continuous action space, a triangle-gait is designed. Six legs of the robot are divided into two groups with three non-adjacent feet as a group. The robot takes three feet first and then the other ones for a walk cycle. The gait is defined as (step length, step height and rising length) with eight walking directions. Available combinations are $(0.1\,m, 0.3\,m, 0.3\,m)$, $(0.2\,m, 0.2\,m, 0.2\,m)$ and $(0.3\,m, 0.1\,m, 0.1\,m)$.

As shown in Fig. 6(a), a simulation experiment is conducted in CoppeliaSim (V-rep) [32]. The robot can step up stairs, make U-turn and handle uneven surface, utilizing the path planned by CIN. An experiment in the real environment is also conducted. As shown in Fig. 6(b), CIN trained in the simulation experiment is utilized to the real robot, which also achieves good performance.

## 5. Conclusion

In this paper, we propose a new VI-based path planning model, *capability iteration network*. CIN utilizes sparse reward maps and state-conditioned transition probabilities, which make CIN consistent with the Bellman equations. Several path planning experiments including on 2D and 3D grid world demonstrate the improvement of CIN compared with previous algorithms. Furthermore, the experiment on the real hexapod robot demonstrates that CIN is able to be utilized for path planning tasks on real robots.

## Acknowledgement

This work is sponsored by the National Natural Science Foundation of China (Grant Nos. 61903247 and 51805318).

## References

[1] T.T. Mac, C. Copot, D.T. Tran, and R. De Keyser, Heuristic approaches in robot path planning: A survey, *Robotics and Autonomous Systems*, *86*, 2016, 13–28.

[2] E. Plaku, L.E. Kavraki, and M.Y. Vardi, Motion planning with dynamics by a synergistic combination of layers of planning, *IEEE Transactions on Robotics*, *26*(3), 2010, 469–482.

[3] P.E. Hart, N.J. Nilsson, and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 1968, 100–107.

[4] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in I.J. Cox and G.T. Wilfong (eds.), *Autonomous robot vehicles* (Berlin: Springer, 1986), 396–404.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, Human-level control through deep reinforcement learning, *Nature*, *518*(7540), 2015, 529.

[6] M. Sadeghzadeh, D. Calvert, and H.A. Abdullah, Autonomous visual servoing of a robot manipulator using reinforcement learning, *International Journal of Robotics and Automation*, *31*(1), 2016, 26–28.

[7] T. Yan, W. Zhang, S.X. Yang, and L. Yu, Soft actor-critic reinforcement learning for robotic manipulator with hindsight experience replay, *International Journal of Robotics Automation*, *34*(5), 2019, 536–543.

[8] Y. Liu, M. Cong, H. Dong, and D. Liu, Reinforcement learning and ega-based trajectory planning for dual robots, *International Journal of Robotics and Automation*, *33*(4), 2018, 367–378.

[9] M. Pflueger, A. Agha, and G.S. Sukhatme, Rover-IRL: Inverse reinforcement learning with soft value iteration networks for planetary rover path planning, *IEEE Robotics and Automation Letters*, *4*(2), 2019, 1387–1394.

[10] K. Arulkumaran, M.P. Deisenroth, M. Brundage, and A.A. Bharath, A brief survey of deep reinforcement learning, arXiv:1708.05866, 2017.

[11] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, Value iteration networks, *Advances in Neural Information Processing Systems*, Barcelona, Spain, 2016, 2154–2162.

[12] R. Bellman, Dynamic programming, *Science*, *153*(3731), 1966, 34–37.

[13] M. Riedmiller, R. Hafner, T. Lampe, *et al.*, Learning by playing solving sparse reward tasks from scratch, *Int. Conf. on Machine Learning*, Stockholm, Sweden, 2018, 4344–4353.

[14] B. Paden, M. Čáp, S.Z. Yong, D. Yershov, and E. Frazzoli, A survey of motion planning and control techniques for self-driving urban vehicles, *IEEE Transactions on Intelligent Vehicles*, *1*(1), 2016, 33–55.

[15] G. Che, L. Liu, and Z. Yu, An improved ant colony optimization algorithm based on particle swarm optimization algorithm for path planning of autonomous underwater vehicle, *Journal of Ambient Intelligence and Humanized Computing*, *11*(8), 2020, 3349–3354.

[16] P.K.Y. Yap, N. Burch, R.C. Holte, and J. Schaeffer, Any-angle path planning for computer games, *Seventh Artificial Intelligence and Interactive Digital Entertainment Conf.*, Palo Alto, CA, USA, 2011.

