Neurocomputing 170 (2015) 257–266

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A hierarchical path planning approach based on A* and least-squares policy iteration for mobile robots

Lei Zuo, Qi Guo, Xin Xu*, Hao Fu

College of Mechatronics and Automation, National University of Defense Technology, Changsha 410073, China

ARTICLE INFO

Article history:
Received 21 January 2014
Received in revised form
7 July 2014
Accepted 19 September 2014
Available online 16 June 2015

Keywords:
Mobile robots
Hierarchical path planning
A* search
Reinforcement learning
Least squares policy iteration (LSPI)
Optimality

ABSTRACT

In this paper, we propose a novel hierarchical path planning approach for mobile robot navigation in complex environments. The proposed approach has a two-level structure. In the first level, the A* algorithm based on grids is used to find a geometric path quickly and several path points are selected as subgoals for the next level. In the second level, an approximate policy iteration algorithm called least-squares policy iteration (LSPI) is used to learn a near-optimal local planning policy that can generate smooth trajectories under kinematic constraints of the robot. Using this near-optimal local planning policy, the mobile robot can find an optimized path by sequentially approaching the subgoals obtained in the first level. One advantage of the proposed approach is that the kinematic characteristics of the mobile robot can be incorporated into the LSPI-based path optimization procedure. The second advantage is that the LSPI-based local path optimizer uses an approximate policy iteration algorithm which has been proven to be data-efficient and stable. The training of the local path optimizer can use sample experiences collected randomly from any reasonable sampling distribution. Furthermore, the LSPI-based local path optimizer has the ability of dealing with uncertainties in the environment. For unknown obstacles, it just needs to replan the path in the second level rather than the whole planner. Simulations for path planning in various types of environments have been carried out and the results demonstrate the effectiveness of the proposed approach.

1. Introduction

Intelligent mobile robots have drawn lots of interests in the past decades. Path planning is one of the fundamental issues in robotics research to improve the autonomy of mobile robots in complex environments. The problem of path planning, whose aim is to find a collision-free path from an initial position to a goal position, has been extensively studied in the literature [1–9]. Many traditional approaches for mobile robot path planning are based on geometry. Such approaches include roadmaps, cell decomposition and artificial potential fields [6]. Geometric path planning methods have been widely applied in intelligent mobile robots. However, geometric path planning methods usually do not take the robot's kinematic and dynamic constraints into consideration, which cause the obtained paths are usually not smooth and feasible for robots to execute.

To incorporate the robot's kinematic or dynamic constraints and improve the efficiency of path planning, sampling-based algorithms have been developed in recent years. The probabilistic roadmap method (PRM) [6,10,11] constructs a roadmap by using uniformly distributed random sampling of the free space. As a popular class of sampling-based planning algorithms, Rapidly Exploring Random Trees (RRT) was first proposed by LaValle [12] and has been applied in various path planning tasks for mobile robots [13–15]. Some other sampling-based methods have also been proposed, such as Expansive Space Trees (ESTs) [16], Sampling-based Roadmap of Trees (SRT) [17] and so on. Although sampling-based planning methods can take kinematic or dynamic constraints into account, they may find a path without optimality. What is more, these methods are based on random sampling, which causes the planning results may be very different from each run.

To improve the path quality, planning methods with hierarchical structures have attracted much attention in the literature [18,19]. In such hierarchical methods, the higher level mainly focuses on finding a geometric, collision-free path from the initial location to the goal location. To find such a path, graph search methods can be used, such as the A* algorithm [20]. After a series of subgoals are provided by the higher level, the lower level generates a feasible path for the mobile robot to reach them sequentially. One of the main advantages of such hierarchical methods is that they can be computationally efficient and simple to implement. However, a challenge for the hierarchical structure is to deal with the uncertainties of the environment. To overcome this problem, it is necessary to improve the learning ability of the mobile robot. Some approaches based on computational intelligence (CI) have been studied, like neural networks [21,22], fuzzy

* Corresponding author.
E-mail address: xuxin_mail@263.net (X. Xu).

http://dx.doi.org/10.1016/j.neucom.2014.09.092
0925-2312/

logic [18] and so on. Many CI-based methods use supervised learning and need prior knowledge or supervised instances to train the mobile robot. However, supervised signals are usually difficult to obtain, especially in uncertain environments. What is more, these methods may not have the ability of performance optimization. For this reason, reinforcement learning (RL) has drawn more and more attention in recent years. Different from supervised learning, no prior supervised instances are presumed in reinforcement learning. Instead, an agent in reinforcement learning receives a feedback (reward) from the environment after taking an action [23]. Reinforcement learning enables an agent to autonomously discover an optimal behavior through trial-and-error by interacting with an initially unknown environment. Thus, it is very promising to apply RL methods in path planning to deal with the uncertainties of the environment.

So far, a number of reinforcement learning algorithms have been applied in robotics. Q-learning is one of the most popular RL algorithms used in mobile robot navigation. Nevertheless, Q-learning is a tabular-based algorithm and only can deal with discrete state spaces. When the state space becomes very large or continuous, it is very difficult and impractical to use Q-learning in mobile robot navigation. In order to deal with large or continuous state spaces, value function approximation methods have been developed in reinforcement learning. However, few of reinforcement learning methods with linear approximation architectures have been used in mobile robot path planning.

In this paper, we propose a hierarchical path planning approach based on reinforcement learning, which aims to find a collision-free path for mobile robots considering both feasibility and optimality of the path. The proposed approach includes two levels. In the first level, A* search is used to quickly search a short, obstacle-free geometric path that ignores the kinematic and dynamic constraints of the mobile robot. However, A* search is based on grids and the path may be not smooth and feasible for mobile robots to execute. Therefore, a local path optimizer based on least squares policy iteration (LSPI) is applied to smooth and optimize the path in the second level. The LSPI-based path optimizer can obtain an optimal or suboptimal planning policy for the mobile robot by interacting with the environment. Moreover, the LSPI-based path optimizer has the ability of dealing with the uncertainties of the environment. Due to the generalization ability of LSPI, the planning policy in the second level is adaptive to new environments without re-learning. In comparison with pure grids-based A* search, our proposed approach can find more smooth collision-free path. Compared with sampling-based methods, the path obtained by the proposed approach is better for some optimality objectives, such as shorter length.

One contribution of our work is the adaptive optimization of the planned path for mobile robots by using the LSPI algorithm, which can take the kinematic characteristics of the mobile robot into account. A* is used to quickly select subgoals for further path planning. In the second level, the LSPI-based path optimizer can find an optimal or near-optimal path by sequentially approaching those subgoals obtained in the first level. The other contribution of this paper is the application of value function approximation with linear architecture in path planning of mobile robots. Tabular-based reinforcement learning methods, like Q-learning, have been applied in path planning with discrete state spaces [24]. However, little work has been done to use linear function approximation methods to deal with the problem of path planning in large or continuous state spaces. In this paper, we apply the LSPI algorithm to optimize the planned path in continuous state spaces. The LSPI-based local path optimizer can efficiently use data sampled at random and has good generalization ability of dealing with the uncertainties in the environment.

The rest of this paper is organized as follows: some related works are introduced in Section 2. Then, the hierarchical path planning approach based on A* and LSPI is presented in Section 3. In Section 4, extensive simulations are carried out and performance evaluation of the proposed approach is discussed. Section 5 concludes with a summary of our contributions and a discussion of future work.

