J Intell Robot Syst (2006) 47: 197–214
DOI 10.1007/s10846-006-9071-3

Path Planning for a Statically Stable Biped Robot Using PRM and Reinforcement Learning

Prasad Kulkarni · Dip Goswami · Prithwijit Guha · Ashish Dutta

Received: 7 June 2006 / Accepted: 31 July 2006 /
Published online: 11 October 2006
© Springer Science + Business Media B.V. 2006

Abstract In this paper path planning and obstacle avoidance for a statically stable biped robot using PRM and reinforcement learning is discussed. The main objective of the paper is to compare these two methods of path planning for applications involving a biped robot. The statically stable biped robot under consideration is a 4-degree of freedom walking robot that can follow any given trajectory on flat ground and has a fixed step length of 200 mm. It is proved that the path generated by the first method produces the shortest smooth path but it also increases the computational burden on the controller, as the robot has to turn at almost all steps. However the second method produces paths that are composed of straight-line segments and hence requires less computation for trajectory following. Experiments were also conducted to prove the effectiveness of the reinforcement learning based path planning method.

Key words potential function · PRM · reinforcement learning · statically stable biped robot

P. Kulkarni
Tata Motors Research Division, Tata Motors, Pune, India

D. Goswami
Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576, Singapore

P. Guha
Department of Electrical Engineering, IIT Kanpur, Kanpur 208 016, India

A. Dutta
Department of Mechanical Engineering, IIT Kanpur, Kanpur 208 016, India

A. Dutta (✉)
Department of Mechanical Science and Engineering, Nagoya University, Chikusa-Ku, Furo-cho, Nagoya, Japan
e-mail: adutta@iitk.ac.in

1 Introduction

In the last three decades several methods for path planning and obstacle avoidance have been developed which were mainly for mobile robots. However with the recent development of biped humanoid robots there has emerged a new need to find path-planning methods that are suitable for biped locomotion. The basic difference between path planning for biped robots and mobile robots is that for mobile robots the path is continuous, while for biped robots it is not. Humanoid robots can be broadly classified into statically stable robots and dynamically stale robots. A dynamically stable robot has ten or more degrees of freedom which gives it greater freedom in placing its foot at a desired point. In case of a statically stable biped robot since the degrees of freedom is limited to two or four, path planning is more difficult than that of a dynamic biped robot. The ability to follow a particular trajectory also depends on the design of the legged robot. Due to this, a path planning method that may work for one type of a robot may not work efficiently for all other types of biped robots.

The main objective of this paper is to compare two methods of path planning, the probabilistic road map (PRM) and the reinforcement learning method, for obstacle avoidance applications involving a statically stable robot. These two methods were chosen to illustrate that the path obtained by the first method is very smooth and the shortest, however the biped has to expend a lot of energy to follow this path. This is because the robot will have to compute every step in such a way that its center of gravity falls on or very near the desired trajectory. Several times if the robot cannot place its foot on the trajectory it will get caught in an infinite loop, that will cause a very high burden on the robot controller. On the other hand, the reinforcement leaning method generates paths consisting of straight lines that will be simpler for the robot to follow. The two methods were compared by measuring the energy expended to follow each path from the initial point to the final goal point, avoiding obstacles.

Several algorithms are available for the general problem of robot motion planning Latombe [11] and Wuril [17]. The PRM method has been extensively used for path planning and normally gives good results for mobile robots. Kavraki et al. [7] have developed methods for path planning in high dimensional configuration space. Clark et al. [4] have proposed a method for motion planning for multiple robotic systems using dynamic networks. Barraquand et al. [1] have studied a random sampling scheme for path planning. After a path has been found using the road map method the potential field method is used for smoothening the path Barraquand et al. [2]. The problem of motion planning of non-holonomic biped miniature climbing robots have been addressed by Dulimarta and Tummala [5]. Kuffner et al. [8, 9] have developed a method of efficient path planning and obstacle avoidance for biped humanoid robots based on optimizing a cost function which determines the best position to place the robots foot. Sensor based obstacle avoidance techniques have been developed by Yagi and Lumelsky [18]. Recently online techniques have been developed for generating trajectories for biped walk, based on footstep placement [14]. Some work in virtual reality addressed the problem of foot placement of virtual characters, and developed motion planners [10]. Research dealing with path generation and obstacle avoidance for statically stable biped robots is not available in the robotics literature. It is important to study path generation and obstacle avoidance for these types of robots, as statically sable robots are very simple to build and control as compared to dynamic robots. Hence very simple biped robots can be used to carry out the requirement of locomotion and obstacle avoidance.

Learning based path planning methods have also been used for path planning. When complete knowledge of the domain is available, the global path planning technique can be applied [15]. However, efficiency of such techniques decreases rapidly in more complex domains since considerable modeling is needed. Acquiring the state-action pairs is especially interesting using approaches where learning capabilities apply, such as reinforcement learning, fuzzy logic and neural networks. Learning the fuzzy logic rule base or providing the necessary target patterns in the supervised neural network learning may be very tedious and difficult [3, 12, 13, 16].

In Section 2 of this paper the biped robot and the involved kinematics, used for trajectory following is explained. In Section 3 path planning for avoiding obstacles using the combination of the roadmap method and potential method is described. In Section 4 the method of reinforcement learning used for path generation is explained, while simulation of the two methods and their comparison is given in Section 5. The experiments and results are given in Section 6, and the conclusions are stated in Section 7.

2 Biped Robot Details

The designed statically stable robot used in the experiments has large feet which overlap so that the center of gravity is always inside the foot perimeter even when the robot is on a single foot stance. The biped (Figure 1) robot is actuated at the hip joint by two actuators, which are connected to the two legs by two parallel four bar mechanisms. The legs, actuated by motors at the hips, move in circular fashion in order to advance to the next step. The turning

