INŽENÝRSKÁ MECHANIKA 2005
NÁRODNÍ KONFERENCE
s mezinárodní účastí
Svratka, Česká republika, 9. - 12. května 2005

PATH PLANNING FOR FOUR-LEGGED WALKING ROBOT USING RAPIDLY EXPLORING RANDOM TREES

J. Krejsa[*], S. Věchet[+]

Summary: *There are several randomized methods for problem of path planning. Rapidly exploring random trees (RRT) is a method which can deal with constraints typical for legged walking robots, e.g. limitations in rotation step resolution. Paper describes the RRT method itself and its use for path planning of four-legged walking robot, including special failure case when robot is capable of only rotating in one direction. The method proved to be robust and fast.*

1. Introduction

There is a number of algorithms for solving the path planning problem. Probably most popular ones are probabilistic roadmap algorithm and randomized potential field algorithm. However, those algorithms generally do not extend well for general nonholonomic planning task. To avoid those problems a method of rapidly exploring random trees (RRT) can be successfully used. Following paragraphs describe the core of the method and show its use for path planning of legged walking robot.

2. Rapidly exploring random trees

Method of rapidly exploring random trees was first introduced by LaValle in 1998. Basically it is randomized data structure, which is sequentially expanded by creating new nodes of a tree structure in the direction of randomly selected points. RRT starts with initial state x_{init} and during its expansion it tries to find a goal state x_{goal} (this corresponds to general description of path planning as a search for continuous path from x_{init} to x_{goal} in metric space x_{goal}). During the search the vertices of RRT must be constructed in such a way that all nodes are in obstacle free space. Therefore it must be possible to test whether given state lies in

[*] Ing. Jiří Krejsa, PhD., ÚT AV ČR, pobočka Brno, Technická 2, 616 69, Brno, Czech Republic; tel: +420 541142885, email: jkrejsa@umt.fme.vutbr.cz

[+] Ing. Stanislav Věchet, PhD., VUT Brno, Technická 2, 616 69, Brno, Czech Republic, email: vechet@fme.vutbr.cz

X_{obst}, while states in X_{obst} can have various meanings, depending on application – most commonly it represents a configuration when robot is in collision with an obstacle.

Actual path planning algorithm which uses RRT is shown in Fig.1. First a root node in x_{init} state is created. Further more the RRT structure is iteratively expanded and once a while the algorithm tries to connect the nodes of RRT structure with goal state x_{goal}. RRT expansion is performed in following way: First a random state x_{rand} is generated with no restrictions or constraints. Then a closest node of existing RRT structure is found and new node is generated in Δx distance from found closest node towards a random state. Furthermore the restrictions are applied to newly generated node and if node meets all restrictions (both new node edge from closest node to new node lies in X_{free}, the complement of X_{obst}) it is added to RRT structure. If new node does not meet the restrictions it is simply discarded and the RRT expansion process continues.

```
1.  RRT.init ( x_init )
2.  repeat
3.      for i=1 to CONNECT_CHECK_INTERVAL
4.          x_rand = random state
5.          x_closest = GetClosestNode( x_rand )
6.          x_new = GenerateNewNode( x_closest , x_rand )
7.          x_new = ApplyRestrictions( x_closest , x_new )
8.          if ( x_new is OK)
9.              RRT.AddNewNode( x_closest , x_new )
10.         else
11.             RRT.Trapped
12.         end if
13.     end for
14.     RRT.TryConnectToGoal
15. until GoalReached
```

Figure 1: RRT construction algorithm for path planning

RRT key advantage is its fast expansion towards unexplored regions of the state space, in spite of simple construction method. This can be viewed in comparison with naive random tree which selects random state, random tree node and generates new node in direction from random tree node towards random state. The comparison of those two approaches for

$\Delta x = 40$ with numbers of three nodes sequentially 20, 60, 500 and 1000 is shown in Figure 2. World in this experiment (and in all further experiments) is of size 1000 x 1000, $x_{init} = (500, 500)$.