2. Related work

Path planning is very important for autonomous mobile robots. Most of the early work on path planning was based on geometry. There are mainly three classes of geometric planning approaches [6]: artificial potential fields, roadmaps and cell decompositions. The artificial potential field (APF) method was first proposed by Khatib [3], whose basic idea is to construct a scalar field in which the robot is attracted to the goal and is repulsed away from the obstacles. The APF method is a traditional approach for mobile robot path planning based on geometry and has been widely applied in mobile robots. However, APF methods usually do not take the robot's kinematic and dynamic constraints into consideration, which cause the obtained paths are usually not feasible for robots to execute. Later, many extensions and applications of the potential function method to path planning have been presented [4,5,25–27]. One of the well-known roadmaps is the visibility graph [28,29], where two points are connected with a straight line if they can see each other. Voronoi graph [30–32] is another important class of roadmaps methods. Geometric planning methods based on cell decompositions [1,2] divide the environment into convex, obstacle-free regions called cells. The idea of roadmaps and cell decompositions is to convert the path planning problem into a graph search problem by discretizing the environment. Based on the graph representation of the map, search algorithms are used to find a path from the initial point to the goal point without collision. A* [20] is one of the most popular search algorithms for path planning over costmaps [33,34]. However, as the number of obstacles increases and the shapes of obstacles become complex, it will be very difficult for roadmaps and cell decompositions to deal with, sometimes even impossible. The basic idea of the Bug algorithms is to follow the contour of each obstacle in the robot's way and thus circumnavigate it. Although such simple obstacle avoidance algorithms are often used in simple mobile robots, they have some difficult problems to be solved. For example, the Bug algorithms don not take into account robot kinematics, which can be especially important with nonholonomic robots.

To improve the quality of the planned path in complex environments, hierarchical planning methods have been developed in the literature [18,19,33,35,36]. Fujimura and Samet [35] proposed a hierarchical strategy for path planning among moving obstacles, in which time was included as one of the dimensions of the model world. Nevertheless, such approach suffers from large size of the search space. A hierarchical approximate cell decomposition method was presented for path planning in [36] that different resolutions were used in cell decomposition. To deal with unknown and dynamical environments, path planning methods based on computational intelligence have attracted more and more research interests [18,37]. In [37], a multi-objective path planning algorithm based on particle swarm optimization is proposed for robot navigation in uncertain environments. Yang et al. [18] proposed a hierarchical planning strategy with two layers, where fuzzy logic was used for the proposed motion planning strategy. However, the optimization and generalization ability of the planner still need to be improved.

Reinforcement learning [23] is a machine learning framework for sequential decision making problems. Prior RL algorithms, like Q-learning and Sarsa [23], work well with discrete state spaces.

Konar et al. [24] have applied Q-learning to path planning of a mobile robot. However, if the state spaces become very large or continuous, these algorithms will be computation expensive and impractical for applications. Approximation methods were considered in RL research to deal with large or continuous state spaces [38–40,44]. Inspired by the least-squares temporal difference learning algorithm [38], Lagoudakis and Ronald [39] proposed the least-squares policy iteration algorithm (LSPI), in which linear architectures were used to approximate the value functions in continuous state spaces. Linear approximation architectures are easy to implement and use. Nevertheless, few studies on the applications of linear value function approximation methods in path planning of mobile robots have been done so far. The LSPI algorithm can use data collected arbitrarily from any reasonable sampling distribution and enjoys the stability and soundness of approximate policy iteration. Therefore, we apply the LSPI algorithm to optimize the planned path in the problem of path planning with continuous state spaces.

3. The hierarchical path planning approach based on A* and LSPI

In this section, we first introduce the path planning problem for mobile robots and the mobile robot used in this paper briefly. Then, we present the hierarchical path planning approach based on A* and LSPI in detail.

3.1. Problem statement

The problem of path planning can be stated as follows. Inputting a map of the environment, a start point $s_0=(x_0, y_0)$ and a goal point $s_g=(x_g, y_g)$, path planning is to find a path with a sequence of points that is safe and feasible for mobile robots moving from the start point to the goal point.

The state of the mobile robot in this paper is defined as $s_t=[x_t, y_t, \theta_t]$ under the global coordinate, where x_t and y_t are the robot's position and θ_t is the angle between the forward direction of the robot and horizontal axis. The mobile robot has an omni-directional wheel and two driving wheels. The angular speeds of the driving wheels can be controlled. Six ultrasonic sensors are equipped in the front part of the robot. The detection distance of each sensor is d, and the detection angles are 30°, as shown in Fig. 1. The kinematic constraints of the mobile robot are

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \frac{R}{2l_b}\begin{pmatrix} -l_a & l_a \\ -l_b & -l_b \\ -1 & 1 \end{pmatrix}\begin{pmatrix} \omega_{W_1} \\ \omega_{W_2} \end{pmatrix} \tag{1}$$

where ω_{W1},ω_{W2} are respectively right wheel rotational velocity and left wheel rotational velocity [rad s^{-1}], R is actuated wheel radius [m], l_a, l_b are distances of wheels from robot's axes [m].

3.2. The framework of the proposed approach

There are two different levels in the framework of the proposed approach. A* search is used in the first level to find a near optimal geometric path and select subgoals for the local planning in the second level. In the second level, a LSPI-based local path optimizer, which takes the subgoals given by the first level as input, is applied to smooth and optimize the path. Fig. 2 shows the schematic representation of the hierarchical path planning method.

As shown in Fig. 2, a number of subgoals are generated in the first level by using the A* algorithm, which uses a heuristic to focus the search from an initial state to the goal state. A* algorithm visits the nodes in order of this heuristic estimate. In each step of a search, a cost estimation f is used to order the queue. The potential of approaching the goal from state s is evaluated by the estimated total cost given by

$$f(s)=g(s)+h(s) \tag{2}$$

where the two cost functions on the right are defined as follows:

1) $g(s)$—the actual cost of going from the initial state to the current state s;
2) $h(s)$—a heuristic estimate of the cost of going from the current state s to the goal state s_g.

Let $l(s_1, s_2)$ denotes the length of the straight line from state s_1 to s_2. In this paper, $g(s)$ is taken as the length of the path linking the geometric centers of the traversed grids, and $h(s)=l(s, s_g)$ as the straight line distance from the current state s to the goal state s_g. Based on the total cost estimation $f(s)$, the A* algorithm can search for a short path from the initial location to the desired location efficiently.

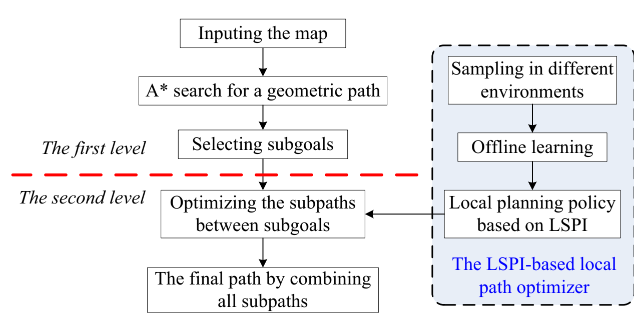


Fig. 2. The flowchart of the hierarchical path planning method.

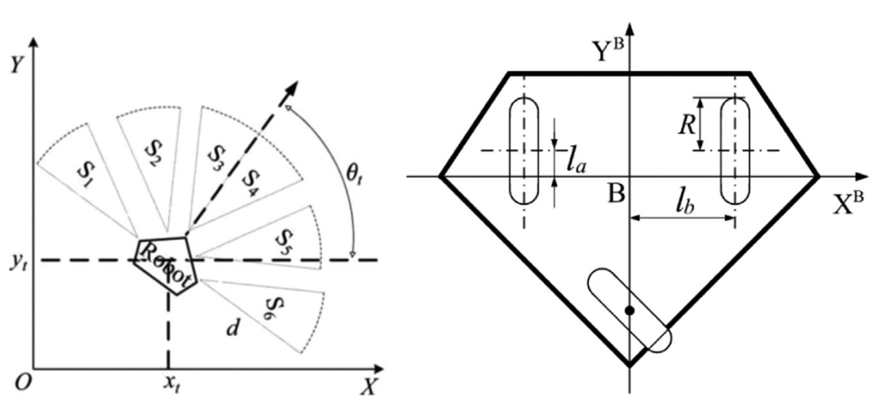


Fig. 1. The coordinate and sensors of the mobile robot.

