doi:10.3772/j.issn.1006-6748.2016.03.013

Path planning based on sliding window and variant A* algorithm for quadruped robot①

Zhang Hui (张 慧)*, Rong Xuewen②*, Li Yibin*, Li Bin*, Zhang Junwen, Zhang Qin**
(*School of Control Science and Engineering, Shandong University, Jinan 250061, P. R. China)
(**School of Electrical Engineering, Jinan University, Jinan 250022, P. R. China)

Abstract

In order to improve the adaptability of the quadruped robot in complex environments, a path planning method based on sliding window and variant A* algorithm for quadruped robot is presented. To improve the path planning efficiency and robot security, an incremental A* search algorithm (IA*) and the A* algorithm having obstacle grids extending (EA*) are proposed respectively. The IA* algorithm firstly searches an optimal path based on A* algorithm, then a new route from the current path to the new goal projection is added to generate a suboptimum route incrementally. In comparison with traditional method solving path planning problem from scratch, the IA* enables the robot to plan path more efficiently. EA* extends the obstacle by means of increasing grid g-value, which makes the route far away from the obstacle and avoids blocking the narrow passage. To navigate the robot running smoothly, a quadratic B-spline interpolation is applied to smooth the path. Simulation results illustrate that the IA* algorithm can increase the re-planning efficiency more than 5 times and demonstrate the effectiveness of the EA* algorithm.

Key words: quadruped robot, path planning, sliding window, A* algorithm

0 Introduction

Quadruped robots such as BigDog[1], LS3[2], HyQ[3] and Scalf[4] are superior to wheeled and tracked robots since they have higher terrain adaptability and possess greater mobility and flexibility. When such robots are operated in unstructured outdoor environment autonomously, the path finding problem has to be solved real time. BigDog robot has tested the variant A* path planner in unstructured forest, and the result verifies the feasibility of the algorithm. HyQ robot presents a comparison of a set of path planning algorithms: Dijkstra, A* and ARA*, and results show that A* and ARA* are more efficient compared with Dijkstra[5]. Both BigDog and HyQ use grid map to describe environment and A* or improved algorithm are used to plan the route.

The path planning algorithm based on grid describes the environment through regularly sized grid cells and a search is run on the grid to solve the point-to-point find-path problem[6]. A*[7] algorithm commonly used and extensively studied can generate optimal paths[8], but it is a static algorithm. It means that if the environment changes, the old path is invalidated and the A* algorithm must be re-run from scratch. This is inefficient since most edge costs do not change between re-planning episodes[9]. The most popular solution to this problem is Focussed Dynamic A* (D*)[10] which combines the efficiency of heuristic and incremental searches. Then, D* Lite, the simplified version of D* was proposed[11]. Beside these methods, the LPA*[12] and ARA*[13] were also proposed within the past decade.

A* or its improved algorithm requires that is at least one position is constant between the robot and goal. But for the quadruped robot, the robot position and the goal projection change over time, on this occasion, all of the path planners mentioned above have to re-plan the path from scratch. So it is necessary to design a fast and efficient path planning algorithm for the quadruped robot. In addition, the path planning algorithm based on grids always treats the robot as a point meanwhile extends the obstacle with a specific size.

① Supported by the National Natural Science Foundation of China (No. 61233014, 61305130, 61503153), the National High Technology Research and Development Program of China (No. 2015AA042201), the Shandong Provincial Natural Science Foundation (No. ZR2013FQ003, ZR2013EEM027), and China Postdoctoral Science Foundation (No. 2013M541912).

② To whom correspondence should be addressed. E-mail: rongxw@sdu.edu.cn

Received on Aug. 10, 2015

But for the quadruped robot, it is hard to select the expansion radius. So how to describe the obstacle also needs to be solved.

In order to solve these problems, the incremental A* path planner (IA*) and the A* algorithm having obstacle grids extending (EA*) are proposed in this paper. IA* firstly searches a path based on A* algorithm, then the optimal route from the current path to the new goal projection is added to generate a suboptimum route incrementally. Compared with the traditional method solving the path re-planning problem from scratch, the IA* algorithm enables the robot to plan path more efficiently. EA* extends the obstacle by means of increasing the g-value of the grid, which makes the route far away from the obstacle and avoids blocking the narrow passage.

The organization of this paper is as follows. In Section 1, the methods of environment modeling and goal projection selection are described. In Section 2, the improved path planning method IA* and EA* are introduced. In Section 3 the path smoothing method using B-splint is introduced. In Section 4, some of the simulation is done to demonstrate the capability of proposed path planning method. In Section 5, we discuss future work is discussed and conclusions are drawn.

1 Environment description and goal generation

As the range of the environment perception sensor is restricted and considered the efficiency of the path planner, it is unreal to structure a map as large as possible to contain the robot and the goal point at the same time especially when the goal point is far away from the robot. So in this paper, a sliding window is used to update the environment information, and the projection of the goal point is selected as the temporary goal when the quadruped robot is operated in unstructured environment.

In the sliding window, a grid map with regularly sized grid cells is used to represent the environment. The goal and robot are known within the grid and modeling the navigation problem as a graph-traversal problem on an eight-connected grid with edges that are either traversable or un-traversable. The obstacle points are expanded according to the size of the robot. Fig. 1 shows the environment map, the area with a pentagon or triangle is the position of the robot and goal, black grids are the obstacle and adjacent blocks are the extended area. The route with gray color is planned by A*.