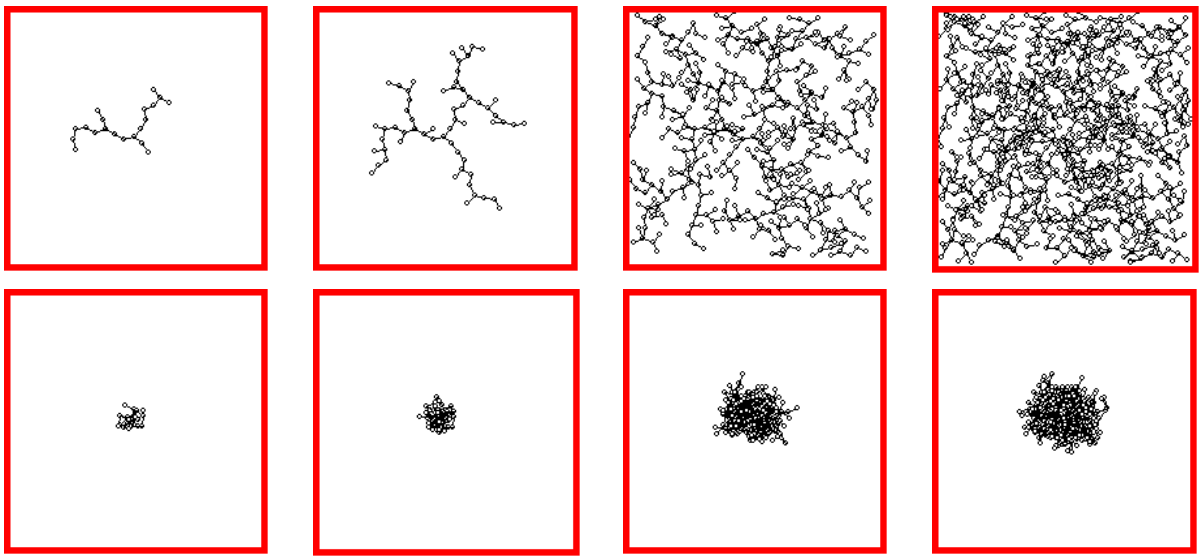


Figure 2. RRT expansion (top row) versus naive random tree (bottom row) expansion.

It is clear that naive random tree is strongly biased towards already places already explored, in contrast with RRT, which quickly expands and uniformly covers the search space. Uniform cover of search space is another advantage of RRT. During the initial expansion the nodes are not distributed uniformly, however as number of nodes increases the cover becomes uniform. This advantageous property is independent on the location of initial state x_{init}, as shown on Figure 3, when development of RRT is shown for the same world and Δx as in previous experiments, however, $x_{init} = (100, 100)$. One can see that RRT quickly expands towards unexplored regions and a uniform coverage is reached after a surprisingly low number of nodes is added to the structure.

