Received: 11 January 2021 | Revised: 19 February 2021 | Accepted: 7 April 2021

IET Cyber-Systems and Robotics

DOI: 10.1049/csy2.12020

ZHEJIANG UNIVERSITY PRESS | IET The Institution of Engineering and Technology | WILEY

REVIEW

A survey of learning-based robot motion planning

Jiankun Wang[1] | Tianyi Zhang[1] | Nachuan Ma[1] | Zhaoting Li[1] | Han Ma[2] | Fei Meng[2] | Max Q.-H. Meng[1,2,3]

[1]Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, China

[2]Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, China

[3]Shenzhen Research Institute of the Chinese University of Hong Kong, Shenzhen, China

Correspondence

Max Q.-H. Meng, 1088 Xueyuan Avenue, Shenzhen 518055, China.
Email: max.meng@ieee.org

Funding information

National Key R&D program of China, Grant/Award Number: 2019YFB1312400; Hong Kong RGC CRF grant, Grant/Award Number: #C4063-18GF; Hong Kong RGC TRS grant, Grant/Award Number: #T42-409/18-R; Hong Kong RGC GRF grant, Grant/Award Number: #14200618; Shenzhen Science and Technology Innovation projects: JCYJ20170413161503220

Abstract

A fundamental task in robotics is to plan collision-free motions among a set of obstacles. Recently, learning-based motion-planning methods have shown significant advantages in solving different planning problems in high-dimensional spaces and complex environments. This article serves as a survey of various different learning-based methods that have been applied to robot motion-planning problems, including supervised, unsupervised learning, and reinforcement learning. These learning-based methods either rely on a human-crafted reward function for specific tasks or learn from successful planning experiences. The classical definition and learning-related definition of motion-planning problem are provided in this article. Different learning-based motion-planning algorithms are introduced, and the combination of classical motion-planning and learning techniques is discussed in detail.

1 | INTRODUCTION

Motion planning is essential for robot deployment in practical applications [1], including industrial [2], surgical [3], autonomous driving [4] and home service robots [5]. Many algorithms have been proposed to address motion-planning problems, such as A* [6], Artificial Potential Field (APF) [7], and Rapidly exploring Random Tree (RRT) [8]. These conventional algorithms can achieve convincing performance either in a general class of problems or under specified scenarios. However, they also suffer some limitations. A* algorithms scale badly in high-dimensional planning problems. In addition, the solutions from A* are *resolution complete*, which means that the solution quality depends on the discretisation of the current environment. APF algorithms often end up at a local minimum and cannot guarantee a globally optimal solution. RRT-based algorithms are very sensitive to the sampling distribution, so the quality of the initial solution and the time used to converge to the optimal solution cannot be guaranteed.

Recently, learning-based methods have begun to show their efficiency at solving motion-planning problems. They either utilise a human-crafted reward function to guide the robot movement or learn feasible solutions from previously successful planning experiences. Generally, the learning-based methods applied to robot motion planning can be classified as supervised learning, unsupervised learning, and reinforcement learning.

Therefore, this survey will

- introduce the learning techniques applied to the robot motion-planning problem;
- provide definitions of classical and learning-based motion-planning problems;
- discuss the existing learning-based motion-planning algorithms.

The rest of this article is organised as follows. A formal definition of robot motion planning is given in Section 2. Sections 3, 4 and 5 present, respectively, supervised, unsupervised and reinforcement learning based robot motion-planning methods. Section 6 presents conclusions for learning-based robot motion-planning.

2 | DEFINITION OF ROBOT MOTION PLANNING

2.1 | Classical formulation

The basic motion-planning problem is defined as follows. Given:

- $\mathcal{C}$: configuration space;
- $\mathcal{C}_{obs}$: configuration space obstacle region;
- $\mathcal{C}_{free} = \mathcal{C} \backslash \mathcal{C}_{obs}$: free space;
- $q\ (x_{init})$: configuration q corresponding to the initial state x; and
- $q\ (x_{goal})$: configuration q corresponding to the goal state x;

compute a time T and a set of controls $u : [0, T] \rightarrow \mathcal{U}$ such that $x(T) = x_{goal}$ and $q(x(t)) \in \mathcal{C}_{free}$, $\forall t \in [0, T]$. In integral form, it can be written as

$$\begin{aligned} x(T) &= x(0) + \int_0^{T-1} f(x(t), u(t))dt \\ s.t.\ x(t+1) &= x(t) + f(x(t), u(t)), \\ x(0) &= x_{init}, \\ x(T) &= x_{goal}, \\ q(x(t)) &\in \mathcal{C}_{free}, \forall t \in [0, T]. \end{aligned} \tag{1}$$

It should be noted that path planning is a subproblem of motion-planning, which is a purely geometric problem without respect to robot dynamics or control constraints. The aim of path planning is to find a collision-free path $\sigma : [0, T] \rightarrow \mathcal{C}_{free}$ such that $\sigma(0) = x_{init}$ and $\sigma(T) = x_{goal}$.

2.2 | Modularised formulation

When concerning learning-based techniques, a typical motion planning framework can be decomposed into the following modules, as shown in Figure 1:

- Preprocessing module: $\mathcal{H} : \mathcal{C} \rightarrow \mathcal{C}_{pro}$. This module takes the current configuration space and extra data from sensors as input and outputs the processed configuration space. The purpose of this module includes extracting the subspace from the whole configuration space to improve search efficiency, encoding the configuration space to another space that is much easier to conduct planning, representing obstacles with low dimensional data. This module tends to be implemented at the beginning of the motion-planning algorithm;
- Prediction module: $\mathcal{P} : \mathcal{U} \times \mathcal{X} \rightarrow \mathcal{C}_{pro}$. The objective of this module is similar to the preprocessing module, while the difference is that the prediction module is implemented many times during the motion-planning process;
- Executing module: $\mathcal{E} : \mathcal{C}_{pro} \times \mathcal{U} \rightarrow \mathcal{X}$. This module selects an action from action space $\mathcal{U}$ according to the current robot configuration $\mathcal{C}_{pro}$, and then a new state is generated;
- Collision-checking module: $\mathcal{O} : \mathcal{C}_{pro} \times \mathcal{X} \rightarrow \{True, False\}$. This module checks whether the new state will collide with the obstacle region in the configuration space.

Among the four modules mentioned above, the executing and collision-checking modules are necessary for all the motion-planning algorithm, while the other two modules are optional. When utilising deep learning to tackle motion-planning problems, neural networks can replace one or more of these modules and serve as a mapping function. Using neural networks to replace all modules is called an end-to-end framework.

2.3 | Markov Decision Process formulation

In Reinforcement Learning (RL) applications, a motion-planning problem is formulated as a Markov Decision Process (MDP). In this section, the basic components of an MDP representation in a motion-planning setting are listed.

The basic MDP can be modelled as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where:

- $\mathcal{S}$ is a state space that includes the configuration space of the robot, and the observation of the environment (e.g. a 2D or 3D map in Cartesian space, data from the robot's sensor). In a general MDP, a state $s_t \in \mathcal{S}$ satisfies the Markov property;

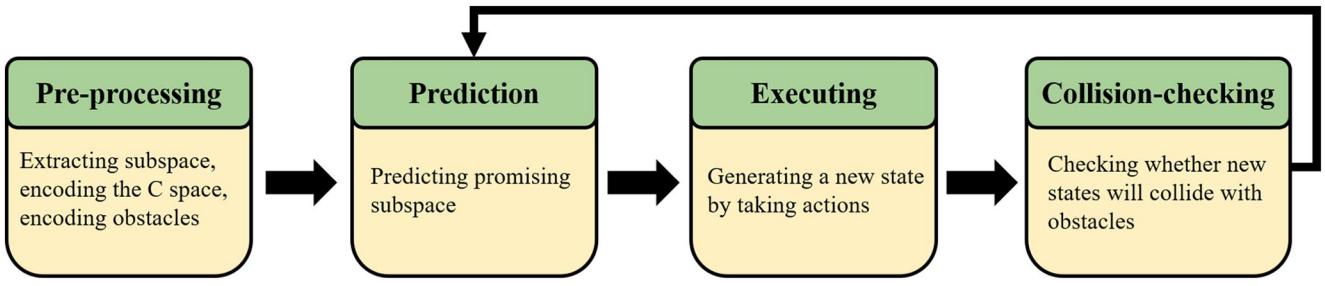

FIGURE 1 The framework of a typical motion-planning algorithm

- $\mathcal{A}$ is action space, $\forall a_t \in \mathcal{A}$, similar to the control space $\mathcal{U}$ defined in Section 2.1;
- $\mathcal{P}$ is a state transition probability; when the robot takes an action a_t from the action space, the robot state s_t will transition to state s_{t+1} according to the state transition probability $\mathcal{P}(s_{t+1}|s_t, a_t)$. The transition probability is either deterministic or stochastic when considering uncertainties;
- $\mathcal{R}$ is a reward function. Giving the robot a reward after an action a_t is taken is the main method in RL to help the robot know whether an action is good at a certain state s_t. The reward $\mathcal{R}(s_t, a_t)$ is similar to the cost function in classical motion-planning is used to describe the desired behaviour;
- γ is a discount factor, $\gamma \in [0, 1]$, that adjusts the value of the reward.