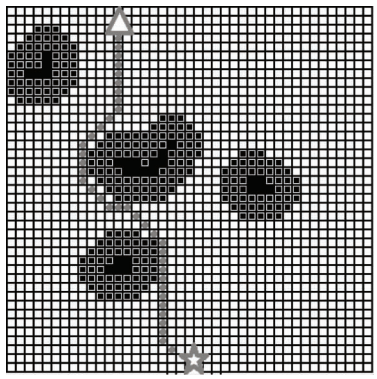


Fig. 1 Environment representation

If the goal point is in the area of sliding window, the path planner will find a route from the robot to goal. On the other hand, if the goal point is out of the sliding window, a goal projection will be selected as the temporary goal. The selection method is that if the intersection between the sliding window and the line connecting the robot and the goal is walkable, a route from the robot to the intersection will be planned; but if the intersection is un-walkable, the planner will search the sliding window margin from the intersection until a walkable grid is found, and then the planner will search a path from the robot to this intersection. Fig. 2 shows the principle of goal point selecting.

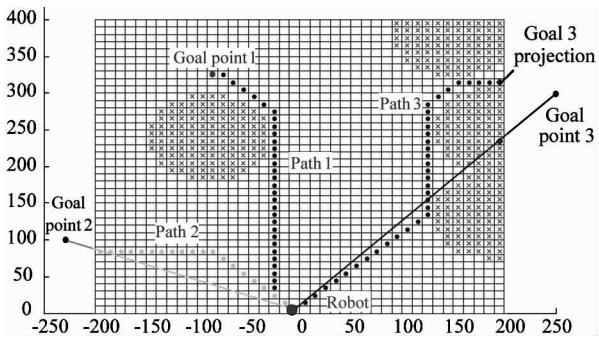


Fig. 2 Goal generation

2 Improved path planning method

2.1 A* algorithm

In the field of heuristic searching algorithm, A* algorithm is widely used in path planning and graph traversal, which plans a path from an initial state $s_{star} \in S$ to a goal state $s_{goal} \in S$, where S is the set of states in some finite state space. This algorithm uses an evaluation function:

$$f(s) = g(s) + h(s) \tag{1}$$

to provide guidance and selection to the expanded nodes of the list OPEN, where $f(s)$ is the assessment of every node which will be searched possibly; $g(s)$ is

the cost from start node to current node; $h(s)$ is the assessment from current node to target node. Then the f-value of the goal is the length of the shortest path.

A complete search process of A* algorithm is shown in Fig. 3(a) and the grid size is 10 × 10cm. The pentagon is the robot and triangle is the target. Obstacles are shown with square blocks. The g-value, h-value and f-value are shown respectively in the lower-left, lower-right and top-left corners of the grid. When the environment is updated in the next time, the path has to be re-planned from scratch, as shown in Fig. 3(b).

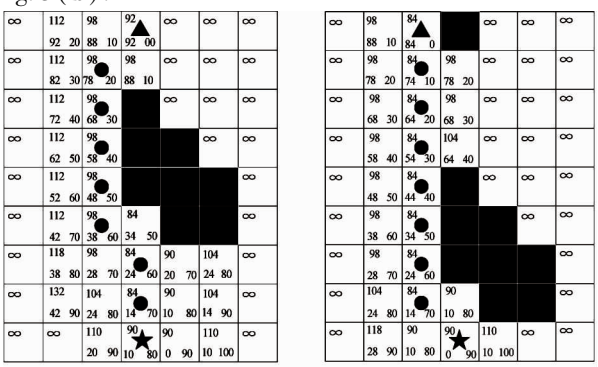


Fig. 3 Search process of the A* algorithm

When the quadruped robot is walking in complex terrain, the changing of the robot position and target are small between adjacent moments, and there are hardly great changes on the environment. So it is in low efficiency that the path is re-planned from scratch every time. In order to solve this problem, IA* is proposed in this part.

2.2 IA* method

In the implementation process of the A* algorithm, the cost of the path point r is stored in $g(r)$, $r \in R$ which is the set of path points. As the route selected by A* is shortest, so the $g(r)$ represents the shortest path between the starting point and the waypoint. In order to make the path planning speed faster, the $g(r)$ is used to build the incremental search algorithm and the search process is as following:

If the optimal path has found by the A* algorithm, shown in Fig. 3(a), then the environment information and the route will be translated firstly and the evaluation function is updated as

$$g(r') = g(r') - g(r_b') \tag{2}$$

$$h(r') = |x_t - x_{r'}| + |y_t - y_{r'}| \tag{3}$$

$$f(r') = g(r') + h(r') \tag{4}$$

where r_b' is the start point of the remainder route; r' is the waypoint designed in the previous time and the robot has not achieved, x_t, y_t is the x-coordinate, y-coordinate of the new target and $x_{r'}$, $y_{r'}$ is the x-coordinate, y-coordinate of the r'. The updated result is shown in Fig. 4(a).

At the same time, the waypoint which has the minimal f-value will be searched out from the end vertex of the route, as the empty circle in Fig. 4(a). Then a new route from this point to the target will be planned by A* algorithm. The new route is shown in Fig. 4(b). At last, the new path will be added to the translated route and constitutes a full path connecting the robot and goal. Noticed that this route does not assure the total length is the shortest, so we call it suboptimum route. The route connection is achieved through updating the g-value of the new route:

$$g(r'') = g(r'') + g(r_e') \tag{5}$$

where r'' is the new route point, r_e' is the end point of the translated route. So $R = \{r', r''\}$ is the full path and $\{g(r'), g(r'')\}$ will be used in the next cycle.