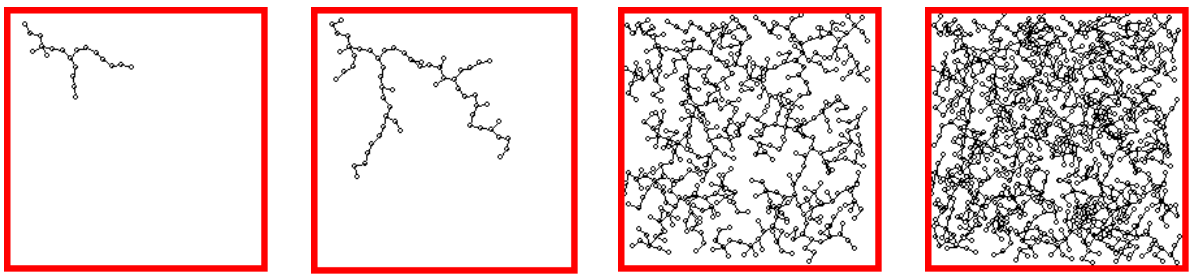
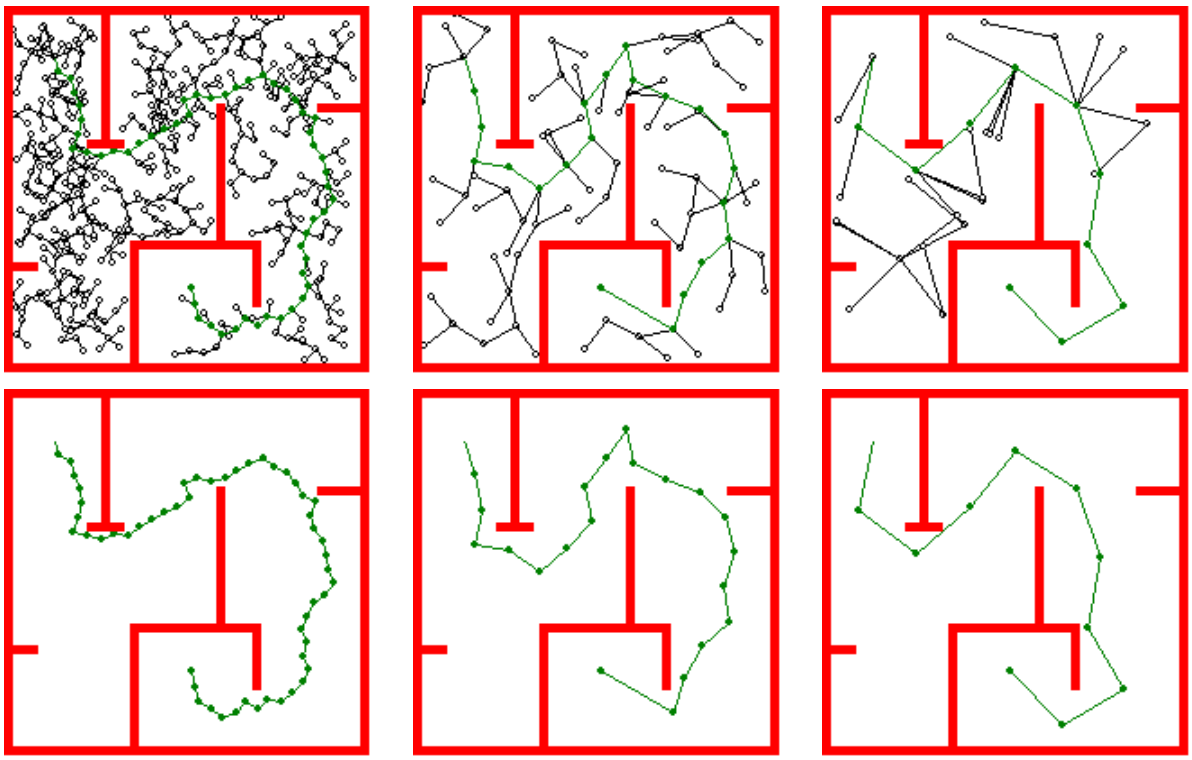


Figure 3. RRT expansion with $x_{init} = (100, 100)$

The distance between nodes is practically the only parameter of the expansion algorithm. Its influence on the speed is essential. However, determining the optimum value is a tricky problem, as the maximum depends on the characteristics of obstacles and character of further restrictions of robot move (velocity constraints, etc.). Important property of RRT is that reduction of Δx will slower the method, however, the search space will be eventually covered and path found. To illustrate the influence of Δx a simple experiment was performed and results are shown on Figure 4.



$\Delta x = 40$	$\Delta x = 100$	$\Delta x = 200$
$N_{nodes} = 490$	$N_{nodes} = 91$	$N_{nodes} = 35$
$N_{path} = 54$	$N_{path} = 22$	$N_{path} = 11$

Figure 4. Influence of the distance between neighboring nodes of RRT structure. N_{nodes} denotes the total number of nodes generated, N_{path} denotes the number of nodes in the path.