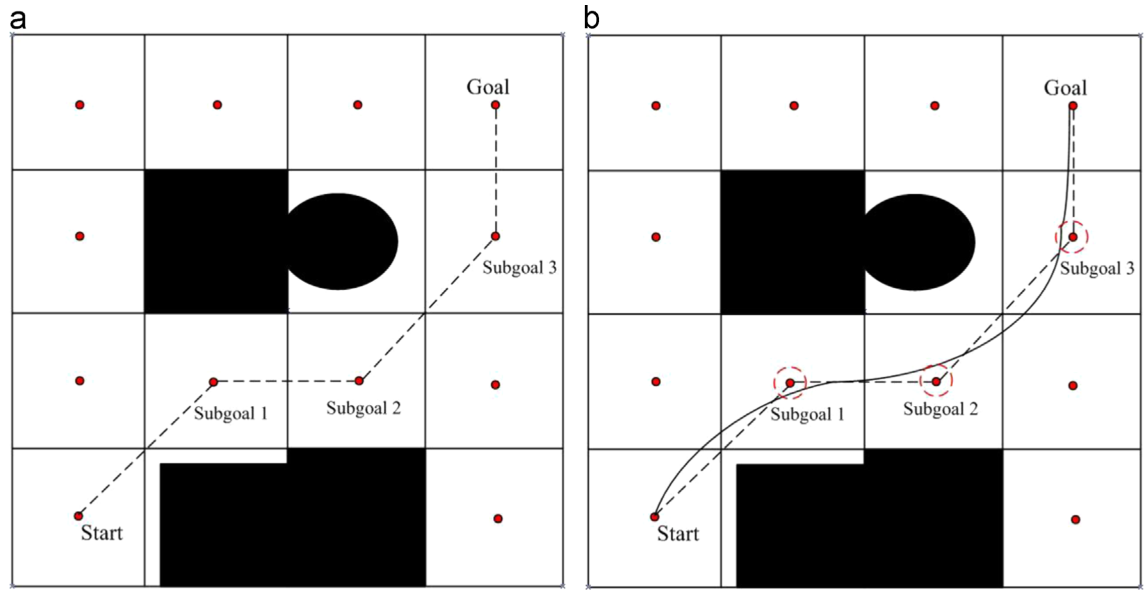


Fig. 3. An example of the planning process of the proposed approach (a) A∗ search for subgoals in the first level (b) smooth the path in the second level.

In the second level, we use a state transition strategy learned by LSPI with kinematic constraints to generate a smooth sub-path between two neighboring sub-targets. Here, when the mobile robot move into a certain range of distance near a subgoal, we consider as it has reached the subgoal. The combination of all sub-paths is the final planning result. An illustration of the process is shown in Fig. 3. More details on how the smooth path is generated will be given in next sub-section.

One advantage of this hierarchical approach is that the LSPI-based local path optimizer can take the kinematic characteristics of the mobile robot into account in the procedure of path optimization. The second advantage is that an efficient and stable approximate policy iteration algorithm, i.e. LSPI, is used in the local path optimizer. The LSPI algorithm can use samples collected arbitrarily to train the local path optimizer offline. Moreover, the good generalization ability of LSPI enables the local path optimizer to deal with the uncertainties in the environment. When there are small changes in the environment, like unexpected obstacles appearing in the way, many planning methods usually need to replan the whole path, which will be costly. But in our approach, when some unexpected obstacles come in the way, it just needs the LSPI-based local path optimizer in the second level to replan the path rather than the whole planner.

3.3. The LSPI-based local path optimizer

For path planning based on LSPI, we separate the process as two sub-problems: learning to approach the target and learning to avoid obstacles. The robot's path which is a sequence of the robot's states has the Markovian property. We can model the process of local path planning as two MDPs in the second level. One MDP is used to describe the behavior of approaching the goal and the other is to model the process of obstacle avoidance. Let d_g denote the distance between the mobile robot location and the target, $\varphi_g \in [0, \pi]$ denotes the angel between the direction of the mobile robot and the target, R_i denote the reading of sensor i ($i = 1,2,\ldots,6$). The two MDPs can be defined as shown in Table 1.

In the process of approaching the target, the robot's next state is determined by the current state and the action it takes. The state vector of the MDP includes the distance d_g between the mobile robot and the target, and the angel $\varphi_g \in [0, \pi]$ between the direction of the mobile robot and the target. When the mobile robot arrives at the target point, the immediate reward is set as 1, otherwise, the reward is set as $\alpha \times d_g - \beta \times \varphi_g$, where α and β are two constants.

During the process of avoiding obstacles, the robot's next states are also determined by the current state and the action it executes. So, the process can be modeled as another MDP. The reward functions are set as shown in Table 2.

In Table 2, $d_{\min} = \min(R_1, R_2, R_3, R_4, R_5, R_6)$, $d_l = \text{average}(R_1, R_2, R_3)$, $d_r = \text{average}(R_4, R_5, R_6)$, $k > 0$ is a proportional constant, L is the safe distance of avoiding obstacles, L_{us} is the distance for an emergency brake. The function is triggered when the reading of any sensor is less than the safety threshold, and the reward is -10.

The LSPI algorithm learns policies from samples, which can be collected offline from sequential episodes of interaction with the process. A sample (s, a, r, s') means action a was taken by the mobile robot at state s and a reward r was received and the resulting next state was s'. During an episode of sampling, the mobile robot starts at an initial state and follows a policy that chooses actions uniformly at random. After executing a random action, the robot observes the next state and receives a reward as set in the above tables. Once the mobile robot reaches the goal or the distance between the mobile robot and obstacles is less than the safety threshold, this sampling episode will end. After the sampling phase, the policies of approaching the target point and avoiding obstacles will be learned by the LSPI algorithm. A block diagram of LSPI is shown in Fig. 4.

Algorithm 1. The LSPI-based local path optimizer training.

Input: x, D, π_0, ε,w_0 **Output:** w

Initialization: $\tilde{A} \leftarrow 0$, $\tilde{b} \leftarrow 0$, $w' \leftarrow w_0$, $\pi' \leftarrow \pi_0$;
while $||w - w'|| < \varepsilon$ **do**
  $w \leftarrow w'$; $\pi \leftarrow \pi'$;
  for each $(s_i, a_i, s_{i+1}, r_i) \in D$ **do**
    $\phi(s_i, a_i) = basis\_function\ (s_i, a_i)$;
    $\tilde{A} \leftarrow \tilde{A} + \phi(s_i, a_i)(\phi(s_i, a_i) - \gamma\phi(s_{i+1}, \pi(s_{i+1}))^T$;
    $\tilde{b} \leftarrow \tilde{b} + \phi(s_i, a_i) r_i$;
  end
    $w' \leftarrow \tilde{A}^{-1}\tilde{b}$;
return w;

Table 1
Definitions of the two MDPs used in the LSPI-based local path optimizer.