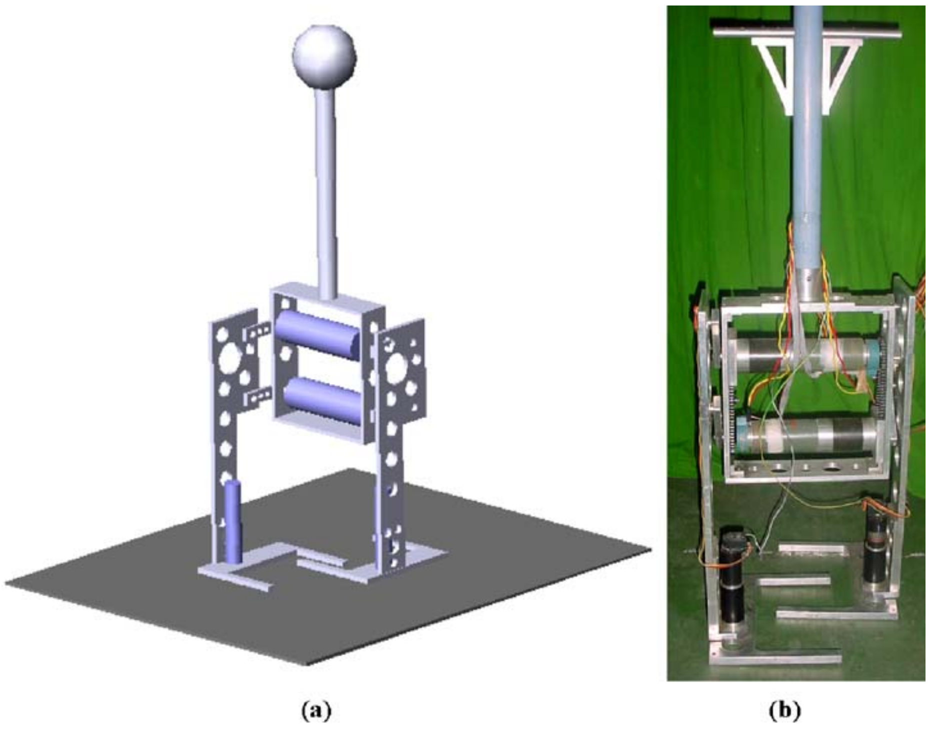


Figure 1 The statically stable biped robot (a) CAD model and (b) fabricated robot.

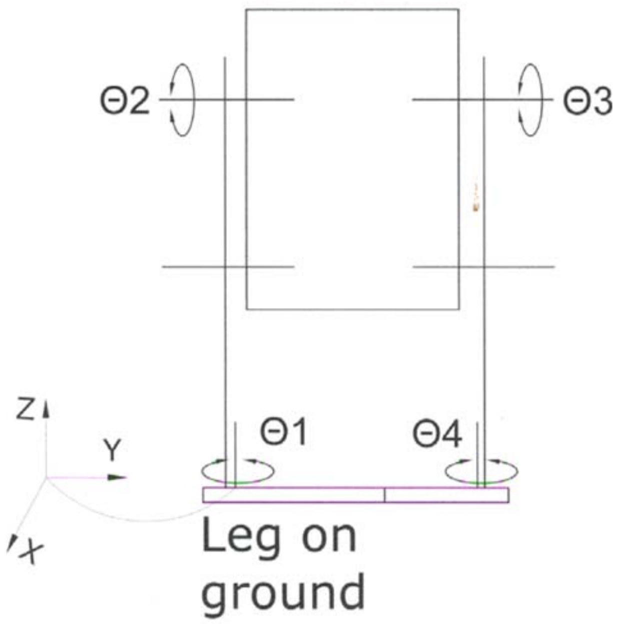


Figure 2 The various angles of the robot.

motion of the robot is provided by the motors at the ankle. When the robot is on a single foot stance the ankle motor of the foot on the ground can rotate, as a result the whole biped takes a turn. Therefore the ankle motor of the foot on the ground will mainly contribute for the rotation of the biped whereas the rotation of the other ankle motor will only make the aerial foot parallel to the trajectory. In order to follow a trajectory it is required to rotate both the ankle motors.

2.1 Trajectory Generation

The inverse kinematics defines the transformation from the work space position vector to the joint space position vector. As the two feet have a common intersecting area the legs of the robot can interfere with each other during gait, if the step length is not properly chosen. Hence for each step the step length has been fixed at 200 mm (180° rotations for both motors θ2 and θ3) as shown in Figure 2. The procedure for following a trajectory is to position the center of gravity of the biped on the desired trajectory as it moves. Due to the physical constraints, the robot can turn left by 10° or right by 35° in any one step, without foot interference. Figure 3 shows the two constraints on the angle for turning.

The direct kinematics defines the transformation from the joint space position vector to the cartesian space position vector. The joint space position vector consists of four angles (θ_1, θ_2, θ_3, θ_4) as shown in Figure 2. The two angles θ_2 and θ_3 rotates simultaneously to advance the leg to the next step. When leg one is on the ground the motor θ1 can rotate to turn the torso about the axis of the motor. The procedure for following a trajectory, as shown in Figure 4, is such that first the two hip motors rotate to move the leg forward and then the ankle motor rotates to place the CG of the robot on or near the trajectory. If the coordinates of the motor axis of the leg on the ground is given by A (X_0, Y_0, Z_0) and the coordinates of the motor axis of the leg that will move given by B (X_1, Y_1, Z_1) then the

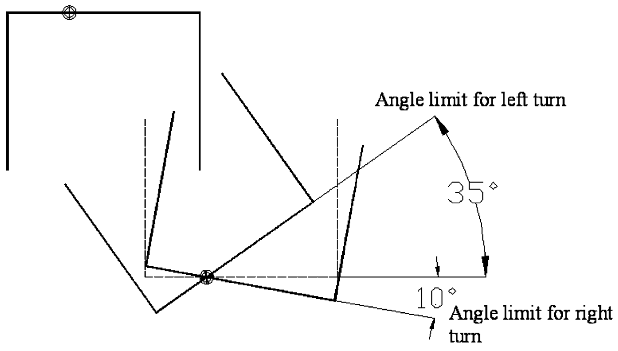


Figure 3 Limits of angles of ankle motor. 5 Rotation of foot in air about M2.

relation between the two is:

$$\begin{aligned} X_1 &= X_0 - 170 \\ Y_1 &= Y_0 + 30 \\ Z_1 &= Z_0 \end{aligned} \tag{1}$$

As the hip motors rotate the point (B) of the leg on the ground is transformed to the point C (X_2, Y_2, Z_2). It is assumed that the motors rotate simultaneously and in the same direction as the leg movers forward. The relation between the coordinate of point C and B is given by:

$$\begin{aligned} X_2 &= X_1 - 100\mathrm{Cos}\theta_2 - 100\mathrm{Cos}\theta_3 \\ Y_2 &= Y_1 \\ Z_2 &= Z_1 + 100\mathrm{Sin}\theta_2 + 100\mathrm{Sin}\theta_3 \end{aligned} \tag{2}$$

Where θ_2 and θ_3 are the angular rotation of the hip motors measured in the clockwise direction. In order to place the CG on the designed trajectory the ankle motor at “A”, rotates by an angle (θ_1) and the new coordinate of D (X_3, Y_3, Z_3) is given by:

$$\begin{aligned} X_3 &= \sqrt{{X_2}^2 + {Y_2}^2}\,[\mathrm{Cos}(\psi - \theta_1)] \\ Y_3 &= \sqrt{{X_2}^2 + {Y_2}^2}\,[\mathrm{Sin}(\psi - \theta_1)] \\ Z_3 &= Z_2 \end{aligned} \tag{3}$$

Where

$$\psi = \tan^{-1}\left(\frac{X_3 - X_0}{Y_3 - Y_0}\right)$$

The only free variable is θ_1 for positioning the point ‘D’ so that the CG is on or near the trajectory. As it is not always guaranteed that the point ‘D’ can be placed on the trajectory an allowable band of ±10 mm has been provided within which the CG may be placed around the desired trajectory. This however introduces an error between the actual path followed and the desired path and could result in hitting the obstacle. An example showing how the object might be hit is given later.