Figure 4. shows both resulting RRT structure and the found path from starting point to the goal point which were set to $x_{init} = (140,140)$, $x_{goal} = (520,780)$ with initial orientation of the robot to 0°. World size remains 1000×1000 as with previous experiments. Please note that left upper corner of the figures denotes the $(0,0)$ coordinates (this orientation is used in all subsequent figures). As can be seen, the increasing Δx speeds up the path planning, however, when further constraints are applied, the whole process can not just be slowed down, but whole path planning task can become unsolvable.

3. RRT for walking robot

Walking robots brings a number of constraints into path planning problem, mainly the limited resolution for translational and rotational movements. Even if in general those resolutions can be arbitrary, for real robot we often coupe with certain values corresponding to the single step of a robot. To use RRT in path planning for a robot with limited translational and rotational resolution we must incorporate those limitations into RRT expansion restrictions. For translation the Δx can be simply set to the single robot step in one direction, or to the multiples of such step. For rotation, a new restriction must be applied in steps 6 and 7 of the algorithm. When generating a candidate node, its position is generated so that orientation between candidate and closest node is as close to the orientation towards random state x_{rand} as possible, while keeping the angle equal to the multiples of single rotational step of the robot. Further the collision detection is applied as usual to keep both new state and vertex to the closes node obstacle free.

Parameters for path planning tasks shown further on were taken from the small four-legged walking robot shown in Figure 5. Each leg has 2DOF and is equipped with Hitec HS322 servodrives. Details can be seen in Věchet (2005).