Assume a robot is in the state $s_t \in \mathcal{S}$ at time step t. If the robot takes an action $a_t \in \mathcal{A}$, the robot state at time step $t + 1$ will transfer to s_{t+1} by the state transition probability $\mathcal{P}(s_{t+1}|s_t, a_t)$, while the robot will receive an immediate reward $r_t = \mathcal{R}(s_t, a_t)$. The objective of RL algorithms is to learn a policy $\pi(a|s)$ that can select actions to maximise the long-term reward, which is called the return/accumulate reward. The return/accumulate reward is usually defined as $\gamma-$discounted accumulated future rewards over time. In value-based RL methods, the return/accumulate reward can be defined as follows:

- $V_\pi(s)$ is a state-value function that represents the return/accumulate reward starting from state s under policy π: $V_\pi(s) = \mathbb{E}[\sum_{i=t}^{\infty} \gamma^{i-t} r_i | s_t]$;
- $Q_\pi(s, a)$ is an action-value function that represents the return/accumulate reward starting from state s and taking action a under policy π: $Q_\pi(s, a) = \mathbb{E}[\sum_{i=t}^{\infty} \gamma^{i-t} r_i | s_t, a_t]$.

The policy can be obtained by choosing the action with the highest action value. In policy-based RL methods, the policy can be directly represented by a neural network:

- $\pi_\theta(a_t|s_t)$, which takes state s as input and outputs the optimal action directly.

3 | SUPERVISED LEARNING BASED MOTION PLANNING

Researchers have proposed many supervised learning-based motion-planning methods in recent years, which can be divided into roughly two categories: (i) learn to completely replace the entire classical motion planner pipeline and (ii) learn to improve one or two existing components of classical motion-planning algorithms. The first-category methods learn to establish entire systems that generate end-to-end collision-free paths or generate trajectories in the next step for the given configuration space directly. The second-category methods learn to improve subsystems of a motion-planning framework, including the preprocessing module, prediction module, executing module, and collision-checking module.

3.1 | End-to-end algorithms

In Ref. [9], Pfeiffer proposed a data-driven end-to-end motion-planning approach for autonomous ground robots that learns from expert demonstrations obtained in simulation. The architecture consists of a Convolutional Neural Network (CNN) part that extracts information about the input laser data and Fully Connected layers that combine extracted feature maps and target positions to generate the required navigation commands. Similarly, Hamandi et al. [10] processed the information of LiDAR scans to predict the speed and direction for robots by a novel Deep Neural Network (DNN) model that utilises Long Short-Term Memory (LSTM) layers and adopts ResNet [11] as the backbone.

To determine end-to-end collision-free trajectories in an iterative manner, algorithms in the works of [12–15] have been raised. Bency et al. [12] introduced a Recurrent Neural Networks (RNNs)-based motion-planning algorithm called OracleNet that can determine end-to-end collision-free trajectories for static environments and generate near-optimal paths iteratively. In Ref. [13], Kurutach presented a causal InfoGAN model inspired by Generative Adversarial Networks (GANs) [16] and their variants, which learns to generate a sequence of high-dimensional feasible observations to guide from an initial configuration to a goal configuration. Unlike the two algorithms previously mentioned, Qureshi et al. [14] proposed a DNN-based iterative motion-planning algorithm called Motion-Planning Networks that can be evaluated under multiple motion-planning cases such as the planning of a seven Degree-Of-Freedom Baxter robot manipulator. Its architecture includes an encoder network part and a planning network part. Through the encoder network, a point cloud of the surroundings is encoded to a latent space fed into the planning network to generate predicted end-to-end collision-free paths from the start configuration to the goal configuration. The results show that it can generate feasible trajectories within 1 s in presented experiments, which is lower than existing state-of-the-art motion-planning algorithms. In Ref. [15], Qureshi expanded this idea more formally and validated it in more challenging and cluttered environments, demonstrating better performance metrics of MPNet. Ichter et al. [17] proposed a latent sampling-based motion-planning algorithm that replaces the modules of classical sampling-based motion-planning algorithm accordingly. It constructs the learnt latent space through autoencoding, dynamic, and collision-checking networks that correspond to the sampling, local connecting, and collision-checking parts of classical sampling-based motion-planning algorithms. In addition, Huh et al. [18] combined the cost-to-go function with supervised learning techniques to propose a c2g-HOF network for a high-dimensional motion-planning problem. The c2g-HOF requires supervision only in the training step and can generate a continuous cost-to-go function directly from sensor inputs. Near-optimal trajectories can be obtained by computing the gradient of the generated cost-to-go function.

3.2 | Module replacement algorithms

In addition to the aforementioned end-to-end and supervised learning-based motion-planning methods, many hybrid learning-based motion-planning algorithms have emerged in the past few years. These hybrid algorithms integrate state-of-the-art deep learning techniques and classical motion-planning algorithms to enhance specific modules of classical motion-planning algorithms.

Deep learning techniques such as Conditional Variational AutoEncoder (CVAE), CNN, GAN and their variants have been widely used to solve motion-planning algorithms by generating processed configuration space in advance to guide the expansion of classical motion-planning algorithms [19–30].

3.2.1 | CNN-related pre-processing module

Wang et al. [19] proposed a novel learning-based optimal path planning algorithm called Neural RRT*, which aims to utilise a non-uniform sampling distribution to improve the performance of classical optimal RRT (RRT*) algorithm. For a given map, the pretrained CNN model can predict the probability distribution of the optimal path, which is used to guide the sampling process of the RRT* algorithm. The CNN model is trained by using a large number of successful cases generated by the A* algorithm. The experimental results show that the proposed Neural RRT* algorithm achieves better performance than traditional sampling-based path planning algorithms such as the RRT* algorithm significantly in terms of time cost and memory usage. The proposed CNN model consists of an encoding part and a decoding part with atrous convolution [31], and adopt ResNet50 [11] as the backbone. To improve the performance of the classical Probabilistic Roadmap (PRM) method, Ichter et al. [30] proposed a novel critical PRM algorithm that trains a CNN model to predict critical samples identified via betweenness centrality method from local environment features to guide the sampling process of classical PRM method. Some researchers have expanded the idea of guiding the expansion of motion planners using CNNs model in high-dimensional configuration space. Naman et al. [20] trained a CNN model with U-Net [32] as the backbone to predict promising states of the given environment and the sampling distribution for robot joints. In Ref. [23], a CNN model is combined with a classical graph search algorithm to predict an optimal sampling distribution for a 31 degree-of-freedom mobile robot. Some researchers also utilise Fully Convolutional Networks (FCNs) to improve the performance of classical motion-planning algorithms. Higueras et al. [24] proposed a learning-based human-aware path planning algorithm by integrating FCNs and classical RRT* algorithm. The algorithm trains FCNs to predict the cost-map and relevant features for robot navigation by learning expert demonstrations. It outperforms two Inverse Reinforcement Learning (IRL) algorithms [33, 34] in experiments. In addition, Ariki et al. [25] trained an FCN model to predict a search heuristic image tha- is utilised to enhance the performance of classical learning method such as backward Dijkstra by imitating the predicted cost-to-go values. Apart from guiding the sampling process for sampling-based motion-planning algorithms, FCNs can also be used to bias the expansion of grid-based algorithms. In Ref. [21], Yonetani proposed a novel search-based algorithm called Neural A*, which integrates a fully convolutional encoder that estimates the movement cost map and a differentiable A* module to generate an optimal path.