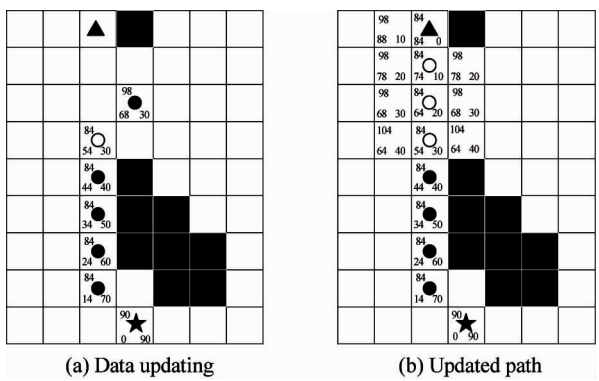


Fig. 4 Search process of the IA* algorithm

In addition, the route generated by IA* has better coherence compared with A* algorithm. Fig. 5 shows a situation which A* method will re-plan a new path totally different from the previous, but IA* method can extend the path incrementally. This feature of IA* can reduce the fluctuation of the robot orientation and make the robot run smoother.

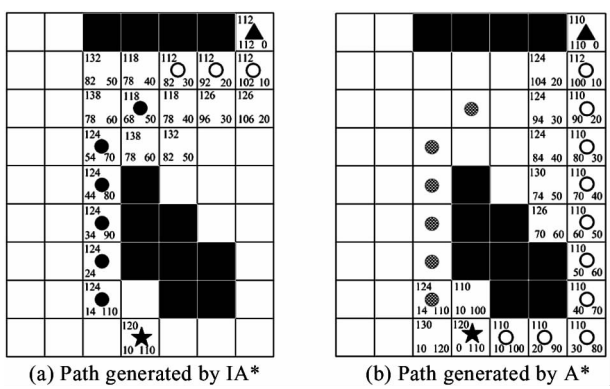
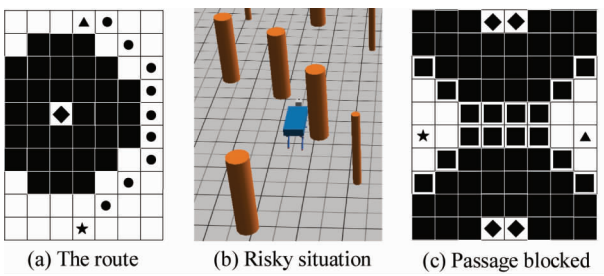


Fig. 5 Path comparison

The route planned by IA* algorithm is the extension of the previous, so is more efficient than re-planning from scratch. The simulation results in Section 4 show the path comparison between generated by A* and IA*. Sometimes, the old route is likely to be influenced by the new obstacle, at this point A* is re-used to find a new path.

2.3 EA* method

The path planning algorithm based on grids always treats the robot as a point and the obstacles are expanded according to the size of the robot, which is shown in Fig. 6(a). But for the quadruped robot, it is hard to select the expansion radius, if using the minimum distance to expand the obstacle, the robot may crash into the obstacle, as shown in Fig. 6(b), but it also faces a problem that the narrow way will be blocked if the obstacle expands using the maximum distance, as shown in Fig. 6(c). To solve this problem, this part proposes the EA* algorithm, which changes the route through increasing the g-value of the additional expanded grids, and makes the route far away from the obstacle and avoids blocking the narrow passage.



(a) The route (b) Risky situation (c) Passage blocked

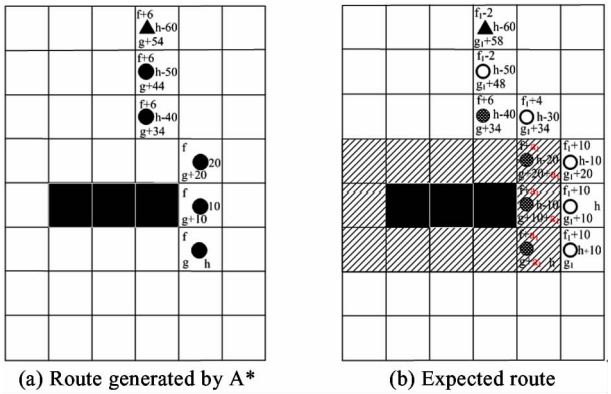
Fig. 6 Obstacle description

As shown in Fig. 7(a), there is a part of route generated by A*, now if it is wanted to extend the obstacle and generate a new path shown in Fig. 7(b) with empty circle. From the principle of A* it can be seen that this new route can be generated only when $f(r_{old})$ is larger than $f(r_{new})$, in order to meet this requirement, the g-value of the expanded grid is added with a_1 and meeting the equation:

$$f + a_1 = g + h + a_1 > f_1 + 10 = g_1 + h + 10 = g + \alpha + h + 10 \tag{6}$$

where α is the difference of initial g-value between the new and old routes. So can be got the value of a_1

$$a_1 > \alpha + 10 \tag{7}$$



(a) Route generated by A* (b) Expected route

Fig. 7 Route modification

If the new route is wanted to be far away from the obstacle, two or three grids or more shown in Fig. 8, the recursion formulae about the a_i which is the increment of g-value can be obtained from the adjacent relation among different grids. The $dflag(s)$ is used to store the danger level which is determined by the extended obstacle grids, and the increment of g-value is store in $igcost(s)$.