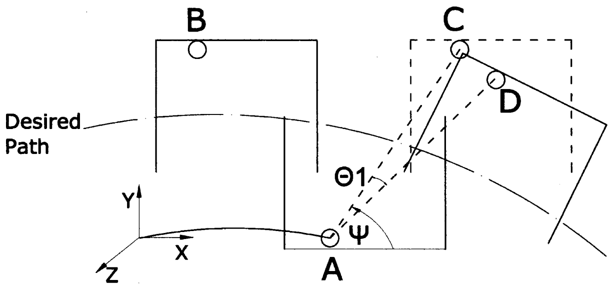


Figure 4 Foot placement for following a trajectory.

Cases of trajectories having high curvature can be followed by back stepping where several front and back half steps are taken, in order to take a large turn standing at the same point. Therefore in the method given above, if the calculated angle required for turning is greater than the limits then back stepping is used. Figure 5 shows how the foot is placed for traversing a trajectory. The energy consumed by the robot will be proportional to the total angular displacement of all the motors. If the robot moves straight then only the hip motors will be rotating. However if the robot has to follow a curved path then all the four motors will have to operate, and the controller would have to calculate the angle of turn at every step. This will also increase the computational burden on the controller and may result in the control algorithm from getting stuck in cases of sharp bends or impossible trajectories.

3 Path Planning Using Road Map and Potential Method

In the first method of path planning a combination of the roadmap method and potential field method is used. The obstacles are enlarged through an amount equal to the dimensions of the biped robot. In the roadmap method, a standardized path network, called roadmap is generated in C-free space (subspace of the configuration space) having no obstacles. Then from the initial point to the goal point a path is found in this network only. Out of several

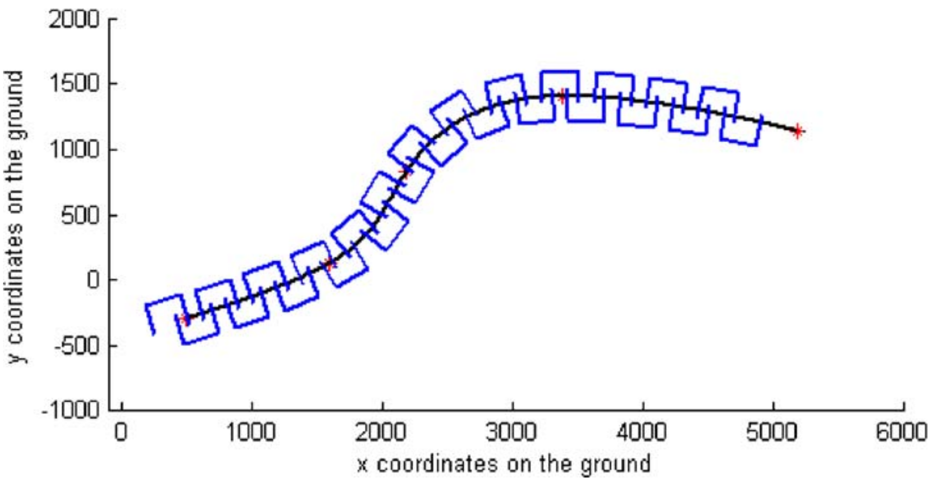


Figure 5 Position of the feet for following the trajectory.

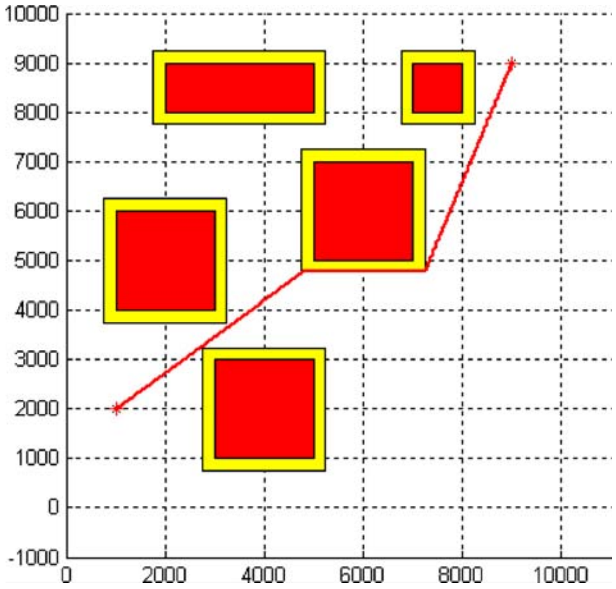


Figure 6 Path from roadmap method.

roadmap methods, the 'visibility graph method' has been used for path planning of the statically stable biped robot. In this approach, the network is generated such that the nodes of the network will be the initial point, goal point and the nodes of the obstacles. All the links of the roadmap are straight lines connecting two nodes such that the complete line lies in the C-free region, i.e., the line does not intersect the interior of the obstacles. Figure 6 shows the roadmap for a workspace of five obstacles. The path generated by the roadmap method is not smooth and it also touches the obstacle. Therefore to increase the distance of the path from the obstacle and to smooth the path potential method is used.

In the potential field method of path planning the robot is considered as a particle starting from the initial position, moving under the influence of several artificial potentials produced by the obstacles and the goal point. The obstacle will produce a repulsive potential whereas the goal point will produce an attractive potential. At every point the direction of the resultant force (due to the attractive and repulsive forces) will be in the direction of motion. By considering all these points, the following potential function is used:

$$F = -\frac{A}{r^2} d_1 d_2 \tag{4}$$

Where d_1 and d_2 are the distances of any point on the path from starting and end point respectively and r is the minimum distance of the point from the obstacle. After calculating the forces on each point, the points are moved in the direction of the force acting on that point. Figure 7 shows the results of the smoothening of the path by potential method.