3.2.2 | CVAE & GAN-related pre-processing module

CVAE [35] and GAN [16] are popular deep learning techniques in the field of computer vision and artificial intelligence as they are capable of generating new data with expected features. In Ref. [26], Ichter et al. utilised a CVAE model to generate heuristic samples through the decoder network for sampling-based motion planners, which is then used to guide the sampling process. The CVAE model has been trained in many successful motion-planning cases. Compared with traditional uniform sampling-based motion planners, the proposed method can achieve significantly better performance in the aspects of success rate and convergence time to the optimal path. However, the proposed CVAE-based model in Ref. [26] may have bad performance when encountering complex obstacle configurations or mismatch between training and testing. To overcome the limitation, Kumar et al. [27] proposed a novel algorithm leveraging experience in roadmap generation for sampling-based planning called LEGO. The LEGO algorithm trains the CVAE model using target samples in bottleneck regions and diverse regions to plan with sparse road maps. Takahashi et al. [28] applied a GAN model with U-Net as the backbone to learning heuristic functions for grid-based A* algorithm that can reduce the search cost in 2-D space, whereas Ma et al. [29] utilised the similar architecture to guide the sampling process of classical sampling-based RRT* algorithm that can accelerate the convergence speed to the optimal path significantly and generate a high-quality initial path. Figure 2 illustrates the experimental results of four maps presented in Ref. [29]. The red and blue circles represent the specified start point and goal point. The green-yellow region represents the predicted probability distribution of feasible paths generated from the GAN model, and the optimal path is highlighted in medium-orchid.

3.2.3 | Prediction module

In recent years, researchers have proposed many learning-based algorithms to predict the structure of the unknown configuration space and feasible trajectories in the motion planning process. Elhafsi et al. [36] presented a map predictive motion-planning algorithm which integrates map prediction and motion-planning for safe and efficient robot navigation. Conditional neural process architecture is utilised to predict the map of the unobserved environment, and the

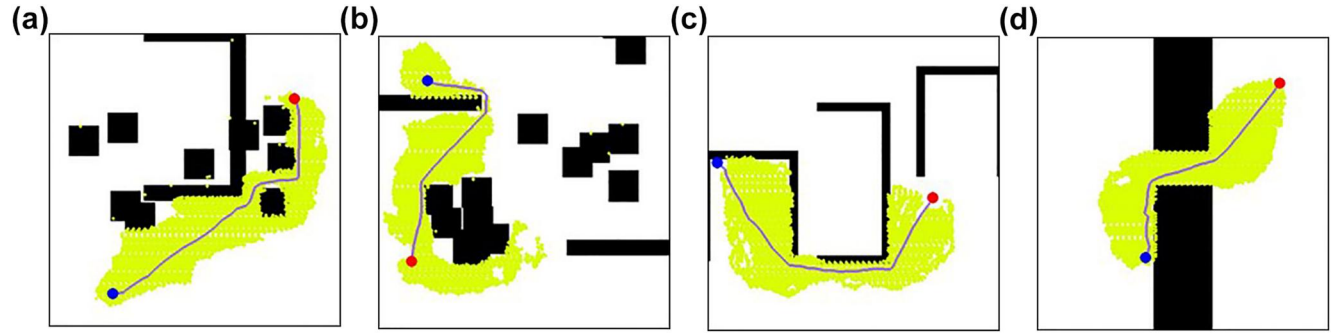

FIGURE 2 The experimental results in Ref. [29] (Reprinted with permission) illustrate the optimal paths generated with a predicted probability distribution for four types of maps. The red and blue circles denote the start point and goal point, respectively. The green-yellow region represents a probability distribution generated from the GAN model, and the medium-orchid represents the optimal path. (a): Map1. (b): Map2. (c): Map3. (d): Map4

prediction is used to guide trajectory generation from the start state to the goal state. Qureshi et al. [37] proposed a deep sampling-based motion planner model to generate samples in the areas that may contain path solutions for sampling-based motion-planning algorithms. Its architecture consists of two parts. One is a Contractive AutoEncoder (CAE) that encodes the information of a given workspace, and another is a Dropout-based stochastic deep feed-forward neural network that generates heuristic samples in an iterative way from the workspace encoding and the initial and the goal states. The proposed model outperforms the existing advanced sampling-based motion-planning algorithms such as the Informed-RRT* [38] and BIT* [39] remarkably in the experimental cases of the point-mass robot, rigid-body, and 6-link robotic manipulator. Algorithms in [40–42] utilised deep learning algorithms to predict trajectories for pedestrian, surrounding vehicles, and heterogeneous traffic-agents in an iterative manner.

3.2.4 | CAE-related collision-checking module

To enhance the performance of the collision-checking module of classical motion-planning algorithms, many learning-based algorithms have been proposed. Kew et al. [43] proposed a collision-checking heuristic algorithm called CN-RRT, which contains ClearanceNet learning to predict the minimum distance between the robot and workspace, and guides the classical RRT algorithm to generate feasible paths by leveraging features extracted from the ClearanceNet. The experimental results reveal that the proposed CN-RRT algorithm can produce shorter paths more quickly than classical path planners. Tran et al. [44] proposed a novel framework to predict sample collision based on neural networks. The framework consists of two parts, one is a CAE part like the Deep Sampling-based Motion Planner model in Ref. [37] that is utilised to obtain an occupied grid representation of the configuration space, and another is a Multi-layer Perceptron part that can predict the collision state of the robot in an efficient way from the information extracted by the CAE part. Moreover, researchers in [45–47] have utilised semantic segmentation techniques to classify each pixel on the image as either positive or negative to obtain collision-free space for self-driving vehicles.

3.2.5 | Executing

Apart from the aforementioned supervised learning-based algorithms, researchers have also proposed several methods to replace the executing module of classical motion-planning algorithms. Soonkyum et al. [48] presented a novel efficient graph search algorithm called learning heuristic A*, which replaces the heuristic function of the classical A* algorithm as a neural network to guide the expansion of vertices and reduce redundant explorations. Similarly, Guzzi et al. [49] presented a novel data-driven motion-planning algorithm combined with a classical sampling-based motion planner. It learns an estimator to predict the outcome of motions which connect two nearby states and utilises the learnt estimator to identify feasible motions with maximum success probability. And the proposed algorithm shows better performance than classical sampling-based motion planners in the simulation of generating both short moves and long-range paths. Garimella et al. [50] utilised an RNN model to replace the connecting mechanism of autonomous vehicles, which is combined with a Nonlinear Model Predictive Control module to output connecting commands. Researchers of Ref. [51, 52] have shown how to predict a steering direction for a Predator Robot and a car with the application of CNN models.

3.2.6 | Other learning-based methods

Supervised learning is also combined with a knowledge-based method to improve traditional motion-planning algorithms. Knowledge-based methods, such as expert and fuzzy systems, use a set of rules to do induction for the motion planner. In motion planning, the most typical application is fuzzy neural networks, which use neural networks to find the parameter of the fuzzy system. Chen et al. [53] introduced fuzzy-kinodynamic RRT to generate heuristic rules based on the traditional RRT algorithm. Jaradat et al. [54] used the fuzzy

logic expert system to improve the APF method in a dynamic environment. This method develops expert if-then rules to present attractive and repulsive forces, which provides the robot with the most appropriate heading towards a stationary or moving target. Zhu et al. [55] proposed a recurrent fuzzy neural network method for motion planning. The recurrent fuzzy neural network generates control instructions for turning left or right according to the obstacle information and its own position, thereby finding a collision-free path from the start state to the end state. Sanz et al. [56] proposed an expert-guided kinodynamic RRT algorithm to improve the random sampling process of RRT. This method encompasses human knowledge to build an expert system, which calculates the deterministic optimal action sequence.