	The MDP model of approaching the goal	The MDP model of avoiding obstacles
state	$[d_g, \varphi_g]$	$(R_1, R_2, R_3, R_4, R_5, R_6)$
action	The combination of the speed of the left and right wheel: forward: [0.5, 0.5] turn right: [0, 0.5] turn left: [0.5, 0]	The combination of the speeds of the left and right wheel: forward: [0.5, 0.5] turn right: [0, 0.5] turn left: [0.5, 0]
reward	$\begin{cases} 1, & \text{if the robot reaches the goal} \\ -d_g-\varphi_g, & \text{else} \end{cases}$	See Table 2, where k=0.9

Table 2
The reward function of avoiding obstacles.

	Conditions	Annotation	Reward
1	$d_{\min} \le L_{us}$	The smallest reading of sensors less than or equal to a certain safety valve	-10
2	$d_l \le L$ $d_r \le L$ $d_{\min} > L_{us}$	The average readings of left and right sensors are both less than safe distance of avoiding obstacles	$-k(L-d_l)-k(L-d_r)$
3	$d_l \le L$ $d_r > L$ $d_{\min} > L_{us}$	The average reading of left sensors is less than safe distance of avoiding obstacles	$-k(L-d_l)$
4	$d_l > L$ $d_r \le L$ $d_{\min} > L_{us}$	The average reading of right sensors is less than safe distance of avoiding obstacles	$-k(L-d_r)$
5	$d_l > L$ $d_r > L$ $d_{\min} > L_{us}$	The average reading of left and right sensors are both lager than safe distance of avoiding obstacles	10

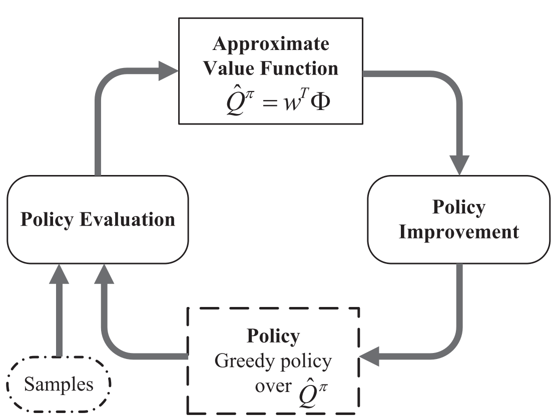


Fig. 4. The block diagram of LSPI.

The value function approximation in LSPI is based on the Bellman equations [39]

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{x' \in S} P(s,a,s') \sum_{a' \in A} \pi(a';s') Q^{\pi}(s',a') \tag{3}$$

where $R(x, a)$ is the reward function, γ is a discount factor.

The target of LSPI is to use a set of linear basis functions to approximate the expected total reward $Q(s, a)$ under an optimal or near-optimal policy. After introducing the coefficients $\boldsymbol{w}$, the state-action value function $Q(s, a)$ can be estimated by the weighted sum of a set of state action bases

$$\hat{Q}^{\pi}(s,a,w) = \sum_{i=1}^{L} \phi_i(s,a) w_i \tag{4}$$

The learning process of LSPI is based on sample data generated from the states transitions, where the sample form is (s, a, r, s'). An incremental update rule can be used to estimate the weights [39]

$$\tilde{w} = A^{-1} b \tag{5}$$

where the matrix A and the vector b can be estimated as:

$$A_{t+1} = A_t + \overrightarrow{\phi}(s_t,a_t)(\overrightarrow{\phi}(s_t,a_t) - \gamma \overrightarrow{\phi}(s_t,a_t'))^T$$
$$b_{t+1} = b_t + \overrightarrow{\phi}(s_t,a_t) r_t \tag{6}$$

Based on the estimation of the state-action value function $\hat{Q}(s, a; w)$, we can get the optimal policy by greedy strategy:

$$\pi(s) = \underset{a \in A}{\operatorname{argmax}} \hat{Q}(s,a;w) \tag{7}$$

Pseudocode of the planning policy learning based on LSPI is given in Algorithm 1. The state action bases are computed by *basis_funciton*()[Line 5].The construction of the basis function is a fundamental issue in LSPI. Polynomial basis functions and radial basis functions (RBFs) are two kinds of popular basis functions. For simplicity, we use polynomial basis functions in this paper.

The state action bases are generated by duplicating the state bases $\phi(s)$ |A| times, and setting all the elements of this vector to 0 except for the ones corresponding to the chosen action. For instance, the state base is $\phi(s) = [\phi_1(s)\ \phi_2(s)]$, and there are two actions $A=\{a_1, a_2\}$. When the action a_1 is taken at state s_1, then the state action base is $\phi(s_1, a_1)=[\phi_1(s_1)\ \phi_2(s_1)\ 0\ 0]$. After learning, the planning policy can be used in the second level of the planner. The algorithm of local path planning based on a learned policy w is shown in Algorithm 2.

Algorithm 2. Path planning based on the local path optimizer.

Input: s_0, s_g, w,   Output: $Path(s_0, s_g)$

```
Initialization: Path(s0, sg) ← {s0};
s ← s0;
while s ≠ sg do
    π = arg max_{a∈A} φ^T(s, a)w;
    s' = nextstate(s, π);
    s ← s';
        Path(s0, sg) ← s;
return Path(s0, sg);
```

In the second level, the local path optimizer uses a state transition strategy learned by LSPI with kinematic constraints to generate a smooth sub-path between two neighboring sub-targets. The process is shown in Fig. 5. Assume there are three actions $\{a_1, a_2, a_3\}$. According to (4), the local path optimizer can compute three approximate state-action value functions at state s_t respect to three actions, as shown in Fig. 5(a). Then, the best action can be selected by (7), taking a_1 for example. The mobile robot executes action a_1 and moves from current state s_t to next state s_{t+1} with kinematic constraints (1). Then, a path episode $\{s_t, s_{t+1}\}$ is generated, as shown in Fig. 5(b). Repeat the above process until the mobile robot reaches the goal and a smooth sub-path between two neighboring sub-targets is generated.

One concern about the LSPI algorithm is whether the sequence of policies learned by the LSPI algorithm converges to a policy that is not far from the optimal one, if it converges at all. Approximate policy iteration had been proven fundamentally sound. When the bounded errors in the policy evaluation process and the policy improvement process can be ensured, the performance of policies generated by approximate policy iteration is not far from the optimal performance [41]. Moreover, as the errors decrease to zero, the difference diminishes to zero as well. The following theorem was given in [39].

Theorem 1. *Let $\pi_0, \pi_1, \pi_2, \ldots, \pi_m$ be the sequence of policies generated by LSPI and let $\hat{Q}^{\pi_1}, \hat{Q}^{\pi_2}, \ldots, \hat{Q}^{\pi_m}$ be the corresponding approximate value functions as computed by LSTDQ. Let ε be a positive scalar that bounds the errors between the approximate and the true value functions over all iterations:*

$$\forall m=1,2,..,||\hat{Q}^{\pi_m}-Q^{\pi_m}||_\infty \le \varepsilon$$

Then this sequence eventually produces policies whose performance is at most a constant multiple of ε away from the optimal performance

$$\limsup_{m\to\infty} ||\hat{Q}^{\pi_m}-Q^*||_\infty \le \frac{2\gamma\varepsilon}{(1-\gamma)^2}$$

This theorem implies that LSPI is a stable algorithm. It will either converge or it will oscillate in an area of the policy space where policies have suboptimality bounded by the value function approximation error ε.