After getting a path avoiding all the obstacles, using the trajectory following algorithm, the required rotation angles of the ankle to follow this path is calculated. The results of the path planning and trajectory following is shown in Figure 8.

The path planning algorithm was as follows:

a) Capture the workspace from the top using a vision camera, and take the initial and final configuration of the robot.

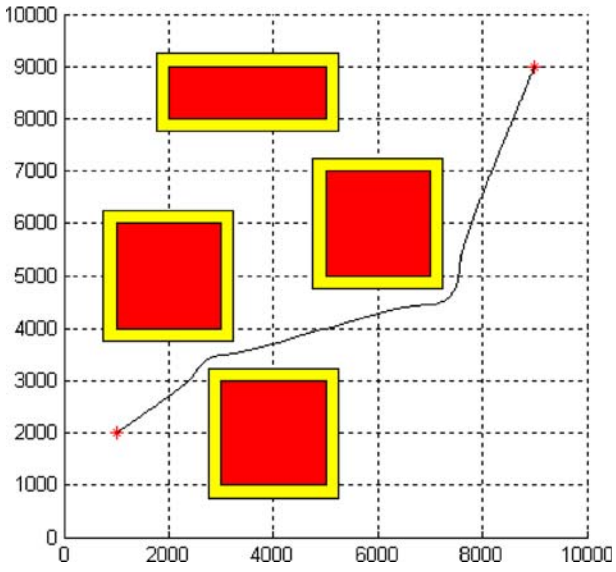


Figure 7 Path after potential method.

b) Using vision processing, the obstacles are detected and then the C-obstacles and the C-space is calculated.
c) Make a roadmap "R" for given initial point, final point and the nodes of C-obstacles.
d) Find a path of minimum length form the initial to the goal point in "R".
e) Apply an artificial negative potential to the obstacles and an attractive potential to the goal point.
f) Smooth the path generated by the roadmap method by potential field method.

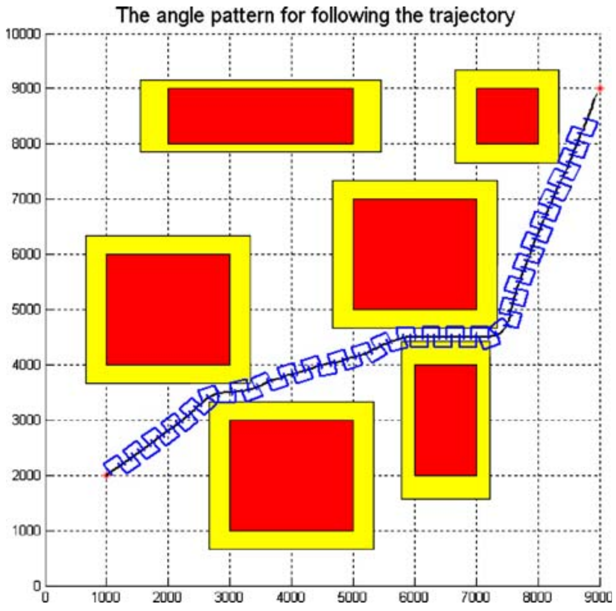


Figure 8 Trajectory following for the path avoiding the obstacles.

This method of path planning works very well for mobile robots, as the paths obtained are very smooth and suitable for mobile robots. However, smooth paths need not necessarily be energy efficient. This would also be a great burden on the controller as at almost every step the controller would have to calculate the exact angle to turn the body to place its trajectory on the path. In case the path has very sharp turns then the robot controller may go into an infinite loop as there may be no way of placing the foot on the trajectory. In the case that the path produced by the road map method is used (Figure 8), then there is a danger that the robot may hit any of the obstacles due to errors in the controller or flexibility at joints, as it tends to move into the C-Space around the obstacle.

4 Reinforcement Learning Using Relaxed-SMART Algorithm

Reinforcement learning (RL) is generally used to solve Markov decision problem (MDP) and semi Markov decision problem (SMDP). The framework of MDP has the following element [6]: (a) state of the system, (b) actions, (c) transition probabilities, (d) transition rewards, (e) a policy, and (f) a performance metric. The states of a system is a set of parameters that can be used to describe a system. The transition from one state to another in an MDP is a random affair. The actions can be defined as all the possible transitions from a particular state. Assume the robot moves in discrete steps and that after every step the robot takes, the robot can 'go north','go south', 'go east', or 'go west'. The transition probability is the probability of going into a particular state from the current state for a particular action. Assume that action a is selected in state i. Let the next state be j. Usually, the system receives an immediate reward (which could be positive or negative) when it transitions from one state to another. A policy is a complete mapping from states to actions, associated with every policy there is a performance metric with which the performance of the policy is judged. The average reward performance metric has been considered. Let the time spent in transition from state i to state j under the influence of action a is denoted by $t(i, a, j)$. For SMDP, $t(i, a, j)$ will vary with i,j, in case of MDP $t(i, a, j)=1$. The basic idea in RL is to store a so-called Q-factor for each state-action pair in the system. Thus, $Q(i, a)$ will denote the Q-factor for state i and action a. The values of these Q-factors are initialized to arbitrary numbers in the beginning. Then the system is simulated (or controlled in real time) using the algorithm. In each state visited, some action is selected and the system is allowed to transition to the next state. The immediate reward (and the transition time) that is generated in the transition is recorded as the feedback. The feedback is used to update the Q-factor for the action selected in the previous state. Roughly speaking if the feedback is good, the Q-factor of that particular action and the state in which the action was selected is increased (rewarded) using the relaxed-SMART algorithm. If the feedback is poor, the Q-factor is punished by reducing its value.