$$a_1 > \alpha + 10$$

$$a_2 > a_1 + 10 + \alpha = 2 \times a_1 \tag{8}$$

$$\cdots\cdots$$

$$a_n > a_{n-1} + 10 + \alpha = n \times a_1$$

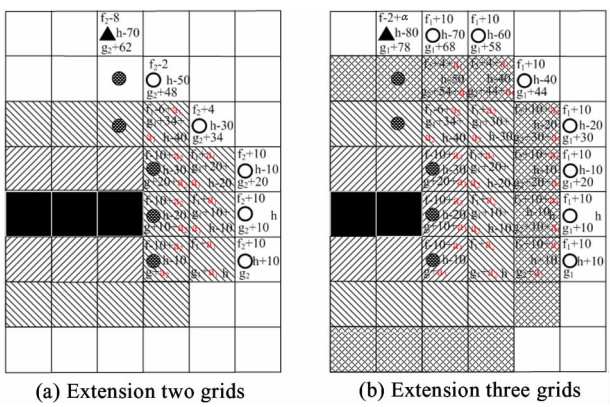
$$dflag(s) = n - dis(s) + 1 \tag{9}$$

$$dis(s) = \begin{cases} |y_s - y_{obst}| & if\ (x_s - x_{obst} = 0) \\ |x_s - x_{obst}| & if(y_s - y_{obst} = 0) \\ \min(|x_s - x_{obst}|, |y_s - y_{obst}|) & else \end{cases} \tag{10}$$

$$igcost(s) = dflag(s) \times a_1 \tag{11}$$

Therefore the $g(s')$ can be rewritten as

$$g(s') = g(s) + c(s,s') - igcost(s) + dflag(s') \times a_1 \tag{12}$$



(a) Extension two grids (b) Extension three grids

Fig. 8 The route with obstacle extension

Because the regular grid map is used in this work, the value of α is 4, 6, 10 or 14, which is shown in

Fig. 9. When α is 4 or 10, there is a public edge between adjacent grids, but it becomes public vertex when α is 6 or 14. From the Fig. 7 and Fig. 8 we can see there are all public edges between the new route and the extended grids, so the value of α is 4 or 10.

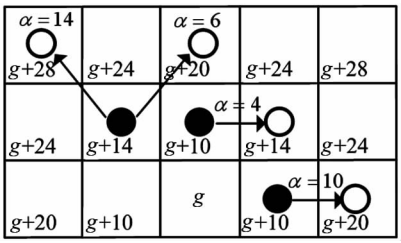
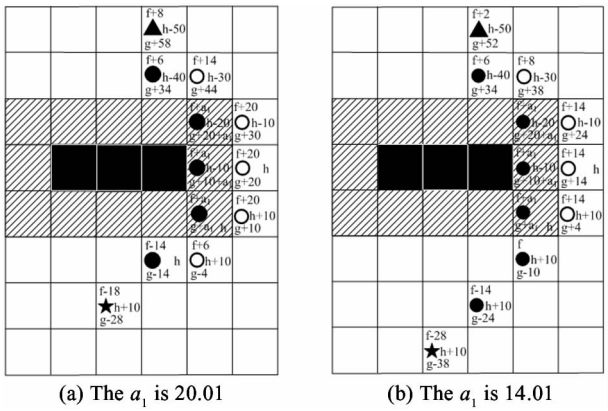


Fig. 9 The value of α

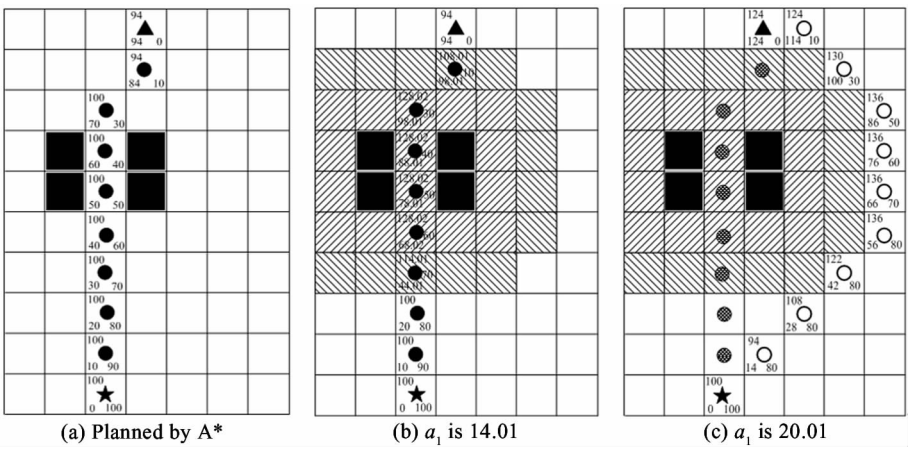
If the value of α is selected as 10, then the value of $a_1 > 10 + \alpha = 20$ assuming a_1 is 20.01, and the new route as shown in Fig. 10(a) and Fig. 10(b) can be planned. But for the route of Fig. 10(a), it is hardly appears during the running of robots because the route tends to the margin of the obstacle so the robot potion is always far away from the obstacle. Fig. 10(b) is closer to the real route and the α value is 4, so a_1 of 14.01 can meet the requirement of route scaling out problems.



(a) The a_1 is 20.01 (b) The a_1 is 14.01

Fig. 10 The route with different

In addition, it is more difficult to go through the grid with bigger a_i, and sometimes it may lead to block the narrow access if a_i is too large. Fig. 11(a) shows an original route planned by A*, and Fig. 11(b) and Fig. 11(c) show the route when a_i is 14.01 and 20.01. From this figure it can be seen that the narrow passage will be blocked if the value of a_1 is 20.01.