In the past few years, some researchers have also used supervised learning methods to improve the performance of traditional iterative learning control. Iterative learning control is a method of system tracking control, which works in a repetitive manner. Xu et al. [57] presented a high-order neural network-based adaptive iterative learning control approach to deal with repetitive tracking control problems. The high-order neural network is designed to iteratively estimate the desired control signals. Zhang et al. [58] proposed neural network-based iterative learning control to address non-repetitive issues. The non-linear part of the outputs of iterative learning control is estimated by a general neural network and a switching neural network. Li et al. [59] used the non-linear mapping and feature extraction ability of deep learning to determine whether the uncertain system satisfies the global Lipschitz condition so as to realize the iterative learning control of the system. Devi et al. [60] used a neural network to exploit the collected data for a learning control to enhance the performance of the controller. This method can reduce the computational effort and guarantee a faster convergence compared with the traditional iterative learning control method.

A table (Table 1) allows readers to conveniently understand the supervised learning-based motion-planning algorithms.

4 | UNSUPERVISED LEARNING BASED MOTION PLANNING

In contrast to enormous supervised learning-based motion-planning algorithms, there exist few unsupervised learning frameworks for motion-planning problems. Sarker et al. [61] proposed a novel motion prediction network called PROM-Net, which learns to make visual predictions for robot motions from raw video frames in a completely unsupervised manner. Compared with supervised learning-based motion planners, the PROM-Net is lightweight and can be easily implemented, especially for platforms with limited computing memory. Inspired by RL, an unsupervised learning path planning algorithm is introduced, called Plan2vec [62]. Plan2vec uses near-neighbour distances to construct a weighted graph and distills path-integral to extrapolate local metric for global embedding. Experimental results reveal that it can significantly amortize the planning cost and enhance reactive planning.

5 | REINFORCEMENT LEARNING BASED MOTION PLANNING

RL originated from optimal control and animal psychology inspired trial-and-error search [63]. There are mainly three types of RL approaches, *value-based*, *policy-based*, and *actor-critic (AC)*. *AC* derives from *policy-based* approaches, which uses a critic to estimate the action-value function. For convenience and clarity, this section is divided into two subsections, motion-planning with value-based RL method and motion-planning with policy-based RL method. Many strategies improving the performance of the motion planning framework by RL have been realized in recent years. According to the role of RL playing in motion-planning, these methods can be roughly divided into two categories, *End-to-end Solution* and *Module Solution*. As shown in Section 2.3, the motion-planning algorithm can be formulated as an MDP problem; thus, some methods map the motion-planning problems to MDPs and solve the MDPs directly. RL policy and the motion-planning algorithm interact with each other and work as a whole as a motion planner. When the RL policy works as a motion planner, it is classified as an *End-to-end Solution*. The other RL methods replace one or two of the components of the motion planner with RL policy, which is classified as a *Module Solution*.

5.1 | Motion planning with value-based RL method

The value-based RL method improves the policy π by acting greedily with respect to the rewards, as shown in Equation (2).

$$\pi(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q_{\pi}(s, a) \tag{2}$$

The improved policy generates new estimation of the rewards $Q_e(s, a)$ used to update the value functions $Q'(s, a)$. Q-learning and value iteration are two commonly used methods of value-based algorithms. In Q-learning, the value function is updated as Equation (3), where α is the learning rate.

$$Q'(s, a) = Q(s, a) + \alpha \times (Q_e(s, a) - Q(s, a)) \tag{3}$$

The value iteration algorithm updates the value function as Equation (4).

$$V'(s) = \max_{a \in \mathcal{A}} Q_e(s, a) \tag{4}$$

Through a large number of iterations, the value function and policy will both converge to the optimal as V^*, π^*.

In 1990, Sutton et al. [64] first used dynamic programing's policy iteration method to solve learning and planning problems. Sutton proposed Dyna-PI, a framework of value-based method on planning problems, and applied Q-learning algorithm on a navigation task as a demonstration. Moreover, four

TABLE 1 Supervised learning based motion planning algorithms

Class		Name	Method	year
End-to-end		Pfeiffer et al. [9]	CNN + FC	2017
		Kurutach et al. [13]	Causal InfoGAN	2018
		Qureshi et al. [14]	CAE + DNN	2019
		Mayur et al. [12]	RNN	2019
		Hamandi et al. [10]	ResNet + LSTM	2019
		Ichter et al. [17]	CNN + FC	2019
		Huh et al. [18]	CNN	2020
		Qureshi et al. [15]	CNN + FC	2020
Module replacement	Proprocessing	Ichter et al. [26]	CVAE	2018
		Higueras et al. [24]	FCN	2018
		Kumar et al. [27]	CVAE	2019
		Ariki et al. [25]	FCN	2019
		Naman et al. [20]	CNN + U-Net	2020
		Yonetani et al. [21]	FCN	2020
		Wang et al. [19]	CNN + ResNet	2020
		Ma et al. [29]	GAN + U-Net	2020
	Prediction	Qureshi et al. [37]	CAE + DNN	2018
		Xue et al. [40]	CNN + LSTM	2018
		Park et al. [41]	LSTM	2018
		Ma et al. [42]	LSTM	2019
		Elhafsi et al. [36]	CNP	2020
	Collision-checking	Hazirbas et al. [47]	CNN	2016
		Lu et al. [46]	CNN	2019
		Kew et al. [43]	DNN	2019
		Fan et al. [45]	CNN	2020
		Tran et al. [44]	CAE + MLP	2020
	Executing	Moeys et al. [51]	CNN	2016
		Garimella et al. [50]	RNN	2017
		Maqueda et al. [52]	DNN	2018
		Soonkyum et al. [48]	CNN	2020
		Guzzi et al. [49]	CNN	2020

Abbreviations: CAE, Contractive AutoEncoder; CNN, Convolutional Neural Network; CVAE, Conditional Variational AutoEncoder; DNN, Deep Neural Network; FC, Fully Connected; FCN, Fully Convolutional Network; GAN, Generative Adversarial Network; LSTM, Long Short-Term Memory; MLP, Multi-layer Perceptron; RNN, Recurrent Neural Network.

key issues for future works were proposed in [64]: large-scale generalisation, the balance between exploration and exploitation, the improvement of reward functions, and the model-free estimation of the world states.

Under the guidance of Dyna-PI [64], many value-based methods are proposed to tackle motion-planning problems. In recent years, deep neural networks are introduced in traditional value-based RL algorithms. In [65–70], value-based RL method serves as an *end-to-end solution* of the motion-planning problem. Aviv Tamar et al. [65] proposed a novel Value Iteration Network (VIN) that is able to directly map the image observation of the environment to the planning computation and generate action predictions. In this method, the classic value iteration planning algorithm is represented by CNN, which enables the policy to be trained end-to-end through back-propagating and makes learning MDP parameters and reward functions much easier. However, this method (1) conducts value iteration over the entire state space, making it difficult to be applied to large-scale and high-dimensional spaces and (2) does not perform well in sparse reward

environments. To address the aforementioned challenges and improve generalisation ability, different methods [66–70] have been raised under the framework of VIN. Sufeng Niu et al. [68], Daniel Schleich et al. [69], and Lisa Lee et al. [70] improved the performance of VIN by changing the network architecture with LSTM module, graph representation, multi-scale inputs etc. [66,67] extended the model to Semi-Markov Decision Process (SMDP) and Partially Observable Markov Decision Process (POMDP). Oh et al. [67] proposed a value-prediction network for SMDPs that learns option-value function and dynamics of the rewards to perform planning. Arbaaz Khan et al. [66] proposed a hierarchical process to learn effective planning in a partially observable environment under sparse rewards. To be specific, the algorithm uses VIN to computes optimal policies in the locally observed environment, and put the locally optimal solution and global sparse feature representation into an LSTM network to generate optimal policy over the whole environment.