4.1 Obstacle Avoidance by RL for Statically Stable Biped

The first thing for using RL for obstacle avoidance of a statically stable biped is to represent the entire domain of motion of the biped in a large number of distinguishable points. These are considered to be the different states of the MDP. The larger the number of states, the longer the processing time but the smoothness of the path will increase. Here we have to make a compromise between the number of states and the smoothness of the path. Relaxed-SMART algorithm has two stages: learning phase and frozen phase. In learning phase, first

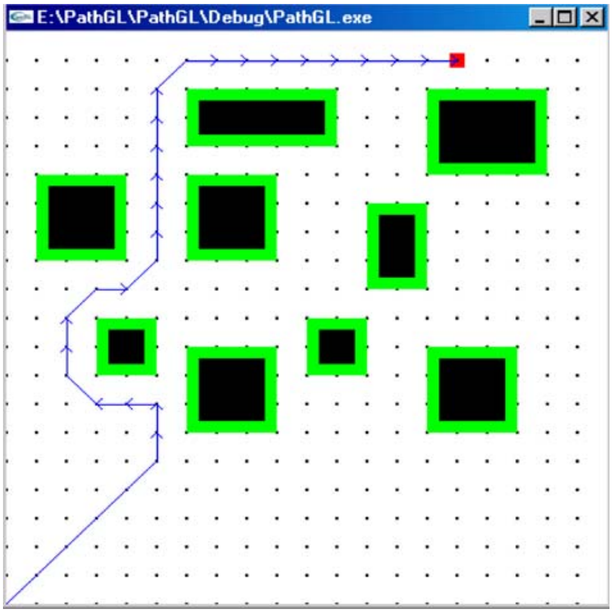


Figure 9 Path finding with nine obstacles.

an arbitrary value of Q-factor at each state-action pair is assigned. Let $A(i)$ denote the set of actions allowed in state i and α^k denote the main learning rate in the kth iteration. Set α^0 to some arbitrary value. Let β^k denote the secondary learning rate and set β^0 to some value smaller than α^0. For the biped there are eight actions possible in each state because from any state in the domain of motion it can move in eight directions. Let ρ^k denote the average reward in the kth iteration of the algorithm, set ρ^0 to 0. Initially we set total reward for all the states zero and total time of transition is also set to zero. Let ϕ denote the greediness of the system and set $\phi=1$ for each state. Both α^k and β^{wk} are positive, decreasing functions of k. Any action a in state i has probability $1/|A(i)|$. Any action a is randomly selected out of the eight possible actions. If a denotes the action associated with the maximum Q-factor of state i, set $\phi=0$ (this means that the action selected is greedy with respect to the Q-factor.). Otherwise, set $\phi=1$. Simulate action a i..e. go a step ahead in the specific direction of that action. Let the next state be denoted by j. Let $r(i; a; j)$ denote the transition reward and $t(i; a; j)$ denote the transition time which is equal to unity in our case. Then update $Q(i; a)$ as follows:

$$Q(i,a) = \left(1-\alpha^k\right)Q(i,a) + \alpha^k\left[r(i,a,j) - \rho^k t(i,a,j) + Q_{max}\right] \tag{5}$$

where $Q_{\max}$ denotes the highest Q-factor (Q-factor with the maximum value) in state j.

If $\phi=0$, update total reward and total time as follows.

$$\text{total reward} = \text{total reward} + r(i,a,j)$$

$$\text{total time} = \text{total time} + t(i,a,j)$$

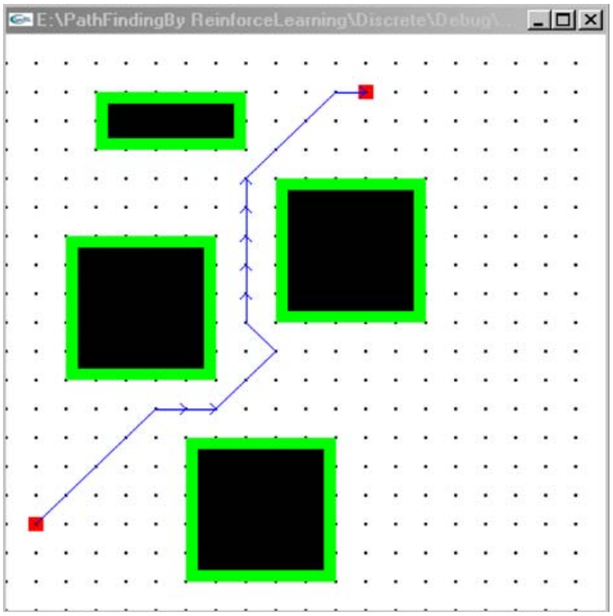


Figure 10 Path finding with four obstacles.

Then update the average reward as:

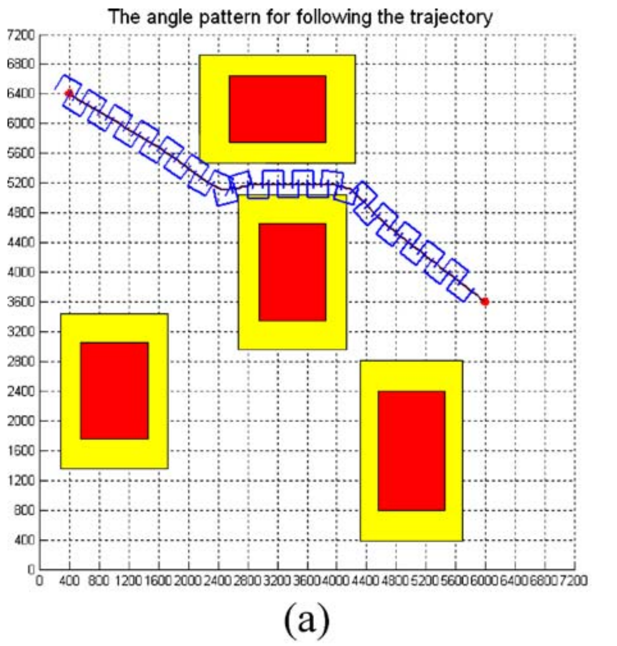
$$\rho^k + 1 = \left(1 - \beta^k\right)\rho^k + \beta^k[\text{total reward}/\text{total time}] \tag{6}$$

If ϕ=1, go to next iteration if destination is not reached. If destination is reached, stop and start the entire process from the beginning. This process is continued for a large number of times. Then for each state the action for which *Q*-factor is maximum is declared as the optimal action. In the frozen phase, for state i, the action which has the maximum *Q*-factor is selected. In reinforcement learning parameters : initial *Q*-value for all the state-action pairs has been taken zero i.e., $Q(i, a)$=0 and $r(i; a; j)$=1 and $t(i; a; j)$=1. Using this procedure simulations were carried out for path generation and obstacle avoidance using reinforcement learning.

The value of the reinforcement learning parameters is problem-specific. The rewards $r(i; a; j)$ and transition times $t(i; a; j)$ are dependent on the application domain. In our case, they are taken to be unity. The value of the learning rate α^{k} and β^{k} is also problem-specific. If the learning rate is too high (close to unity) then the learning at that particular state-transition will be very good at the cost of forgetting the learning at the previous transitions. So this makes the learning process unstable. If the learning rate is very low then the learning process will be very slow. Therefore, a trade-off between these two aspects has be to made during choosing learning parameters α^{k} and β^{k}.