(a) Planned by A* (b) a_1 is 14.01 (c) a_1 is 20.01

Fig. 11 Route comparison with different a_1

Besides, if the a_i is linear increased using Eq. (12), the route which is shown in Fig. 12(a) may be generated. This route tends to the grid with smaller a_i value, though this increases the safety of the robot, it also makes the robot turn at a large angle. In order to maintain the stable change of the route, the value of a_i is adjusted according the $dflag(s)$, if $dflag(s) \neq dflag(s')$, then Eq. (11) is used, but when $dflag(s) = dflag(s')$, Eq. (12) is used.

$$g(s') = g(s) + c(s,s') - igcost(s) + dflag(s') \times 14.01 - 0.02 \quad (13)$$

Through this equation the route which is shown in Fig. 12(b) can be obtained, this route is smoother than the previous and it is in favor of the steering stability of the robot.

3 Path smoothing

The path output by searching over a regular grid is jagged. The large changes in heading in this path can induce undesirable steering commands[14]. In order to make the route smoother, a quadratic B-splines curve is used to smooth the path. B-splines are a more general type of curve than Bezier curves, and they are con-

structed from polynomial pieces, joined at certain knots values. Once the knots are given, it is easy to compute the B-splines recursively. B-splines functions[15,16] have properties that make it suitable for smooth path planning. Much work has been devoted to planning or smoothing trajectories using B-splines[17,18].

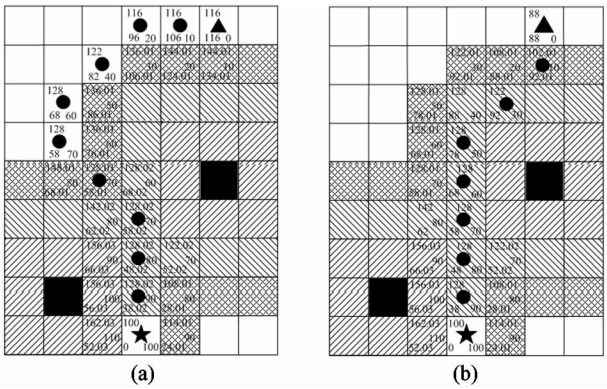


Fig. 12 Route improvement

In order to use B-splines to smooth the route, the route is firstly decomposed into several segments according to the slope. Then, the endpoints are selected out. Fig. 13(a) shows the result, the gray point is the route, the pentagon is the start point and the triangle is the target, inflection points are shown with gray cycle. Then the closely spaced inflection points will be combined. Eq. (14) is used to generate the smooth path, and the smoothed route is shown in Fig. 13(b).

$$P_{0,2}(t) = \frac{1}{2}[1 \quad t \quad t^2]\begin{bmatrix} 1 & 1 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix}\begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} \tag{14}$$

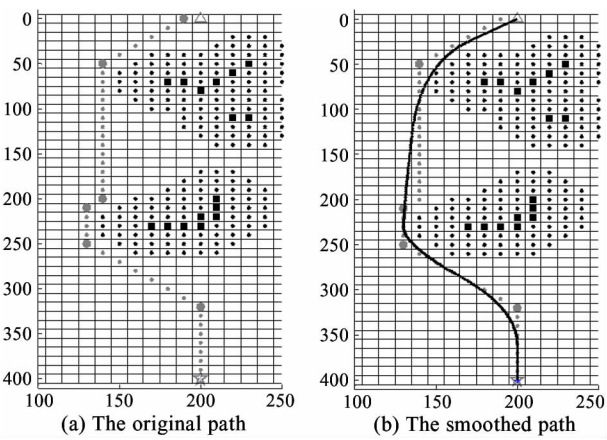


Fig. 13 Path smooth using B-spline

where P_0, P_1 and P_2 are the adjacent inflection points, and $P_{0,2}(t)$ is the smooth points connecting P_0, P_1 and P_2. $t \in [0,1]$.

4 Simulation

In order to evaluate the path planning method proposed in this article, dynamic simulation is made by using the Webots robotics simulator which is a development environment used to model, program and simulate mobile robots to design complex robotic setups, with one or several, similar or different robots, in a shared environment.

The quadruped robot and the environment model used for simulation are shown in Fig. 14(a). The length of the robot model is 1.1m and the width is 0.5m. The simulation environment is designed as unstructured forest environment. The robot perceives environment through the SICK LMS291 and Kinect sensor, its position obtained by GPS. A compass is used to measure the direction angle of the robot.

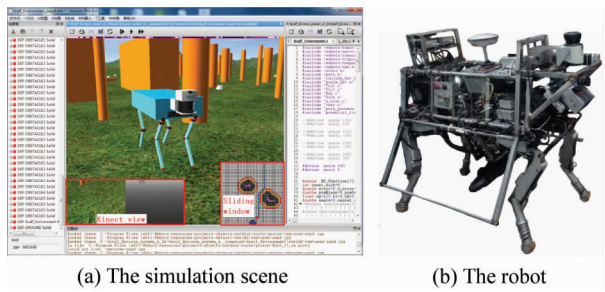


Fig. 14 The simulation scene

A sliding window with regularly sized grid cells of 10cm × 10cm is used to represent the environment, and the total size of the window is 4.0m × 4.0m. Obstacle points are extended with 3 grids so as the quadruped robot can be considered as a particle point.