3.4. The A*–LSPI path planning algorithm

Pseudocode of the whole proposed approach is given in Algorithm 3. In Algorithm 3, O is a list used to contain the centers of grids that A* considers for expansion and $O.Insert(s, f(s))$ [Lines 20, 43] inserts point s with the cost estimation $f(s)$ into the list O. $O.Remove(s)$ [Line 42] deletes s from O, and $O.GetMin()$ [Line 22] removes a point with the minimum cost estimation from O and returns it. V is a set of vertices that A* has already expanded. It ensures that A* expands every grid center at most once. $findsubgoal(s_0,s_g)$ finds subgoals from the start location to the goal location, where the parent $parent(s)$ is used after A* terminates [Line 24]. The list $Path(s_0,s_g)$ is used to store the planned path containing a sequence of points.

As listed in Algorithm 3, there are two levels of the procedure: in the first level, A* is used to search for subgoals. If the list O is empty meaning no subgoal is found, then it reports that there is no path [Line 44]. Otherwise, it finds a point with the minimum cost estimation in O [Line 22]. If this point is the goal point, then A* finds a path from the start point to the goal point with a sequence of points and takes these points as subgoals [Line 24]. These subgoals are found by following the parents form the goal to the start point and retrieving them in reverse. In the second level, the planning policy base on LSPI is used to find a feasible path by reaching theses subgoals sequently [Lines 25–31]. Algorithm 2 is executed in $findpath(s, s_{sub})$ [Line 29].

The A* algorithm is complete meaning that it will always find a path if one exists. A* is optimal if no closed set is used under the condition that the heuristic function h never overestimate the actual minimal cost of reaching the goal. If a closed set is used, then h must also be monotonic (or consistent) for A* to be optimal [42,43]. The time complexity of A* is $O(|V| \lg |V|+|E|)$, in which $|V|$ and $|E|$ are the numbers of edges and vertices, respectively, in the graph representation of the discrete planning problem [7]. Therefore, the speed of A* is related to the resolution of the grids. The higher the resolution is the slower A* is. In this paper, we prefer to use girds with low resolution to make the approach fast. However, how to choose a suitable resolution of the grid still needs further study.

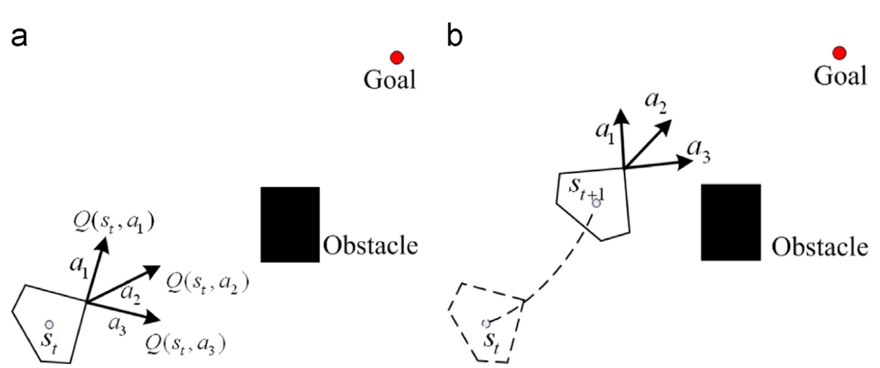


Fig. 5. An illustration of the planning process using the LSPI-based local path optimizer.

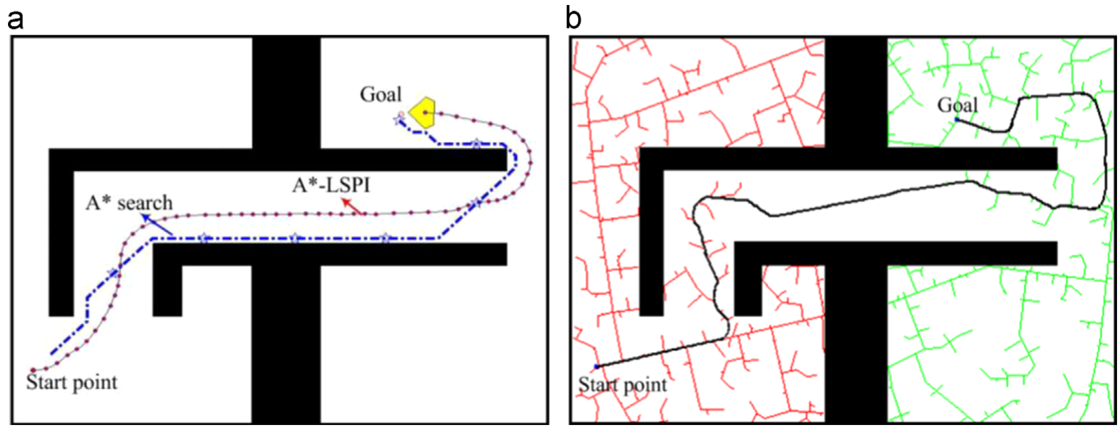


Fig. 6. Planning results of different algorithms in a narrow passage. (a) The result of the proposed algorithm (b) the result of the RRT-Connect algorithm.

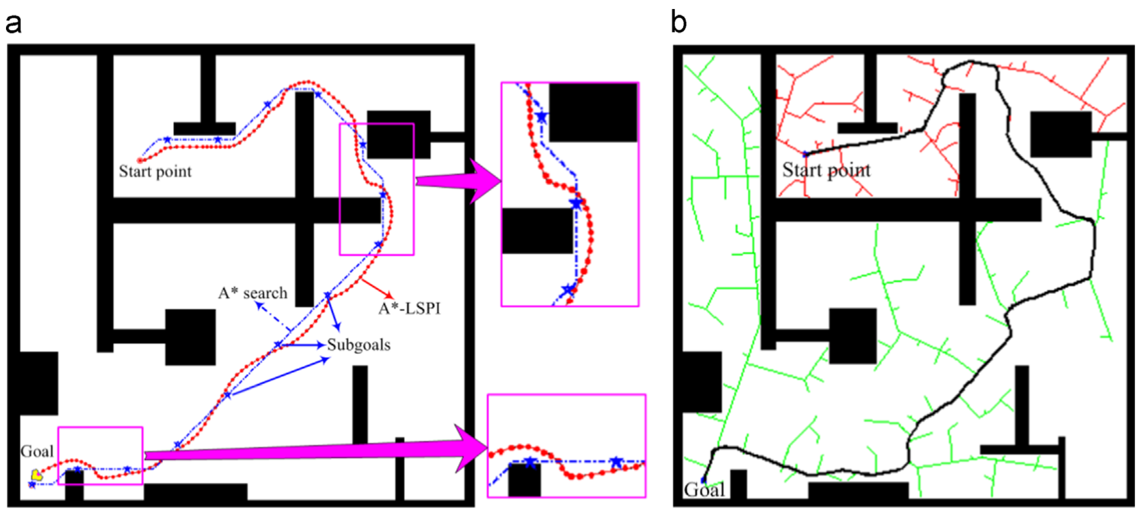


Fig. 7. Planning results of different algorithms. (a) The result of the proposed algorithm (b) the result of the RRT-Connect algorithm.

Table 3
Average planning time and number of path nodes in each method.