In the simulation the top-view of the entire domain of motion has been divided into 400 points and nine obstacles of different size have been taken. We have initialized ρ^0=0, α^0= 0.95, β^0=0.90 and taken maximum iterations = 1,000. The black dots indicate the states of the domain of motion, and the distance between each dot is equal to two foot steps of the robot (400 mm). The initial position is the left corner and the goal position is the top right position.

The simulation took 3 min 50 s to find the path. It can vary with positioning of the obstacle.



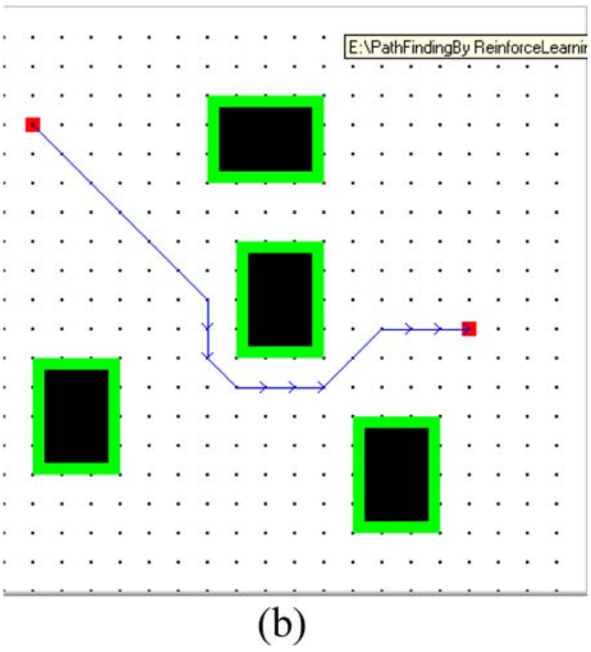


Figure 11 Case I: four obstacles (a) path generated by road map and potential method, (b) path generated by reinforcement learning method.

The choice of total number of iterations is also dependent on the positioning of the obstacle. Up to nine obstacles 1,000 iteration gives good result. Time of simulation is also dependent on the positioning of the source and destination points. Two cases of path planning and obstacle avoidance is shown in Figures 9 and 10.

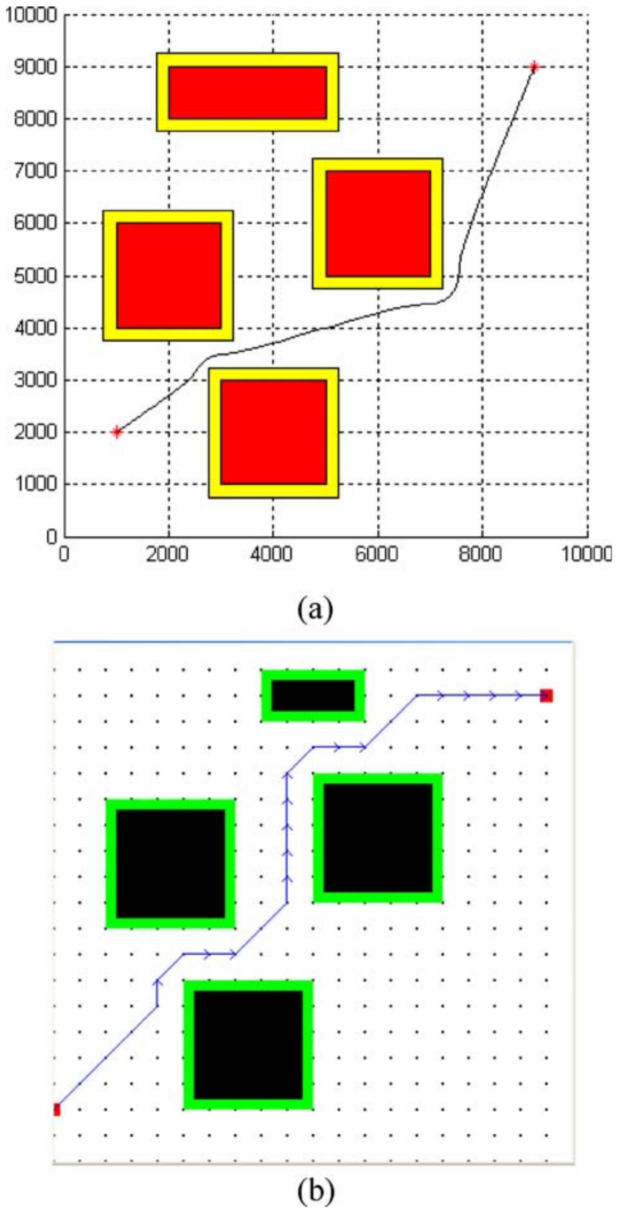
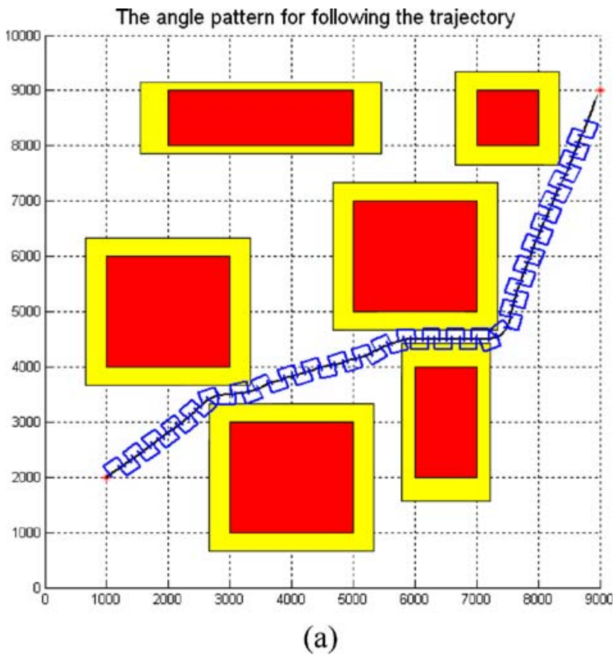


Figure 12 Case II: four obstacles (a) path generated by road map and potential method, (b) path generated by reinforcement learning method.

5 Comparison of the Two Methods

The simulation results for the two methods of path planning were compared for obstacle avoidance and path planning for the same case with different number of obstacles. In Case I there were four obstacles (Figure 11), in Case II there were four obstacles but different goal points (Figure 12), in Case III there were six obstacles (Figure 13) while Case IV shows a case where reinforcement learning fails (Figure 14).