[17] S. Koenig and M. Likhachev, D* lite, *Eighteenth national Conf. on Artificial intelligence*, CA, USA, 2002, 476–483.

[18] J. Wang, X.-J. Lin, H.-Y. Zhang, G.-D. Lu, Q.-L. Pan, and H. Li, Path planning of manipulator using potential field combined with sphere tree model, *International Journal of Robotics and Automation*, *35*(2), 2020, 148–161.

[19] S.M. LaValle, Rapidly-exploring random trees: A new tool for path planning, Technical Report, Computer Science Department, Iowa State University, Ames, USA, 1998.

[20] Y. Liu, M. Cong, H. Dong, D. Liu, and Y. Du, Time-optimal motion planning for robot manipulators based on elitist genetic algorithm, *International Journal of Robotics and Automation*, *32*(4), 2017, 396–405.
[21] J. Xin, J. Zhong, J. Sheng, P. Li, and Y. Cui, Improved genetic algorithms based on data-driven operators for path planning of unmanned surface vehicle, *International Journal of Robotics and Automation*, *34*(6), 2019, 713–722.
[22] T.P. Lillicrap, J.J. Hunt, A. Pritzel, *et al.*, Continuous control with deep reinforcement learning, arXiv:1509.02971, 2015.
[23] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, Playing atari with deep reinforcement learning, arXiv:1312.5602, 2013.
[24] M. Deisenroth and C.E. Rasmussen, PILCO: A model-based and data-efficient approach to policy search, *Proc. of the 28th Int. Conf. on Machine Learning (ICML-11)*, Bellevue, Washington, 2011, 465–472.
[25] R. McAllister and C.E. Rasmussen, Data-efficient reinforcement learning in continuous state-action gaussian-pomdps, *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, 2017, 2040–2049.
[26] D. Ha and J. Schmidhuber, Recurrent world models facilitate policy evolution, in *Advances in Neural Information Processing Systems*, Montreal, Canada, 2018, 2450–2462.
[27] S. Niu, S. Chen, H. Guo, C. Targonski, M.C. Smith, and J. Kovačević, Generalized value iteration networks: Life beyond lattices, *Thirty-Second AAAI Conf. on Artificial Intelligence*, New Orleans, LA, USA 2018.
[28] L. Lee, E. Parisotto, D.S. Chaplot, E. Xing, and R. Salakhutdinov, Gated path planning networks, *Int. Conf. on Machine Learning*, Stockholm, Sweden, 2018, 2947–2955.
[29] T. Chen and C. Guestrin, XGBoost: A scalable tree boosting system, *Proceedings of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, CA, USA, 2016, 785–794.
[30] P. Abbeel and A.Y. Ng, Apprenticeship learning via inverse reinforcement learning, *Proc. of the Twenty-first Int. Conf. on Machine Learning*, Alberta, Canada, 2004, 1.
[31] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, WA, USA, 2016, 770–778.
[32] E. Rohmer, S.P. Singh, and M. Freese, V-REP: A versatile and scalable robot simulation framework, *2013 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Tokyo, Japan, 2013, 1321–1326.

## Biographies

*Buqing Nie* is currently a Postgraduate in the Department of Automation at Shanghai Jiao Tong University, Shanghai, China. His main research interest is reinforcement learning and motion planning.

*Yue Gao* is currently an Associate Professor in the AI Institute, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. She received her Ph.D. degree from Cornell University in Computer Science. Her research focuses on legged robot intelligent control, including sim-to-real reinforcement learning for legged robot locomotion, force control in unstructured environment and trajectory planning for legged robot in unknown environment.

*Yidong Mei* is currently a Postgraduate in the Department of Automation at Shanghai Jiao Tong University, Shanghai, China. His main research interest is deep reinforcement learning and agent motion planning.

*Feng Gao* is a Professor at Shanghai Jiao Tong University. He received his M.E. degree in Mechanical Engineering from Northeast Heavy Machinery Institute (now Yanshan University), Qiqihar, China, in 1982 and his Ph.D. degree in Mechanical Engineering from Beijing University of Aeronautics and Astronautics, Beijing, in 1991. From 1995 to 1999, he worked as a Postdoctoral Research Fellow in Simon Fraser University, Canada and Yanshan University. His research areas include the parallel manipulator and its application, mechanical design and robotics. He received the National Science Fund for Distinguished Young Scholars, in 2001.