Apart from the aforementioned end-to-end methods, value-based RL can also be used as *Module Solution* [71–76]. For example, value-based RL can be used as heuristics for motion-planning algorithms, such as A* and RRT [71–73]. Chen et al. [71] built a learnable neural-based expand operator based on tabular Q-learning and VIN to learn promising search directions for tree-based motion-planning algorithms. Yao et al. [76] introduced Deep Q Network (DQN) into the APF method to address the local-stable-point problem. Their method takes the potential field around the robot as the state of RL. In addition, the model is trained by curriculum learning. Bernhard et al. [72] integrated Double-DQN in the heuristic functions of Hybrid A* planner to leverage experiences for better performance. Huh et al. [73] proposed a learning softmax node selection method based on Q-function approximations to improve the random sampling strategy of sampling-based motion-planning algorithms. Bhardwaj et al. [74] addressed the shortest path problem by integrating RL to the lazy graph search algorithm. To be specific, the edge selection component is mapped to MDP and solved by the tabular Q-learning method. Eysenbach et al. [75] combined traditional planning algorithm and RL to deal with long-horizon, sparse reward tasks. The method decomposes the task of reaching a distant goal into subgoals and turns it into a series of easy tasks. To solve this problem, the value function generated by RL is used to build the graph of waypoints, and a graph search algorithm is applied to find the shorted path in the corresponding graph.

5.2 | Motion planning with policy-based RL method

Policy-based RL approaches are proposed to address the optimization problem under the context of POMDP. They optimise the stochastic policies in the form of probability function mapping state to action directly, which is different from selecting deterministic actions according to the value-action function learnt by a value-based RL method. This means that they are naturally applicable to exploration in high-dimensional or continuous action spaces because only a set of parameters of the policy needs to be learnt instead of the value function to express the entire action space. Thus, they have better convergence properties than those of the value-based method. As a research focus of policy-based approaches, the AC method normally comprises actor and critic networks. The actor learns the policy parameters to generate actions, in the direction given by the critic, that update the value function parameters to evaluate the reward of the actions. In policy optimization, the objective function is to maximise

$$J(\theta) = E_{\tau\sim\pi_\theta}[R(\tau)] \tag{5}$$

where π_θ is the policy parameterised by θ and τ is the trajectory sampled according to π_θ. $R(\tau)$ is the total reward of τ. The policy gradient is

$$\nabla_\theta J(\theta) = E_\tau\left[\sum_{t=0}^{T-1} G_t \cdot \nabla_\theta log\pi_\theta(a_t|s_t)\right] \tag{6}$$

where $G_t = \sum_{tl=t}^{T-1} r_{tl}$ is the return for a Monte-Carlo trajectory. G_t is the unbiased but noisy estimate of $Q^{\pi_\theta}(a_t, s_t)$. In AC, G_t is replaced by critic $Q_w(a_t, s_t)$, which is parameterised by w. Then the gradient function becomes

$$\nabla_\theta J(\theta) = E_\tau\left[\sum_{t=0}^{T-1} Q_w(a_t, s_t) \cdot \nabla_\theta log\pi_\theta(a_t|s_t)\right] \tag{7}$$

where $\pi_\theta(a_t\,|\,s_t)$ is the actor policy and $Q_w(a_t, s_t)$ is the critic value.

Detailed derivation can be found in [77]. Based on the AC framework, many effective methods have been developed to take advantages of policy gradient and value function simultaneously, such as Trust Region Policy Optimization [78], AC using Kronecker-Factored Trust Region [79], Proximal Policy Optimization (PPO) [80], Deep Deterministic Policy Gradient (DDPG) [81], Twin Delayed Deep Deterministic Policy Gradient [82], and Zeroth-Order Supervised Policy Improvement [83]. Some of them have been utilised in motion planning to improve planning performance.

Training time for the imitation task for motion planning can be reduced by applying the Generative Adversarial Imitation Learning (GAIL) method [84, 85] incorporated DDPG into the GAIL approach, thus generating expert demonstration trajectories. The RL method is utilised to learn search heuristics as a part of the planning algorithms on the basis of expert demonstrations or solved instances. The learning setting is provably capable of improving the efficiency of motion planner in highly dynamic environments [86]. However, because of inadequate training data distribution near obstacles, training neural motion planning with imitation learning in high-dimensional domains may suffer from low precision and success rate. Therefore, RL becomes a promising tool to carry

out active and sufficient exploration for finding a collision free trajectory. A DDPG method is proposed for motion planning [87], which is modified with reduced variance in the actor update. By utilising a known system transition function to estimate expected discounted future rewards, the actor network experiences low estimation errors in the policy gradient process. Meanwhile, learning from previous experiences, the agent will gradually become 'smart' to try recorded successful actions rather than randomly starting from scratch. Thus, the training time is shortened, and a success rate of near 1.0 is reached with the neural motion planner trained by DDPG motion-planning algorithm, inspiring researchers to combine deep RL-based methods with prevailing approaches for motion planning.

To address the challenges in using PRM to construct a long-range roadmap under task constraints as well as environmental and system model uncertainties for robots, a point-to-point navigation policy with DDPG was proposed as the local planner for the hierarchical approach PRM-RL in [88]. Connectivity in the map is determined by the RL agent, demonstrating more resilient results compared with those produced by conventional PRMs with straight-line interpolation.

Furthermore, RL-RRT is developed in the context of RRT for long-range motion planning under kinodynamic constraints [89]. An Auto-RL algorithm operates as a steering function for local state transition in the same way PRM-RL does. In addition, a reachability estimator to definite distance metric is trained via a supervised learning method taking into account dynamic constraints and obstacles. Although PRM-RL [88] and RL-RRT [89] both contain the high-level sampling-based planners with an RL-based local obstacle-avoiding policy, the latter RL agent can be deployed directly without the need of tuning reward and network structure. To achieve online and model-free kinodynamic planning, [90–92], were proposed. Benefiting from the vertex extension and metric role of the AC neural network, the framework can learn the optimal policy without model information to tackle every two-point boundary value problem from the RRTs.

With the purpose of accelerating task-and-motion planning and increasing its flexibility, an RL agent for searching extension heuristics can be trained with the acquired expert examples and resolved problems instances instead of manually derived logical descriptions [93, 94]. Solving a task-and-motion planning problem is more complicated than solving a traditional motion-planning problem because of the sparse reward training settings of long-range planning with multiple high-level multi-steps. [93] proposed a curious sample planner that applies a self-supervision signal rather than supervising on the training instances to find interesting configuration plans so that accelerating planning. A search tree over the state space using parameterised macroactions is constructed by maximising the curiosity signal, which increases the sampling rate of the feasible novel action adopted by the actor net.

By taking thelayout of the obstacles (i.e. the workspace configuration) as prior information, the planner may achieve better performance. Many neural motion-planning methods try to encode the workspace with deep neural networks. In [95], Strudel et al. proposed to learn a function that encodes the obstacles and the goal configuration as a vector. A PointNet-like network is used to encode point cloud in their method. The obstacle representation jointly with the motion-planning policies is learnt in Soft Actor Critic (SAC) [96] framework. In this work, the motion planner is constructed by the SAC framework. RL methods can hardly handle situations with sparse rewards, while sampling-based motion-planning struggles with tasks that involve rich interaction with the environment. There are also some works trying to combine RL and sampling-based motion-planning algorithm to alleviate their drawbacks. In [97], Yamada et al. added motion planner augmented action spaces to RL. Motion-planning algorithm and model-free RL are used adaptively according to target state difference. They also used SAC to train their model. In this work, motion planning algorithm helps to train the RL agent to efficiently find a better policy. Jurgenson et al. [98] proposed a subgoal tree trajectory structure, and then PPO is used to predict subgoals based on this structure. This method is also applied to neural motion-planning in et al. [14]. RL methods are also used to bias the sampling distribution of sampling-based motion planner. [99] used a policy gradient method to learn the implicit sampling distribution. The results shows that the learnt policy decreases the execution time, path length, number of tree nodes ,and number of collision checks.

In addition, many RL based motion-planning methods for specific tasks have emerged. Regarding diverse motion-planning tasks in navigation and manipulation, AC structure is generally built for an end-to-end motion-planning system utilising sensor information as input. DDPG [100–102] and PPO [103–108] are often adopted to optimise an action sequence of the robot for the task because of their outstanding policy approximation abilities in continuous action space. In [109], Saxena et al. utilised model-free RL to solve motion planning in the dense traffic of autonomous vehicle, and the model was trained by PPO. In Ref. [110], Wang et al. formulated autonomous driving as a hierarchical behaviour and motion-planning problem and trained the policy by PPO. Kamali et al. [111] proposed a dynamic-goal deep RL method to address the problem of robot arm motion planning in tele-manipulation applications. Their method leverages PPO to train the policy network with the robot arm joint value and the reference trajectory. Sivanathan et al. [112] presented an RL based decentralised motion-planning framework for the task of multiple-robot navigation. They also used PPO to train the policies that takes the observation vector as input.