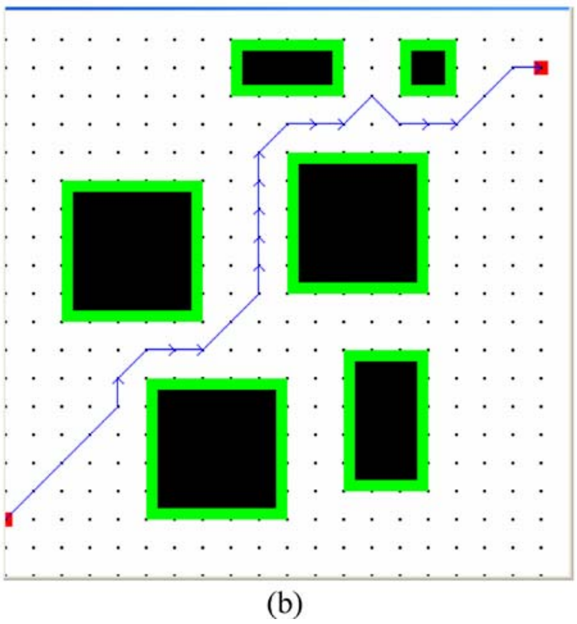


Figure 13 Case III: six obstacles (a) path generated by road map and potential method, (b) path generated by reinforcement learning method.

The comparison between the two methods was made based on the total number of steps required to move the robot from the initial point to the goal point, and the total angles turned by the motors at the ankle for following the path. The results are given below in Table I.

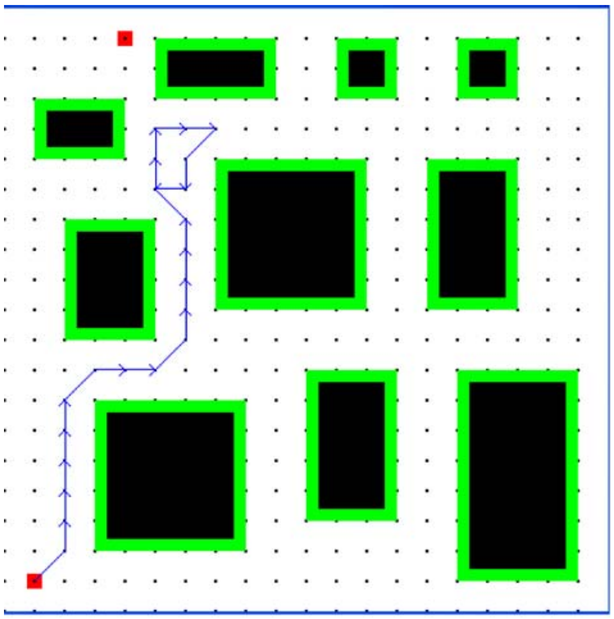


Figure 14 Case IV: reinforcement learning fails in this case with nine obstacles.

The above results show that the paths generated by the road map method were shorter than the path generated by the reinforcement learning method. This is because the path is generated by connecting the initial and final goal point by straight line segments that are then smoothened by the potential method in order to avoid obstacles. As it is generated by straight line segments it gives the shortest distance between two points. However as the path is smoothened it does not remain exactly a straight line and hence the robot has to compute the angle required to turn for every step, to follow the trajectory. This greatly increases the computational burden on the motion controller. This problem of increase in computational load on the controller is unique to walking robots as path planning methods were basically invented for wheeled robots. Also it is clear from the simulation that in case of PRM based path, the robot could enter the C-space around the obstacle, due to errors in path following and hence the risk of hitting the obstacle increases. The path finding by reinforcement learning is a random yet systematic way of obstacle avoidance. It finds the path which is simpler to follow but it does not guarantee minimum energy consumption unlike road map method. In the case of the reinforcement learning method the robot can only move along the side or the diagonal of each discrete square element. This is because

Table I Comparison of reinforcement learning and roadmap path planning

Case no.	Method used	Total number of steps	Total angle turned (degrees)
Case I	Reinforcement learning	41	225
	Road map method	31	167
Case II	Reinforcement learning	58	405
	Road map method	44	145
Case III	Reinforcement learning	59	540
	Road map method	52	185