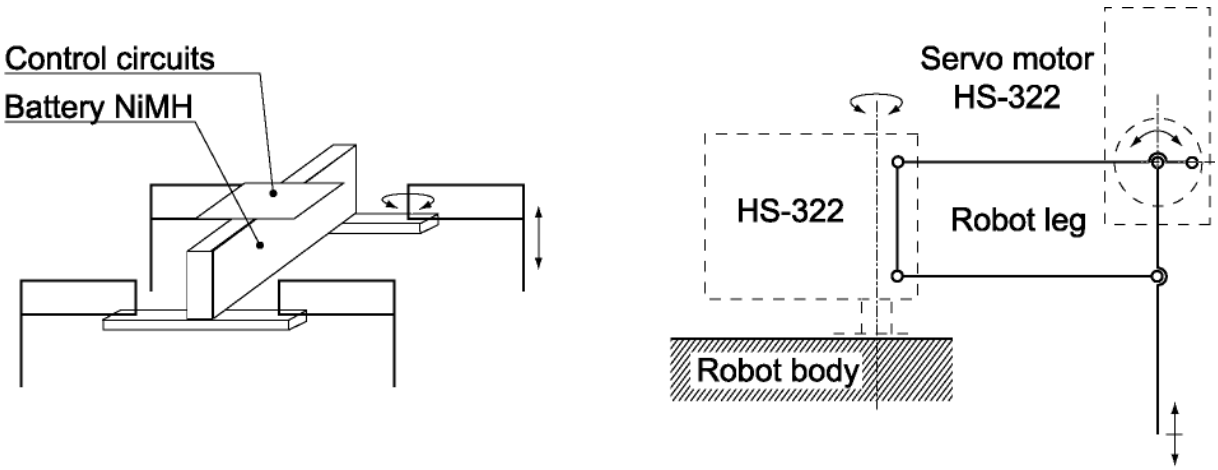


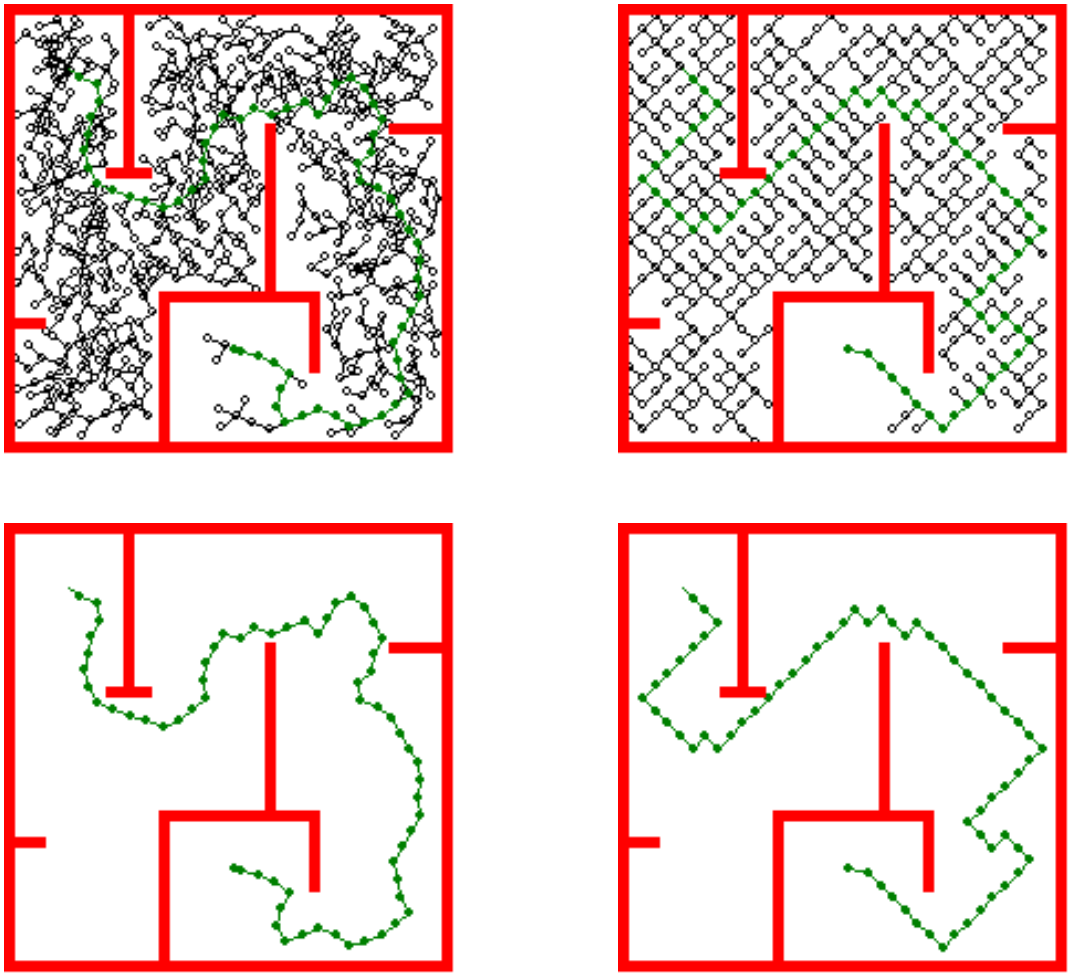
Figure 5. Four-legged walking robot.

The distance between neighboring nodes of RRT structure Δx was taken as the single step of the robot $\Delta x = 40$, smallest rotational resolution of the robot is 5°. For the initial experiments the rotational restriction was set to 20° and further to 45°. An example of path planning under those restrictions is shown in Figure 6.

RRT offers a simple way of incorporating further restrictions, which might for example cover a failure states of the robot. Further restrictions increase the time of solving the path planning problem, but even under rough restrictions the path is found. As an example the following restrictions were applied. 1. Robot can only rotate to the right (in range $\langle 0^\circ, 90^\circ \rangle$) with

$\alpha_{res} = 20^\circ$. 2. Robot can only rotate to the right with $\alpha_{res} = 10^\circ$, but in limited range $\langle 10^\circ, 90^\circ \rangle$ - it can not even go straight. For both cases the RRT found the path successfully, resulting full RRT tree structures and found path are shown on Figure 7.