To allow readers to conveniently understand the RL-based motion-planning algorithms, Table 2 has been included for reference.

6 | CONCLUSIONS

In this article, the learning-based robot motion-planning algorithms are discussed from four aspects: classical definition and learning-related definition, motion planning with supervised learning method, motion-planning with unsupervised

learning method, and motion planning with the reinforcement learning method. It serves as a survey for readers to understand the learning-based robot motion-planning algorithms conveniently.

In the future, the development of learning-based motion-planning algorithms with high generalisation ability is an important research direction. Specifically, most of the current learning-based motion-planning algorithms are restricted to

TABLE 2 Reinforcement learning based motion planning algorithms

Class		Work	RL-Method	Role (Pipeline)	Year
Value-based	End-to-end solution	Tamar et al. [65]	VIN	—	2016
		Oh et al. [67]	VIN		2017
		Khan et al. [66]	VIN		2017
	Module solution	Bernhard et al. [72]	Double-DQN	CF(A*)	2018
		Huh et al. [73]	DQN	HS(SBP)	2018
		Chen et al. [71]	Tabular Q-learning	HS(SBP)	2019
		Bhardwaj et al. [74]	DQN	SP(GBP)	2019
		Eysenbach et al. [75]	DQN	GBP(LTP)	2019
		Yao et al. [76]	DQN	LSP(APF)	2019
Policy-based	End-to-end solution	Jurgenson et al. [98]	PPO	—	2020
		Strudel et al. [95]	SAC		2020
		Yamada et al. [97]	SAC		2020
		Zuo et al. [85]	DDPG		2020
		Jurgenson et al. [87]	DDPG		2019
		Sun et al. [103]	PPO		2019
		Yang et al. [100]	DDPG		2020
		Butyrev et al. [101]	DDPG		2019
		Kim et al. [113]	TD3 + HER		2020
		Zhang et al. [114]	PG		2020
		Rong et al. [115]	PG		2020
		Zhao et al. [104]	PPO		2020
		Kim et al. [105]	PPO		2020
		Tsounis et al. [106]	PPO		2020
		Coad et al. [107]	PPO		2020
		Wu et al. [102]	DDPG		2020
		Al-Hilo et al. [108]	PPO		2020
	Module solution	Zhang et al. [99]	AC	HS(SBP)	2018
		Faust et al. [88]	DDPG	LP(PRM)	2018
		Curtis et al. [93]	PPO	HS(SBP)	2020
		Kim et al. [94]	AC	TAMP(SBP)	2019
		Gao et al. [116]	TD3	LP(PRM)	2020
		Chiang et al. [89]	DDPG	TPBVP(RRT)	2019
		Kontoudis et al. [91]	AC	TPBVP(RRT)	2019
		Kontoudis et al. [90]	AC	TPBVP(RRT)	2019
		Kontoudis et al. [92]	AC	TPBVP(RRT)	2020

Abbreviations: AC, Actor-Critic; CF, Cost Function; DDPG, Deep Deterministic Policy Gradient; DQN, Deep Q Network; GBP, Graph-based Planning; HER, Hindsight Experience Replay; HS, Heuristic Sampling; LP, Local Planner; LSP, Local-stable-point problem; LTP, Long-term Planning problem; PPO, Proximal Policy Optimization; PRM, Probabilistic Roadmap; RRT, Rapidly exploring Random Tree; SAC, Soft Actor Critic; SBP, Sampling-based Planning; SP, Shortest Path problem; TAMP, Task-and-motion Planning; TD3, Twin Delayed Deep Deterministic Policy Gradient; TPBVP, Two-point Boundary Value Problem; VIN, Value Iteration Network.

environments similar to those of the training data, thereby performing poorly when transferred to more complex conditions, or ones with larger differences. Additionally, many methods cannot deal with large-scale environment maps. Therefore, it is necessary to develop algorithms that can work in different scale environments.

In addition, as far as we know, the main difficulty that supervised learning-based methods face is the collection of full-scale data, while RL-based methods can be inefficient because of the necessity for robots to interact with environments. It is an interesting topic to combine the advantages of these two methods to reduce the negative aspects of each method.

ACKNOWLEDGMENTS

This research was funded by National Key R&D program of China with Grant No. 2019YFB1312400, Hong Kong RGC GRF grant No.14200618, Hong Kong RGC TRS grant No. T42-409/18-R and Hong Kong RGC CRF grant No. C4063-18GF.

ORCID

Jiankun Wang https://orcid.org/0000-0001-9139-0291
Tianyi Zhang https://orcid.org/0000-0003-2223-5460
Nachuan Ma https://orcid.org/0000-0002-7107-6577
Zhaoting Li https://orcid.org/0000-0002-9873-0858
Han Ma https://orcid.org/0000-0003-1960-5432
Fei Meng https://orcid.org/0000-0001-9225-040X
Max Q.-H. Meng https://orcid.org/0000-0002-5255-5898

REFERENCES

1. Siciliano, B., Khatib, O.: Springer Handbook of Robotics. Springer, Berlin, Heidelberg (2016)
2. Meyes, R., et al.: Motion planning for industrial robots using reinforcement learning. Procedia CIRP. 63, 107–112 (2017)
3. Torres, L.G., Alterovitz, R.: Motion planning for concentric tube robots using mechanics-based models. In: Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5153–5159. IEEE, San Francisco (2011)
4. Wang, J., Meng, M.Q.H., Khatib, O.: EB-RRT: Optimal motion planning for mobile robots. IEEE Trans. Autom. Sci. Eng. 17 (4), 2063-2073 (2020)
5. Wang, J., Meng, M.Q.-H.: Socially compliant path planning for robotic autonomous luggage trolley collection at airports. Sensors. 19, 2759 (2019)
6. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4, 100–107 (1968)
7. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. 5, 90–98 (1986)
8. LaValle, S.M., Kuffner, J.J., Jr.: Randomized kinodynamic planning. Int. J. Robot. Res. 20, 378–400 (2001)
9. Pfeiffer, M., et al.: From perception to decision: a data-driven approach to end-to-end motion planning for autonomous ground robots. In: Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 1527–1533. IEEE, Singapore (2017)
10. Hamandi, M., D'Arcy, M., Fazli, P.: Deepmotion: learning to navigate like humans. In: Proceedings of the 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN). pp. 1–7. IEEE, New Delhi (2019)
11. He, K., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778. Las Vegas (2016)
12. Bency, M.J., Qureshi, A.H., Yip, M.C.: Neural Path Planning: Fixed Time, Near-optimal Path Generation Via Oracle Imitation. arXiv preprint arXiv:1904.11102 (2019)
13. Kurutach, T., et al.: Learning plannable representations with causal infogan. Adv. Neural. Inf. Process Syst., pp. 8733–8744. Montreal (2018)
14. Qureshi, A.H., et al.: Motion planning networks. In: Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), pp. 2118–2124. IEEE, Montreal (2019)
15. Qureshi, A.H., et al.: Motion planning networks: bridging the gap between learning-based and classical motion planners. IEEE Trans. Robot. 37(1), 48–66 (2020)
16. Goodfellow, I., et al.: Generative adversarial nets. Adv. Neural Inf. Process. Syst. 27, 2672–2680, Montreal (2014)
17. Ichter, B., Pavone, M.: Robot motion planning in learnt latent spaces. IEEE Robot. Autom. Lett. 4, 2407–2414 (2019)
18. Huh, J., Isler, V., Lee, D.D.: Cost-to-Go Function Generating Networks for High Dimensional Motion Planning. arXiv preprint arXiv:2012.06023 (2020)
19. Wang, J., et al.: Neural RRT*: learning-based optimal path planning. IEEE Trans. Autom. Sci. Eng. 17 (4), 1748-1758 (2020)
20. Shah, N., Srinet, A., Srivastava, S.: Learning Sampling Distributions for Efficient High-Dimensional Motion Planning. arXiv preprint arXiv:2012.00658 (2020)
21. Yonetani, R., et al.: Path Planning using Neural A* Search. arXiv preprint arXiv:2009.07476 (2020)
22. Molina, D., Kumar, K., Srivastava, S.: Learn and link: learning critical regions for efficient planning. In: Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 10605–10611. IEEE, Paris (2020)
23. Cheng, R., Shankar, K., Burdick, J.W.: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). In: Learning an Optimal Sampling Distribution for Efficient Motion Planning. IEEE, Las Vegas (2020)
24. Pérez-Higueras, N., Caballero, F., Merino, L.: Learning human-aware path planning with fully convolutional networks. In: Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–5. IEEE, Brisbane (2018)
25. Ariki, Y., Narihira, T.: Fully Convolutional Search Heuristic Learning for Rapid Path Planners. arXiv preprint arXiv:1908.03343 (2019)
26. Ichter, B., Harrison, J., Pavone, M.: Learning sampling distributions for robot motion planning. In: Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 7087–7094. IEEE, Brisbane (2018)
27. Kumar, R., et al.: Lego: Leveraging Experience in Roadmap Generation for Sampling-based Planning. arXiv preprint arXiv:1907.09574 (2019)
28. Takahashi, T., et al.: Learning heuristic functions for mobile robot path planning using deep neural networks. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 29, pp. 764–772. Berkeley (2019)
29. Ma, N., Wang, J., Meng, M.Q.H.: Conditional Generative Adversarial Networks for Optimal Path Planning. arXiv preprint arXiv:2012.03166 (2020)
30. Ichter, B., et al.: Learnt critical probabilistic roadmaps for robotic motion planning. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 9535–9541. IEEE, Paris (2020)
31. Chen, L.C., et al.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 801–818. Munich (2018)
32. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Proceedings of the International Conference on Medical Image Computing and Computer-assisted Intervention, pp. 234–241. Springer, Munich (2015)
33. Pérez-Higueras, N., Caballero, F., Merino, L.: Learning robot navigation behaviours by demonstration using a RRT* planner. In: Proceedings of