4.1 IA* method simulation

In our simulation, differences of the path planning result using traditional A* and the IA* method are tested. Fig. 15 shows a group of sequential path planned using A* and IA* method with the environment updating in the sliding window. In this figure, the extended area is shown with gray grids and obstacles are surrounded by it, other black grids are searched area. The smoothed route is shown with gray line. From this figure it can be seen that using the IA* method, the amount of the searched grid in the new planning cycle can be reduced dramatically.

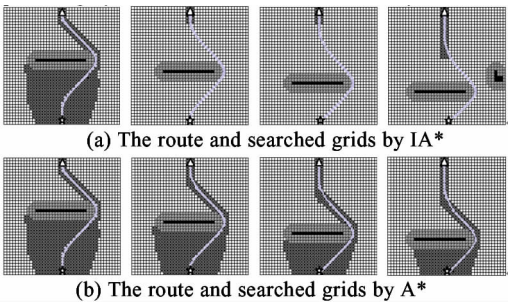


Fig. 15 Path planning using IA* and A*

Fig. 16 shows the route planning results. The circle and square indicate the start and target point. The black route is generated by A* and gray route is generated by IA*. This figure shows that the route is not completely the same between these two kinds of method and the robot may collide with the obstacle when using A*, but the route generated by IA* can get to the target with no collision.

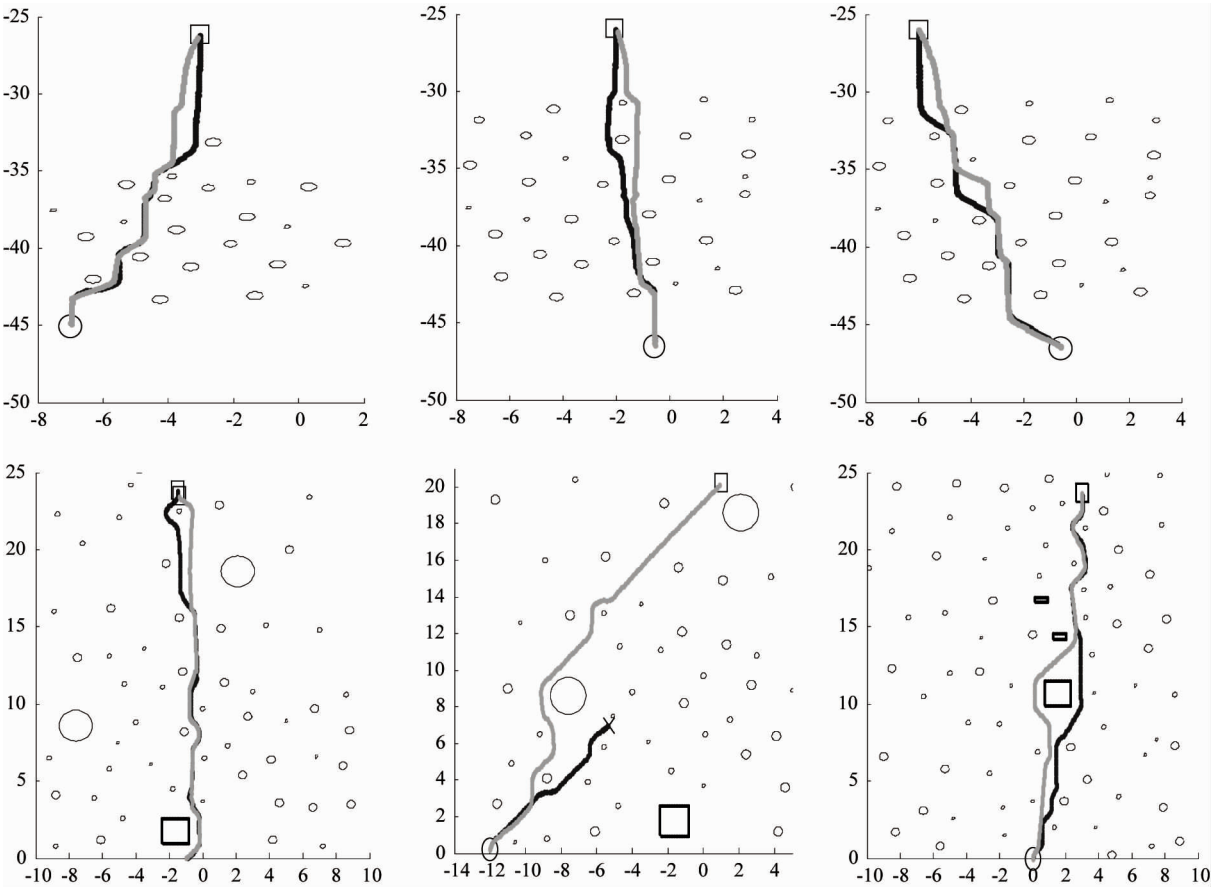


Fig. 16 The route generated by A* and IA*

Table 1 shows the comparison of the route length and travel time, in which the route length planned by IA* is almost less than planned by A*, but the travel time is the opposite. It is because the smoothed route generated by IA* tends to oblique line connecting the starting point and goal, but it is more like a broken line generated by A*, as shown in Fig. 17. On the other hand, there are more curves in the route generated by IA*, which leads to spending more time.