Algorithms	Planning time (s)	Path length (step)
A*	8.453	945
RRT-Connect	6.268	1930
A*–LSPI	9.890	1170

Algorithm 3. The A*–LSPI path planning algorithm

Input: s_0, s_g Output: $Path(s_0, s_g)$

Initialization: $g(s_0) \leftarrow 0$, $parent(s_0) \leftarrow s_0$, $O \leftarrow \varnothing$, $V \leftarrow \varnothing$, $S_{sub} \leftarrow \varnothing$;
  $O.Insert\ (s_0, g(s_0)+h(s_0))$;
  while $O \neq \varnothing$ do
    $s \leftarrow O.GetMin()$;
    if $s = s_g$ then
      $S_{sub} \leftarrow$ findsubgoal(s_0, s_g);
      $s \leftarrow s_0$;
      while $s \neq s_g$ do
        for $i = 1$: size(S_{sub}) do
          $s_{sub} \leftarrow S_{sub}(i)$;
          Path(s_0, s_g) $\leftarrow$ findpath(s, s_{sub});
          $s \leftarrow s_{sub}$;
      return $Path(s_0, s_g)$;
    $V \leftarrow V \cup \{s\}$;

Input: s_0, s_g Output: $Path(s_0, s_g)$

    for each s' $\in$ neighbor(s) do
      if s' $\notin$ V then
        if s' $\notin$ O then
          $g(s') \leftarrow \infty$;
          $parent(s') \leftarrow$ NULL;
        if $g(s) + l(s, s') < g(s')$ then
        $g(s') \leftarrow g(s) + l(s, s')$;
          parent(s') $\leftarrow$ s;
          if s' $\in$ O then
            O.Remove(s');
          O.Insert(s', g(s')+h(s'));
  return "no path found";

After given the subgoals by A* quickly, the LSPI-based local optimizer smoothes and optimizes the path by approaching to the subgoals sequentially. The training of the LPSI-based local optimizer is based on samples, which means the training process could be time consuming if the scale of samples is very large. Fortunately, the local path optimizer is trained by LSPI offline. The training process will not influence the running time of the LSPI-based local path optimizer in the procedure of path optimization. Due to the linear architecture, the planning speed of the LSPI-based local path optimizer can be quite fast. Meanwhile, the LSPI-based local path optimizer has good generalization ability after offline learning.

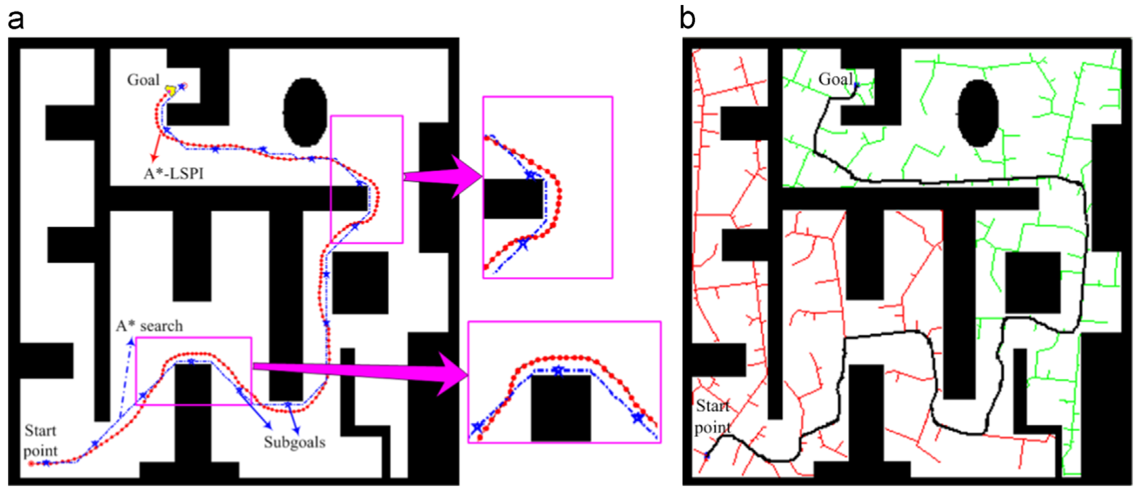


Fig. 8. Planning results of different algorithms. (a) The result of the proposed algorithm (b) the result of the RRT-Connect algorithm.

4. Simulation results and analysis

In this section, the performance of the A*–LSPI algorithm was tested by using the mobile robot model introduced in Section 3.1. Constants about the mobile robot were set as $l_a=0$ m, $l_b=1$ m and $R=0.5$ m in all the simulations. First, we compared the hierarchical path planning algorithm with pure A* search and the RRT-Connect algorithm. Note that the RRT-Connect algorithm is based on random sampling and the paths generated from two run times may be quite different. In this paper, we just showed one of the results obtained by RRT-Connect. In the simulation results of RRT-Connect, the robot is represented by a round point for simplicity, but the state is updated strictly according to the kinematics of the mobile robot. The generalization ability of the method to deal with unexpected obstacles in the environment was also evaluated.

The first simulation tested the performance of different planning methods for the mobile robot passing through a narrow passage (as shown in Fig. 6). The size of the grid maps is 60 m $\times$ 40 m with 1 m resolution. If there is no obstacle in a grid, then the value of this grid is set to 0, otherwise 1. The LSPI-based local path optimizer in the second level was trained offline by using 5000 samples. The samples were sampled from different environments with random policy. The discount factor γ was set to 0.99 and polynomial basis functions were used. After training, the performance of the proposed method was tested. Fig. 6(a) shows the planning results by using pure A* search and our method respectively and Fig. 6(b) shows one result with the RRT-Connect algorithm. As can be seen, the path generated by A* search is shorter than the other two methods. But, as shown in Fig. 6(a) the path generated by A* is too close to the obstacles and not quite safe for the mobile robot. In contrast to A* search and our method, the path obtained by the RRT-Connect algorithm is too long and not smooth either. It can be seen that the path obtained by our method is smooth and safe for the mobile robot and is better than both that of A* search algorithm and the RRT-Connect algorithm.

In the next simulation, we trained the local planning policy offline by using 8000 samples, which were obtained from different environments with random sampling policy. Then the performance of the planner was tested in a new environment and comparisons with other methods were done. The map we used was of size 280 m $\times$ 280 m with 1 m resolution. The planning results are shown in Fig. 7.

As we can see in Fig. 7(a), the result of A* search is not smooth enough for mobile robots to execute. After replanning with the local planner, our method can generate a smoother path and make the length of the path as short as possible. Fig. 7(b) shows the result of the RRT-Connect algorithm. Using the same map shown in Fig. 7, each method was run 20 times and the cost of planning time and the number of path length were averaged. The results are shown in Table 3. Compared with our method, the A* search can obtain a shorter path. However, it does not take the kinematics of the mobile robot into account, thus the path is not feasible for the robot to execute. Compared with the RRT-Connect algorithm, our method can obtain a shorter path and take the kinematics of the mobile robot into account as well.

Then, to test the generalization of the LSPI-based local path optimizer, we changed the map of the environment and did the simulation again without re-learning the local planning policy. Fig. 8 shows the test results. As can be seen, the local planning policy learned by LSPI works well in the new environment without re-learning.

Generally, when part of the environment changes, like unexpected obstacles appearing in the way, it needs the whole planner to replan the path, which usually costs much time. One advantage of the proposed method in this paper is that the LSPI-based local path optimizer has good generalization ability to deal with the uncertainties in the environment after offline learning. When unexpected obstacles come in the way, it just needs LSPI-based local path optimizer in the second level to replan the path rather than the whole planner. Note that the obstacles in this paper mean static obstacles. Static obstacles appearing unexpectedly may be caused by the error of environment sensing and modeling.

We first tested the method on a grid map of size 280 m $\times$ 280 m with 1 m resolution. Fig. 9 shows the replanning results of the proposed method when there are some obstacles appearing unexpectedly. Fig. 9(a) shows the result with no obstacle appearing in the path and Fig. 9(b) shows the replanned result of our approach when there are some static obstacles appearing unexpectedly. When unexpected obstacles come in the way, the LSPI-based local path optimizer can use Algorithm 2 to replan the path without relearning. Details about how the proposed method can generate an obstacle-avoidance and smooth path have been introduced in Section 3.3 the 12th paragraph. The states were updated according to (1). We can see that the subgoals generated by A* are unchanged and the path is replanned only in the second level. The new path is still smooth and safe for the mobile robot, which means that our approach is adaptive to small changes of the environment.