the International Conference on Social Robotics, pp. 1–10. Springer, Kansas (2016)
34. Shiarlis, K., Messias, J., Whiteson, S.: Rapidly exploring learning trees. In: Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 1541–1548. IEEE, Singapore (2017)
35. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. Adv. Neural Inf. Process. Syst. 28, 3483–3491, Montreal (2015)
36. Elhafsi, A., et al.: Map-predictive motion planning in unknown environments. In: Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 8552–8558. IEEE, Paris (2020)
37. Qureshi, A.H., Yip, M.C.: Deeply informed neural sampling for robot motion planning. In: Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6582–6588. IEEE, Madrid (2018)
38. Gammell, J.D., Srinivasa, S.S., Barfoot, T.D.: Informed RRT*: optimal sampling-based path planning focussed via direct sampling of an admissible ellipsoidal heuristic. In: Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2997–3004. IEEE, Chicago (2014)
39. Gammell, J.D., Srinivasa, S.S., Barfoot, T.D.: Batch informed trees (BIT*): sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In: Proceedings of the 2015 IEEE international Conference on Robotics and Automation (ICRA), pp. 3067–3074. IEEE, Seattle (2015)
40. Xue, H., Huynh, D.Q., Reynolds, M.: SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction. In: Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1186–1194. IEEE, Lake Tahoe (2018)
41. Park, S.H., et al.: Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. In: Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1672–1678. IEEE, (2018)
42. Ma, Y., et al.: Trafficpredict: trajectory prediction for heterogeneous traffic-agents. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 6120–6127. AAAI, Honolulu (2019)
43. Kew, J.C., et al.: Neural Collision Clearance Estimator for Fast Robot Motion Planning. arXiv preprint arXiv:1910.05917 (2019)
44. Tran, T., Denny, J., Ekenna, C.: Predicting Sample Collision with Neural Networks. arXiv preprint arXiv:2006.16868 (2020)
45. Fan, R., et al.: Sne-roadseg: incorporating surface normal information into semantic segmentation for accurate freespace detection. In: Proceedings of the European Conference on Computer Vision, pp. 340–356. Springer, Glasgow (2020)
46. Lu, C., van de Molengraft, M.J.G., Dubbelman, G.: Monocular semantic occupancy grid mapping with convolutional variational encoder-decoder networks. IEEE Robot. Autom. Lett. 4, 445–452 (2019)
47. Hazirbas, C., et al.: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In: Proceedings of the Asian Conference on Computer Vision, pp. 213–228. Springer, Taipei (2016)
48. Kim, S., An, B.: Learning heuristic A*: efficient graph search using neural network. In: Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 9542–9547. IEEE, Paris (2020)
49. Guzzi, J., et al.: Path planning with local motion estimations. IEEE Robot. Autom. Lett. 5, 2586–2593 (2020)
50. Garimella, G., et al.: Neural network modelling for steering control of an autonomous vehicle. In: Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2609–2615. IEEE, Vancouver (2017)
51. Moeys, D.P., et al.: Steering a predator robot using a mixed frame/event-driven convolutional neural network. In: Proceedings of the 2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP), pp. 1–8. IEEE, Krakow (2016)
52. Maqueda, A.I., et al.: Event-based vision meets deep learning on steering prediction for self-driving cars. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5419–5427. Salt Lake City (2018)
53. Chen, L., et al.: Fuzzy kinodynamic RRT: a dynamic path planning and obstacle avoidance method. In: Proceedings of the 2020 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 188–195. IEEE, Athens (2020)
54. Jaradat, M.A.K., Garibeh, M.H., Feilat, E.A.: Autonomous mobile robot dynamic motion planning using hybrid fuzzy potential field. Soft Comput. 16, 153–164 (2012)
55. Zhu, Q., et al.: Motion planning of autonomous mobile robot using recurrent fuzzy neural network trained by extended Kalman filter. Comput. Intell. Neurosci. 2019 (2019)
56. Sanz, J.M., et al.: Expert-guided kinodynamic RRT path planner for non-holonomic robots. In: Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6540–6545. IEEE, 2018)
57. Xu, Q.Y., Li, X.D.: HONN-based adaptive ILC for pure-feedback nonaffine discrete-time systems with unknown control directions. IEEE Trans. Neural Netw. Learn. Syst. 31, 212–224 (2019)
58. Zhang, D., Wang, Z., Masayoshi, T.: Neural-network-based iterative learning control for multiple tasks. IEEE Trans. Neural Netw. Learn. Syst. (2020). https://doi.org/10.1109/TNNLS.2020.3017158
59. Li, J., et al.: DNN-based implementation of data-driven iterative learning control for unknown system dynamics. In: Proceedings of the 2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS), pp. 1037–1042. IEEE, Guangxi (2020)
60. Lakshmidevinivas, D., Deniz, M., Balakrishnan, S.: Neural network augmented intelligent iterative learning control for a nonlinear system. In: Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE, Glasgow (2020)
61. Sarkar, M., Pradhan, P., Ghose, D.: Planning Robot Motion Using Deep Visual Prediction. arXiv preprint arXiv:1906.10182 (2019)
62. Yang, G., et al.: Plan2Vec: Unsupervised Representation Learning by Latent Plans. arXiv preprint arXiv:2005.03648 (2020)
63. Nian, R., Liu, J., Huang, B.: A review on reinforcement learning: introduction and applications in industrial process control. Comput. Chem. Eng. 139, 106886 (2020)
64. Sutton, R.S.: Integrated architectures for learning, planning, and reacting based on approximating dynamic programing. In: Machine Learning Proceedings 1990, pp. 216–224. Elsevier (1990)
65. Tamar, A., et al.: Value iteration networks. Adv. Neural Inf. Process. Syst. 29, 2154–2162, Barcelona (2016)
66. Khan, A., et al.: Memory Augmented Control Networks. arXiv preprint arXiv:1709.05706 (2017)
67. Oh, J., Singh, S., Lee, H.: Value prediction network. Adv. Neural Inf. Process. Syst., 6118–6128, Long Beach (2017)
68. Niu, S., et al.: Generalized value iteration networks: life beyond lattices. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32. New Orleans (2018)
69. Schleich, D., Klamt, T., Behnke, S.: Value Iteration Networks on Multiple Levels of Abstraction. arXiv preprint arXiv:1905.11068 (2019)
70. Lee, L., et al.: Gated Path Planning Networks. arXiv preprint arXiv:1806.06408 (2018)
71. Chen, B., et al.: Learning to Plan in High Dimensions via Neural Exploration-Exploitation Trees. arXiv preprint arXiv:1903.00070 (2019)
72. Bernhard, J., et al.: Experience-based heuristic search: robust motion planning with deep Q-learning. In: Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 3175–3182. IEEE, Maui (2018)
73. Huh, J., Lee, D.D.: Efficient sampling with Q-learning to guide rapidly exploring random trees. IEEE Robot. Autom. Lett. 3, 3868–3875 (2018)
74. Bhardwaj, M., et al.: Leveraging Experience in Lazy Search. arXiv preprint arXiv:1907.07238 (2019)