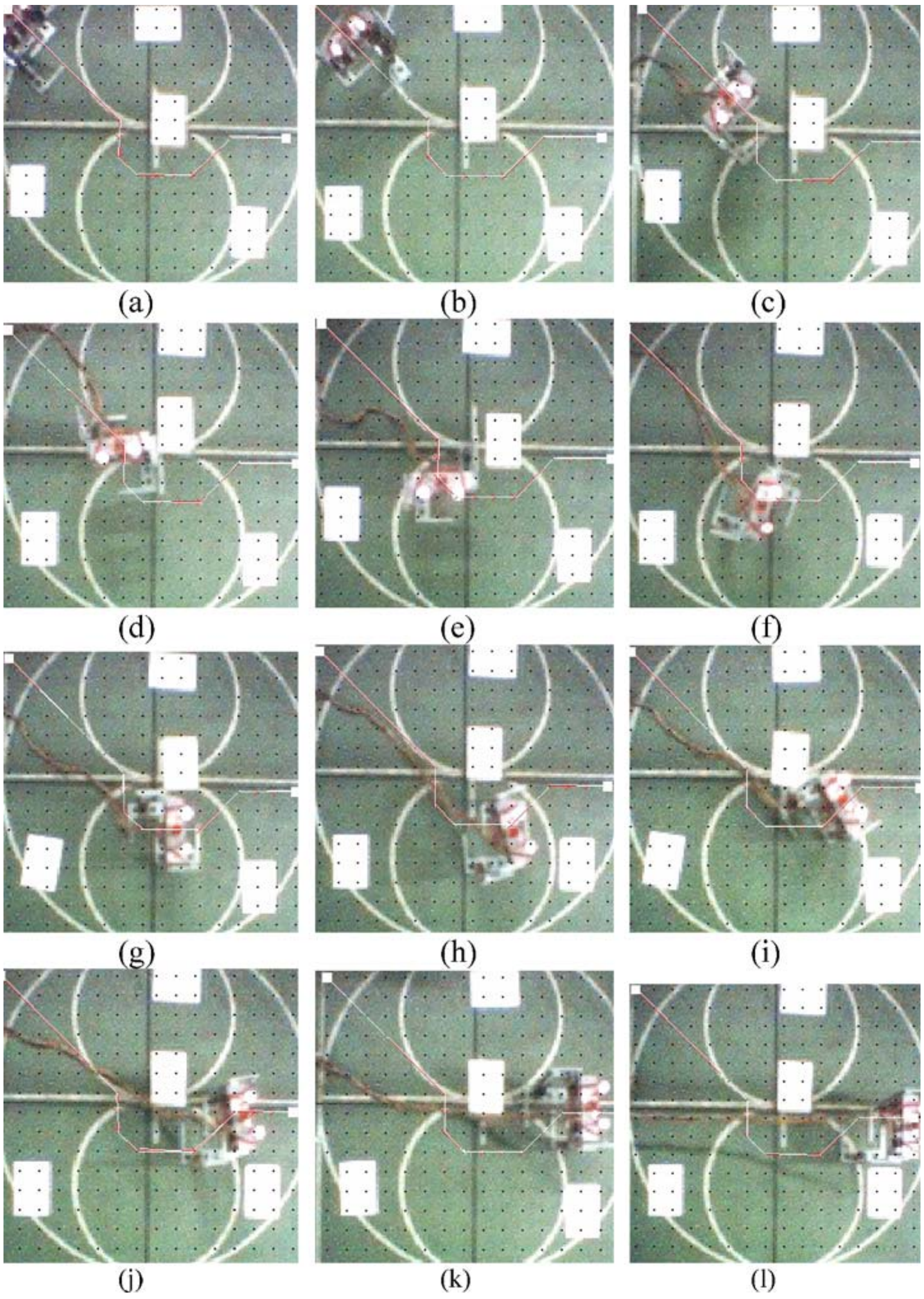


Figure 15 The robot following the trajectory generated for avoiding four obstacles.

the robot has to choose between eight states and hence this result in a path that is longer but the computational burden on the robot controller is less, as it has to mainly walk straight and turn at a few points. In the last Case IV it has been shown that if the path between two obstacles is extremely narrow then the reinforcement learning algorithm fails, as the robot keeps looping. However the road map method still gives a path as the robot can move between two points in infinite number of ways. Hence it can navigate very narrow passages or sharp corners. In all the simulation cases the paths generated by both the methods were different because in the PRM method the robot can choose to move in any direction between two points, however in the RL method the robot has to choose one path in between eight possible paths (states) to move between two points.

6 Experimental Results

The path generated by the reinforcement learning method as shown in the simulation Figure 11 was experimentally evaluated. Four obstacles (white boxes) were placed in a flat space and the initial point and goal point were marked. An overhead camera took images of the surface and the obstacles were detected using image processing. The robot controller then calculated the path from the initial point to the goal point. A thin white line marked the desired trajectory. An overhead camera was used to capture the image of the robot as it followed the trajectory. The two ends of the hip of the robot were marked by two white lights and are visible in the Figure 15. The robot was initially placed at the initial point and it followed the trajectory to reach the goal point. The results of the trajectory following experiment are as shown in Figure 15 by a series of photographs.

7 Conclusion

This paper discusses aspects of path planning using PRM and reinforcement learning for a statically stable biped robot. Motion of biped robots is discrete and hence path-planning methods that are designed for mobile robot are not always suitable for biped foot placement. It was shown that PRM will give the shortest path but it also increases the risk of collusion with obstacles due to errors in footstep placement and makes trajectory following more difficult. Reinforcement learning gives a longer path but it is simpler to follow, hence reducing the burden on the controller. The importance of path planning for statically stable robots is that such robots are not only simple to build and control, but they can also be used for locomotion.

References

1. Barraquand, J., Kavraki, L.E., Latombe, J.C., Raghavan, P.: A random sampling scheme for path planning. Int. J. Rob. Autom. **16**(6), 759–774 (1997)
2. Barraquand, J., Kavraki, L.E., Latombe, J.C.: Numerical potential field techniques for robot path planning. IEEE Trans. Syst. Man Cybern. **22**(2), 224–241 (1992)
3. Beom, H., Cho, H.: A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. IEEE Trans. Syst. Man Cybern., **SMC-25**(3), 464–477 (1995)
4. Clark, M., Rock, S.M., Latombe, J.C.: Motion planning for multiple robot systems using dynamic networks. Proceedings of the IEEE International Conference on Robotics and Automation (2003)
5. Dulimarta, H., Tummala, R.L.: Design and control of miniature climbing robots with non-holonomic constraints. Proceedings of the World Congress on Intelligent Control and Automation, Shanghai, pp. 3267–3271 (2001)
6. Gosavi, A.: A Tutorial for Reinforcement Learning. The State University of New York at Buffalo
7. Kavraki, L.E., Svestka, P., Latombe, J.C., Overman, M.: Probabilistic roadmap for path planning in high dimensional configuration space. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
8. Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M.: Footstep planning among obstacles for biped robots. Proceedings of IROS (2001)

9. Kuffner, J.J., Kagami, S., Inaba, M., Inoue, H.: Dynamically stable motion planning for humanoid robots. Auton. Robots. **12**(1), 105–118 (2002)
10. Kalisiak, M., VanDe Panne, M.: A grasp-based motion planning algorithm for character animation. J. Viz. Comput. Animat. **12**(3), 117–129 (2001)
11. Latombe, J.C.: Robot Motion Planning. Kluwer, Academic (1991).
12. Lee, C.T., Lee, C.S.G.: Reinforcement structural parameter learning for neural-network based fuzzy logic control system. IEEE Trans. Fuzzy Syst. **2**(1), 46–63 (1994)
13. Macek, K., Petrovic, I., Peric, N.: A reinforcement learning approach to obstacle avoidance of mobile robots Advanced Motion Control, 2002. 7th International Workshop on 3–5 July 2002, pp. 462–466
14. Nishikawa, K., Sugihara, T., Kagami, S., Inaba, M., Inoue, H.: Online mixture and connection of basic motions for humanoid walking control by footprint specifications. Proceedings of the International Conference on Robotics and Automation (2001)
15. Schwartz, J.T., Shirir, M.: A survey of motion planning and related geometric algorithm. Artificial Intelligence Journal **37**, 157–169 (1988)
16. Sehad, S., Touzet, C.: Self-organizing Map For Reinforcement Learning: Obstacle-avoidance With Khepera From Perception to Action Conference, 1994. Proceedings, pp. 420–424 September 7–9 (1994)
17. Wuril, C.: Multiple robot path coordination using artificial potential fields. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 500–505 (1990)
18. Yagi, M., Lumelsky, V.: Biped robot locomotion within scenes with unknown obstacles. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 375–380 (1999)