We put the mobile robot in another environment and did the same test. The obstacle map we used was of size 280 m $\times$ 280 m with 1 m resolution. The results are shown in Fig. 10. Compared with Fig. 10(a), we can see in Fig. 10(b) that there are some small

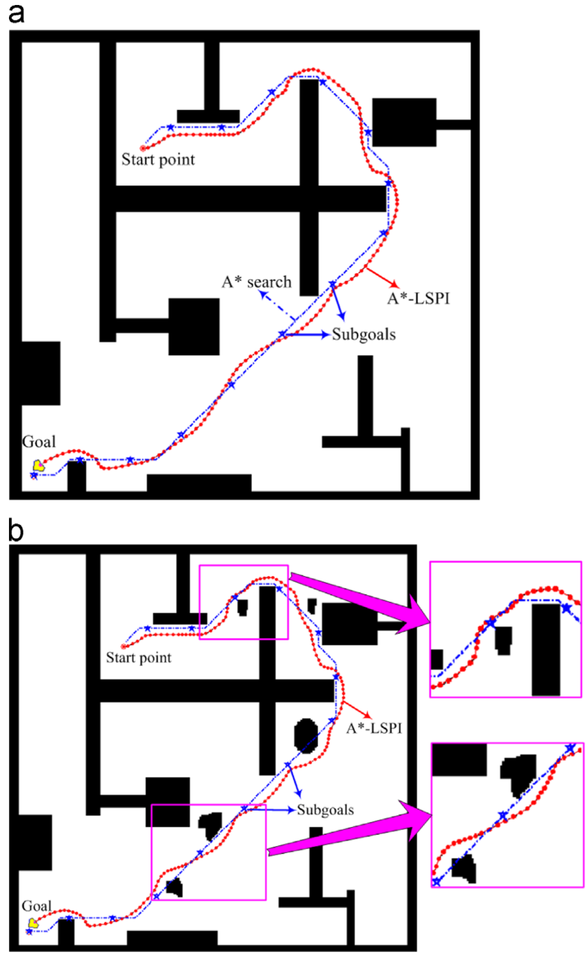


Fig. 9. Planning results of the LSPI based planner with unexpected obstacles in the environment. (a) Without dynamic obstacles (b) with dynamic obstacles.

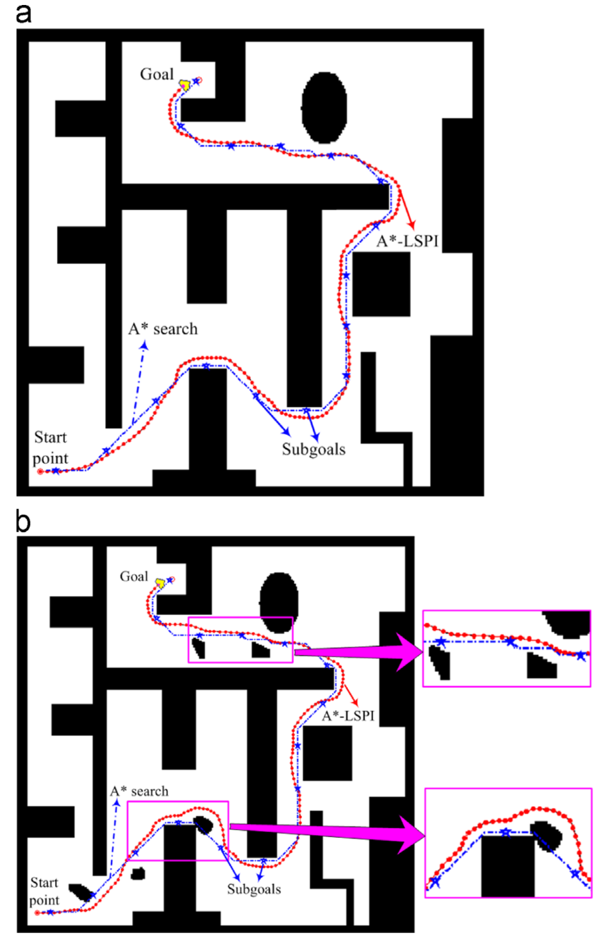


Fig. 10. Planning results of the LSPI based planner with different obstacles in the environment. (a) Planning results without dynamic obstacles (b) planning results with dynamic obstacles.

changes in the environment that some obstacles appear unexpectedly after the A* search. In our approach, there is no need to replan the path in the first level for new subgoals, as long as the obstacles are not too large to completely block the path. With the same subgoals, the second level of the planner can optimize the path accordingly to the obstacles and make sure the new path is smooth and safe for the mobile robot. Thus, it can reduce the whole planning time if the subgoals are not changed. Note that if obstacles are large enough to block several subgoals, then it still needs the first level with A* to replan the path and generate new subgoals.

5. Conclusions

In this paper, a hierarchical path planning approach based on reinforcement learning is proposed. In the proposed approach, we combine the A* algorithm and the LSPI algorithm together to get better path planning results with a two-level structure. In the first level, A* search is used for geometric path planning and several points on the geometric path are chosen as subgoals for further planning in next level. In the second level, we apply a LSPI-based local path optimizer to smooth and optimize the path by sequentially approaching those subgoals obtained from the first level. The local path optimizer uses the LSPI algorithm to learn a near-optimal state transition strategy offline based on samples collected at random with kinematic constraints. Meanwhile, the LSPI-based local path optimizer has good generalization performance which is adaptive to new environments without re-learning. The simulation results have demonstrated the effectiveness of the proposed approach. In future work, the performance of this approach will be proved on a real mobile robot.

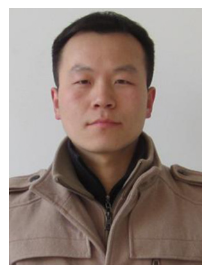
Acknowledgments

This paper is supported by National Natural Science Foundation of China under Grant 91220301, the Program for New Century Excellent Talents in University under Grant NCET-10-0901.

References

[1] R.A. Brooks, Solving the find-path problem by good representation of free space, IEEE Trans. Syst. Man Cybern. SMC-13 (3) (1983) 190–197.
[2] R.A. Brooks, T. Lozano-Perez, A subdivision algorithm in configuration space for find path with rotation, IEEE Trans. Syst. Man Cybern. SMC-15 (2) (1985) 224–233.
[3] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robot. Res. 5 (1) (1986) 90–98.
[4] J. Barraquand, B. Langlosi, J.-C. Latombe, Numerical potential field techniques for robot path planning, IEEE Trans. Syst. Man Cybern. 22 (2) (1992) 224–241.
[5] S.S. Ge, Y.J. Cui, New potential functions for mobile robot path planning, IEEE Trans. Robot. Autom. 16 (5) (2000) 615–620.
[6] H. Choset, K. Lynch, S. Hutchinson, et al., Principles of Robot Motion: Theory, Algorithms, and Implementations, The MIT Press, Boston, MA, 2005.
[7] S.M. LaValle, Planning Algorithms, Cambridge University Press, New York, 2006.
[8] L. Jaillet, J. Cortes, T. Simeon, Sampling-based path planning on configuration-space costmaps, IEEE Trans. Robot. 26 (4) (2010) 635–646.