Table 1 Comparison of the length and time

Time	Total length (m)		Total time (s)	
	A*	IA*	A*	IA*
1	26.59	26.53	90.2	96.2
2	21.53	21.38	73.71	78.46
3	20.03	19.9	73	75.3
4	21.33	21.51	71.34	72.31
5	22.74	23.02	80.31	83.43
6	17.52	16.95	64.12	68.54
7	21.87	21.21	75.73	79.56
8	17.89	17.48	62.24	64.45
9	19.03	18.49	65.71	65.39
10	15.33	14.63	59.24	55.43

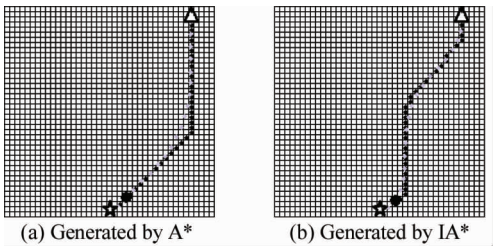


Fig. 17 Route comparison

Though there are little differences between the route and its length generated A* and IA*, the difference of search efficiency is very large. Fig. 18 shows the number of grid which is searched at each planning using these two methods. The black line indicates the number searched by A* and the gray line indicates the number searched by IA*. From this figure we can see that the grid number searched by IA* is less than A*.

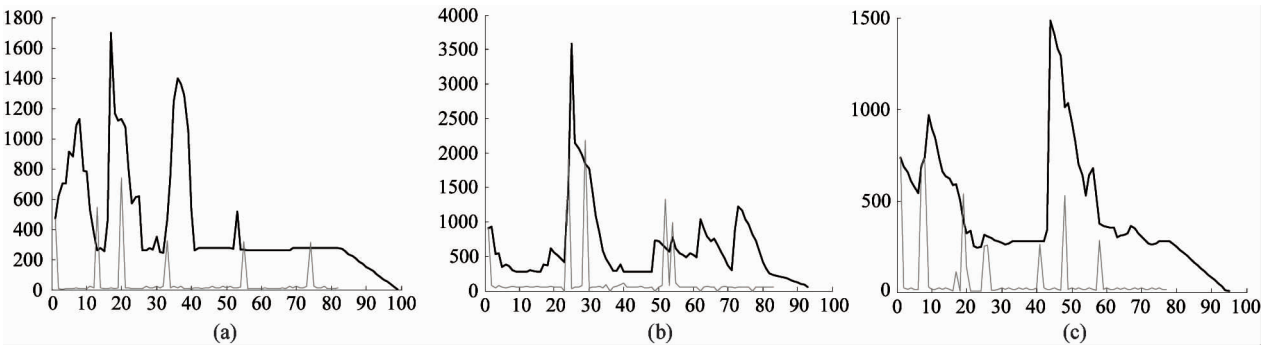


Fig. 18 The number of searched grids

Only when IA* uses A* to re-plan the route to peaks will appear the gray line. Table 2 shows the total number of grids searched by these two kinds of methods, it indicates that the efficiency of IA* is more than 5 times of the A*.

Table 2 Total number of grids searched by A* and IA*

Time	Total number of grids		A*/IA*
	A*	IA*	
1	39257	5510	7.12
2	38128	4435	8.60
3	38319	4071	9.41
4	53006	9035	5.86
5	41264	5610	7.35
6	42973	3712	11.57
7	23180	4177	5.5
8	41864	5861	7.14
9	19185	3120	6.15

4.2 EA* method simulation

In order to show the difference between the route generated by A* and EA* intuitively, a route simulation of avoiding a single obstacle is done and the path is shown in Fig. 19. Fig. 19(a) is the 3D simulation scene and there is a square obstacle with the size of 1.6×1.6m. Fig. 19(b) is the route generated by A* and EA*, and the obstacle is shown with black line. The line shown with '+' is the path planned by A* and it is the closest to the obstacle, and the path shown with 'o', 'x' and '*' is generated by EA* corresponding to the obstacle expansion of 1, 2 and 3 grids respectively. The color definition is all the same in the following. From this figure we can see that the distance between the robot and obstacle can be effectively increased by EA*.

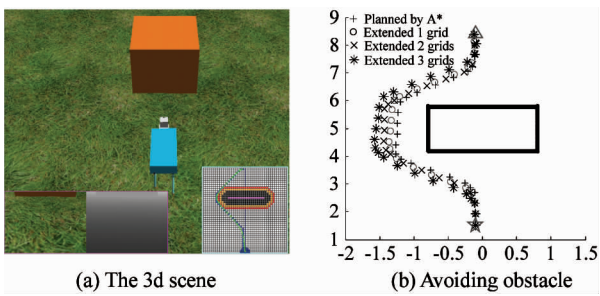


Fig. 19 Avoiding a single obstacle

In addition, the ability of passing through the narrow passage using EA* is tested too. Fig. 20(a) shows the scene, and the passage width is 1.5m. From the Fig. 20(b), it can be seen that the passage will not be blocked by extending the obstacle through EA*.

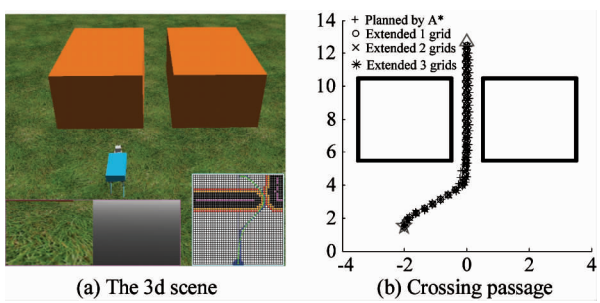


Fig. 20 Passing through the narrow passage

The Fig. 21 shows the route planned under more complex environment and the route reaction to obstacles is different from the increase of extended grids especially when the obstacle is extended with three grids. Through many simulations, it can be found that extending the obstacle with two or more grids can avoid collision effectively.