75. Eysenbach, B., Salakhutdinov, R.R., Levine, S.: Search on the replay buffer: bridging planning and reinforcement learning. Adv. Neural Inf. Process. Syst. pp. 15246–15257, Vancouver (2019)
76. Yao, Q., et al.: Path planning method with improved artificial potential field-A reinforcement learning perspective. IEEE Access. 8, 135513–135523 (2020). https://doi.org/10.1109/ACCESS.2020.3011211
77. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT press, Cambridge (2018)
78. Schulman, J., et al.: Trust region policy optimization. In: Proceedings of the International Conference on Machine Learning, pp. 1889–1897. Lille (2015)
79. Wu, Y., et al.: Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. Adv. Neural Inf. Process. Syst. pp. 5279–5288, Delft (2017)
80. Schulman, J., et al.: Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347 (2017)
81. Lillicrap, T.P., et al.: Continuous Control with Deep Reinforcement Learning. arXiv preprint arXiv:1509.02971 (2015)
82. Fujimoto, S., Van Hoof, H., Meger, D.: Addressing Function Approximation Error in Actor-critic Methods. arXiv preprint arXiv:1802.09477 (2018)
83. Sun, H., et al.: Zeroth-Order Supervised Policy Improvement. arXiv preprint arXiv:2006.06600 (2020)
84. Ho, J., Ermon, S.: Generative Adversarial Imitation Learning. arXiv preprint arXiv:1606.03476 (2016)
85. Zuo, G., et al.: Deterministic generative adversarial imitation learning. Neurocomputing 388, 60-69 (2020)
86. Groshev, E., et al.: Learning generalized reactive policies using deep neural networks. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 28. Delft (2018)
87. Jurgenson, T., Tamar, A.: Harnessing Reinforcement Learning for Neural Motion Planning. arXiv preprint arXiv:1906.00214 (2019)
88. Faust, A., et al.: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In: Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 5113–5120. IEEE, Brisbane (2018)
89. Chiang, H.-T. L., et al.: Rl-rrt: kinodynamic motion planning via learning reachability estimators from rl policies. IEEE Robot. Autom. Lett. 4, 4298–4305 (2019)
90. Kontoudis, G.P., Vamvoudakis, K.G.: Kinodynamic motion planning with continuous-time Q-learning: an online, model-free, and safe navigation framework. IEEE Trans. Neural Netw. Learn. Syst. 30, 3803–3817 (2019)
91. Kontoudis, G.P., Vamvoudakis, K.G.: Robust kinodynamic motion planning using model-free game-theoretic learning. In: Proceedings of the 2019 American Control Conference (ACC), pp. 273–278. IEEE, Philadelphia (2019)
92. Kontoudis, G.P., Xu, Z., Vamvoudakis, K.G.: Online, model-free motion planning in dynamic environments: an intermittent, finite horizon approach with continuous-time Q-learning. In: Proceedings of the 2020 American Control Conference (ACC), pp. 3873–3878. IEEE, Denver (2020)
93. Curtis, A., et al.: Flexible and Efficient Long-Range Planning Through Curious Exploration. arXiv preprint arXiv:2004.10876 (2020)
94. Kim, B., Kaelbling, L.P., Lozano-Pérez, T.: Adversarial actor-critic method for task and motion planning problems using planning experience. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 8017–8024. AAAI, Honolulu (2019)
95. Strudel, R., et al.: Learning Obstacle Representations for Neural Motion Planning. arXiv preprint arXiv:2008.11174 (2020)
96. Haarnoja, T., et al.: Soft Actor-critic: Off-policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. arXiv preprint arXiv:1801.01290 (2018)
97. Yamada, J., et al.: Motion planner augmented action spaces for reinforcement learning. In: Proceedings of the RSS Workshop on Action Representations for Learning in Continuous Control, Corvalis (2020)
98. Jurgenson, T., et al.: Sub-Goal Trees–A Framework for Goal-based Reinforcement Learning. arXiv preprint arXiv:2002.12361 (2020)
99. Zhang, C., Huh, J., Lee, D.D.: Learning implicit sampling distributions for motion planning. In: Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3654–3661. IEEE, Madrid (2018)
100. Yang, J., Peng, G.: Deep Reinforcement Learning with Stage Incentive Mechanism for Robotic Trajectory Planning. arXiv preprint arXiv:2009.12068 (2020)
101. Butyrev, L., Edelhäußer, T., Mutschler, C.: Deep Reinforcement Learning for Motion Planning of Mobile Robots. arXiv preprint arXiv:1912.09260 (2019)
102. Wu, Y.H., et al.: Reinforcement learning in dual-arm trajectory planning for a free-floating space robot. Aero. Sci. Technol. 98, 105657 (2020)
103. Sun, Y., et al.: Mapless motion planning system for an autonomous underwater vehicle using policy gradient-based deep reinforcement learning. J. Intell. Rob. Syst. 96, 591–601 (2019)
104. Zhao, X., et al.: An actor-critic approach for legible robot motion planner. In: Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 5949–5955. IEEE, Paris (2020)
105. Kim, S., et al.: Motion planning by reinforcement learning for an unmanned aerial vehicle in virtual open space with static obstacles. In: Proceedings of the 2020 20th International Conference on Control, Automation and Systems (ICCAS), pp. 784–787. IEEE, Busan (2020)
106. Tsounis, V., et al.: Deepgait: planning and control of quadrupedal gaits using deep reinforcement learning. IEEE Robot. Autom. Lett. 5, 3699–3706 (2020)
107. Coad, J., Qiao, Z., Dolan, J.M.: Safe Trajectory Planning Using Reinforcement Learning for Self Driving. arXiv preprint arXiv:2011.04702 (2020)
108. Al-Hilo, A., et al.: UAV-assisted content delivery in intelligent transportation systems-joint trajectory planning and cache management. IEEE Trans. Intell. Transport Syst. (2020)
109. Saxena, D.M., et al.: Driving in dense traffic with model-free reinforcement learning. In: Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 5385–5392. IEEE, Paris (2020)
110. Wang, J., et al.: Learning Hierarchical Behavior and Motion Planning for Autonomous Driving. arXiv preprint arXiv:2005.03863 (2020)
111. Kamali, K., Bonev, I.A., Desrosiers, C.: Real-time motion planning for robotic teleoperation using dynamic-goal deep reinforcement learning. In: Proceedings of the 2020 17th Conference on Computer and Robot Vision (CRV), pp. 182–189. IEEE, Ottawa (2020)
112. Kandhasamy, S., et al.: Decentralized Motion Planning for Multi-Robot Navigation using Deep Reinforcement Learning. arXiv preprint arXiv:2011.05605 (2020)
113. Kim, M., et al.: Motion planning of robot manipulators for a smoother path using a Twin delayed deep deterministic policy gradient with hindsight experience replay. Appl. Sci. 10, 575 (2020)
114. Zhang, J., et al.: Reinforcement learning-based motion planning for automatic parking system. IEEE Access. 8, 154485–154501 (2020)
115. Rong, J., Luan, N.: Safe reinforcement learning with policy-guided planning for autonomous driving. In: Proceedings of the 2020 IEEE International Conference on Mechatronics and Automation (ICMA), pp. 320–326. IEEE, Beijing (2020)
116. Gao, J., et al.: Deep reinforcement learning for indoor mobile robot path planning. Sensors. 20, 5493 (2020)

How to cite this article: Wang, J., et al.: A survey of learning-based robot motion planning. IET Cyber-Syst. Robot. 3(4), 302–314 (2021). https://doi.org/10.1049/csy2.12020