[9] A.K. Pamosoaji, K.-. Hong, A path-planning algorithm using vector potential functions in triangular regions, IEEE Trans. Syst. Man Cybern.: Syst. 43 (4) (2013) 832–842.
[10] L.E. Kavraki, P. Svestka, J.-C. Latombe, et al., Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robot. Autom. 12 (4) (1996) 566–580.
[11] J.D. Marble, K.E. Bekris, Asymptotically near-optimal planning with probabilistic roadmap spanners, IEEE Trans. Robot. 29 (2) (2013) 432–444.
[12] S.M. LaValle, J.J. Kuffner, Rapidly-exploring random trees: progress and prospects, in: B.R. Donald, K.M. Lynch, D. Rus (Eds.), Algorithmic and Computational Robotics, New Directions, A.K Peters, Wellesley, MA, 2001, pp. 293–308.
[13] J. J. Kuffner, S. M. LaValle, RRT-Connect: an efficient approach to single-query path planning, in: Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, 2000, pp. 995-1001.
[14] M. Kothari, I. Postlethwaite, A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees, J. Intell. Robot. Syst. (2012).
[15] L. Jaillet, J.M. Porta, Path planning under kinematic constraints by rapidly exploring manifolds, IEEE Trans. Robot. 29 (1) (2013) 105–117.
[16] D. Hsu, J.-C. Latombe, R. Motwani, Path planning in expansive configuration spaces, in: Proceedings of IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, 1997, pp. 2719-2726.
[17] E. Plaku, K.E. Bekris, B.Y. Chen, et al., Sampling-based roadmap of trees for parallel motion planning, IEEE Trans.n Robot. 21 (4) (2005) 597–608.
[18] X. Yang, M. Moallem, R.V. Patel, A layered goal-oriented fuzzy motion planning strategy for mobile robot navigation, IEEE Trans. Syst. Man. Cybern.Part B: Cybern. 35 (6) (2005) 1214–1224.
[19] X.-C. Lai, S.S. Ge, A.A. Mamum, Hierarchical incremental path planning and situation-dependent optimized dynamic motion planning considering accelerations, IEEE Trans. Syst. Man Cybern.Part B: Cybern. 37 (6) (2007) 1541–1554.
[20] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (2) (1968) 100–107.
[21] H. Qu, S.X. Yang, A.R. Willms, et al., Real-time robot path planning based on a modified pulse-coupled neural network model, IEEE Trans. Neural Netw. 20 (11) (2009) 1724–1739.
[22] X. Li, Y. Ma, X. Feng, Self-adaptive autowave pulse-coupled neural network for shortest path problem, Neurocomputing 115 (2013) 63–71.
[23] R. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, The MIT Press, Cambridge, Massachusetts, USA, 1998.
[24] A. Konar, I.G. Chakraborty, S.J. Singh, et al., A deterministic improved Q-learning for path planning of a mobile robot, IEEE Trans. Syst. Man Cybern.: Syst. 43 (5) (2013) 1141–1153.
[25] E. Rimon, D.E. Koditschek, Exact robot navigation using artificial potential functions, IEEE Trans. Robot. Autom. 8 (5) (1992) 501–518.
[26] J. Chuang, N. Ahuja, An analytically tractable potential field model of free space and its application in obstacle avoidance, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 28 (5) (1998) 729–736.
[27] J. Ren, K.A. McIsaac, R.V. Patel, Modified Newton's method applied to potential field-based navigaiton for mobile robots, IEEE Trans. Robot. 22 (2) (2006) 384–391.
[28] T. Lozano-Perez, An algorithm for planning collision-free paths among polyhedral obstacles, Commun. ACM 22 (1979) 560–570.
[29] H. Rohnert, Shortest path in the plane with polygonal obstacles, Inf. Process. Lett. 23 (1986) 71–76.
[30] J.F. Canny, B.R. Donald, Simplified voronoi diagrams, Discret. Comput. Geom. 24 (3) (1988) 219–236.
[31] O. Takahashi, R.J. Schilling, Motion planning in a plane using generalized voronoi diagrams, IEEE Trans. Robot. Autom. 5 (2) (1989) 143–150.
[32] N. Kalra, D. Ferguson, A. Stentz, Incremental reconstruction of generalized Voronoi diagrams on grids, Robot. Auton. Syst. 57 (2009) 123–128.
[33] D. Dolgov, S. Thrun, M. Montemerlo, et al., Path planning for autonomous vehicles in unknown semi-structured environments, Int. J. Robot. Res. 29 (5) (2010) 485–501.
[34] L. Murphy, P. Newman, Risky planning on probabilistic costmaps for path planning in outdoor environments, IEEE Trans. Robot. 29 (2) (2013) 445–457.
[35] K. Fujimura, H. Samet, A hierarchical strategy for path planning among moving obstacles, IEEE Trans. Robot. Autom. 5 (1) (1989) 61–69.
[36] D. Zhu, J.-C. Latombe, New heuristic algorithms for efficient hierarchical path planning, IEEE Trans. Robot. Autom. 7 (1) (1991) 9–20.
[37] Y. Zhang, D. Gong, J. Zhang, Robot path planning in uncertain environment using multi-objective particle swarm optimization, Neurocomputing 103 (2013) 172–185.
[38] S.J. Bradtke, A.G. Barto, Linear least-squares algorithms for temporal difference learning, Mach. Learn. 22 (1996) 33–57.
[39] M.G. Lagoudakis, R. Parr, Least-squares policy iteration, J. Mach. Learn. Res. 4 (2003) 1107–1149.
[40] Xin Xu, Chunming Liu, Simon Yang, Dewen Hu, Hierarchial approximate policy iteration with binary-tree state space decomposition, IEEE Trans. Neural Netw. 22 (12) (2011) 1863–1877.
[41] D.P. Bertsekas, J.N. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, Belmont, Massachusetts, 1996.
[42] 〈http://en.wikipedia.org/wiki/A*_search_algorithm〉.
[43] J. Pearl, Heuristics: Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley Pub. Co., Boston, Massachusetts, USA, 1984.
[44] Xin Xu, Zhongsheng Hou, Chuanqiang Lian, Haibo He, Online learning control using adaptive critic designs with sparse kernel machines, IEEE Trans. Neural Netw. Learn. Syst. 24 (5) (2013) 762–775.



Lei Zuo received the Bachelor's degree and the Master's degree both from the College of Mechatronics and Automation (CMA), NUDT, PR China, in 2009, and 2011, respectively. He is currently a Ph.D. student at the Institute of Unmanned Systems, CMA, NUDT, PR China.

His research interests include reinforcement learning, approximate dynamic programming (ADP), and autonomous vehicles. He has coauthored more than 10 papers in international journals and conferences.

Xin Xu received the B.S. degree in electrical engineering from the Department of Automatic Control, National University of Defense Technology (NUDT), Changsha, PR China, in 1996 and the Ph.D. degree in control science and engineering from the College of Mechatronics and Automation (CMA), NUDT, PR China. He has been a visiting scientist for cooperation research in the Hong Kong Polytechnic University, University of Alberta, University of Guelph, and the University of Strathclyde, UK, respectively. He is currently a Full Professor at the Institute of Unmanned Systems, CMA, NUDT.

His research interests include reinforcement learning, approximate dynamic programming (ADP), machine learning, robotics and autonomous vehicles. He has coauthored four books and published more than 90 papers in international journals and conferences. He is a Committee Member of the IEEE Technical Committee on Approximate Dynamic Programming and Reinforcement Learning (ADPRL) and the IEEE Technical Committee on Robot Learning. He has served as a PC member or Session Chair in many international conferences. Dr. Xu is one of the recipients received the 1st class Natural Science Award from Hunan Province, PR China, in 2009 and the Fork Ying Tong Youth Teacher Fund of China in 2008.