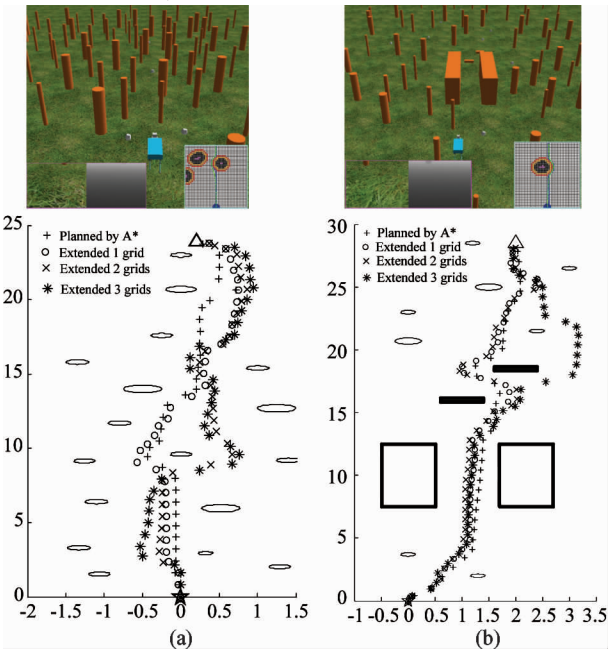


Fig. 21 Path comparison under complex environment

5 Conclusions

A quadruped robot path planning method based on sliding window and variant A* in unknown environment is presented. In order to improve the path re-planning efficiency and the security of the robot, IA* and EA* are proposed.

IA* adds the new path connecting the current route to the new goal projection to generate a suboptimum route incrementally. Compared with using A* to solve each path planning problem from scratch, IA* increases the search efficiency more than 5 times. EA* extends the obstacle by means of increasing the g-value of the grid, which makes the route far away from the obstacle meanwhile avoids blocking the narrow passage.

Though IA* and EA* have some advantages compared with A*, IA* will change the course more frequent and the amount of computations will be increased by EA*. In the next, the path planning method needs to be further perfected and some experiments will be done on the quadruped robot.

References

[1] Raibert M, Blankespoor K, Nelson G, et al. BigDog, the rough-terrain quadruped robot. In: Proceedings of IEEE International Conference on Robotics and Automation, Alaska, USA, 2008. 4736-4741
[2] Michal P, David M B, Jonathan K C, et al. Leader Tracking for a Walking Logistics Robot. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. Hamburg, Germany, 2015. 2994-3001
[3] Alexander W W, Carlos M, Ioannis H, et al. Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain. In: Proceedings of IEEE International Conference on Robotics and Automation, Washington, USA, 2015, 5148-5154
[4] Chai H, Meng J, Rong X W, et al. Design and implementation of SCalf, an advanced hydraulic quadruped robot. *Robot*, 2014, 36(4): 385-391
[5] Arain M A, Havoutis I, Semini C, et al. A comparison of search-based planners for a legged robot. In: Proceedings of the 9th International Workshop on Robot Motion and Control, Wasowo Palace, Wasowo, Poland, 2013. 104-109
[6] Wooden D. Graph-based Path Planning for Mobile Robots, Dissertation, Thesis. Georgia Institute of Technology, 2006. 102-150
[7] Hart P E, Nilsson N J, Raphael B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*. 1968, 4(2): 100-107
[8] Likhachev M, Ferguson D, Gordon G, et al. Anytime search in dynamic graphs. *Artificial Intelligence*, 2008, 172(14):1613-1643
[9] Stentz A. Optimal and efficient path planning for partially-known environments. In: Proceedings of IEEE International Conference on Robotics and Automation, 1994. 3310-3317
[10] Stentz A. The focussed D algorithm for real-time replanning. In: Proceedings of the International Joint Conference on Artificial Intelligence, 1995. 1652-1659
[11] Koenig S, Likhachev M. Fast Replanning for Navigation in Unknown Terrain. *IEEE Transactions on Robotics*, 2005, 21(3): 354-363
[12] Koenig S, Likhachev M, Furcy D. Lifelong Planning A*. *Artificial Intelligence Journal*, 2003, 155(1-2): 93-146
[13] Likhachev M, Gordon G, Thrun S. ARA*: Anytime A* with provable bounds on sub-optimality. In: Advances in Neural Information: Processing Systems, MIT Press, 2003. 52-89
[14] Wooden D, Malchano M, Blankespoor K, et al. Autonomous Navigation for BigDog. In: Proceedings of IEEE International conference on Robotics and Automation, Alaska, USA, 2010. 4736-4741
[15] Boor C D. A Practical Guide to Splines. Springer-Verlag, 1978
[16] Dierckx P. Curve and Surface Fitting with Splines. New York: Clarendon Press, 1995. 63-80
[17] Piazzi A, Visioli A. Global minimum-jerk trajectory planning of robot manipulators. *IEEE Transactions on Industrial Electronics*, 2000, 47(1): 140-149
[18] Song G, Amato N. Randomized motion planning for car-like robots with c-prm *. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Hawaii, USA, 2001. 37-42

Zhang Hui, He is now a doctoral student in the school of control science of engineering, Shandong University, China. He received his bachelor and master degrees from Shandong Agricultural University, China, in 2009 and 2012 respectively. He research interests include environmental perception and path planning for the quadruped robot.