As one can see, the resulting structure is quite large, the time for solving the path planning task were approximately 3 minutes for the first task and 13 minutes for the second, in contrast to previous experiments when search never exceeded half second (all experiments were performed on AMD Athlon/1500MHz). However, the path planning task which includes such restrictions is rather difficult to solve.



$\Delta x = 40,\ N_{nodes} = 809,\ N_{path} = 64$

$\alpha_{res} = 20^\circ$

$\Delta x = 40,\ N_{nodes} = 851,\ N_{path} = 67$

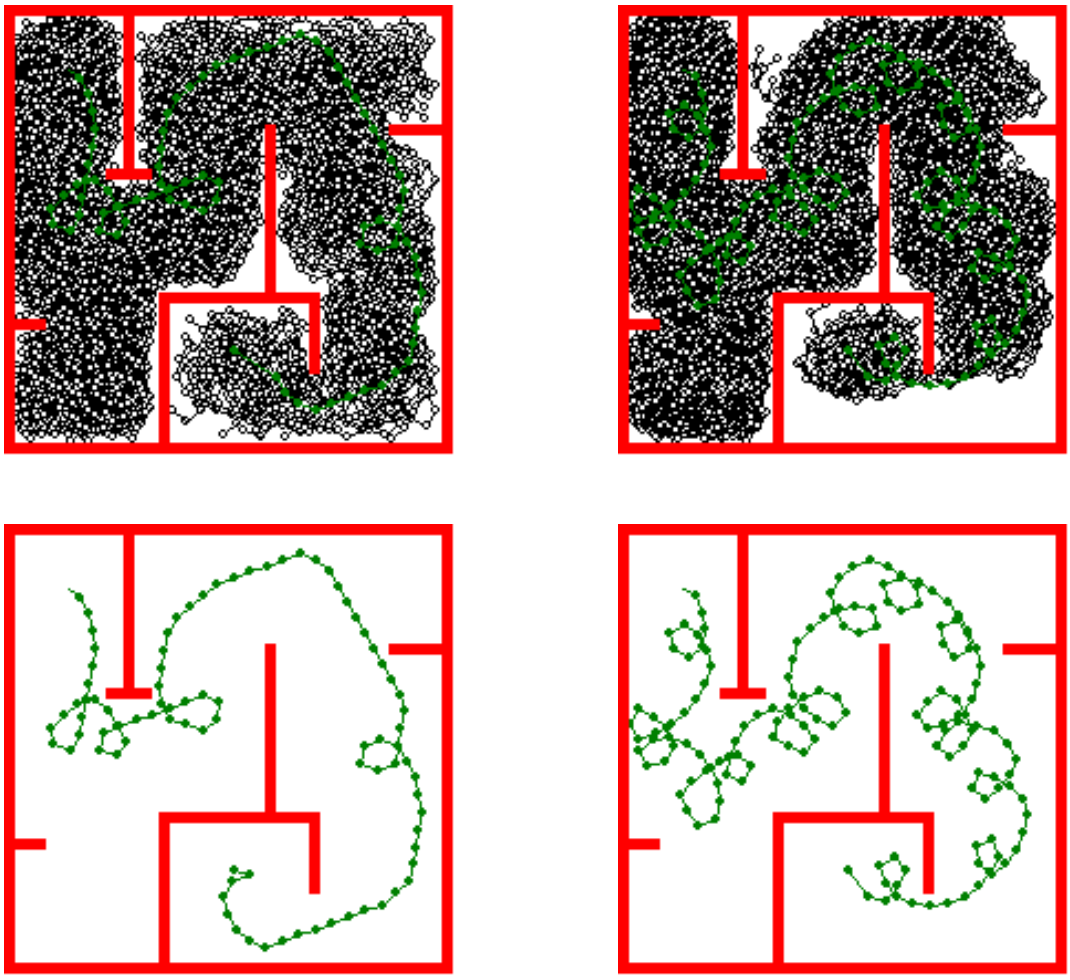
$\alpha_{res} = 45^\circ$

Figure 6. Path planning with limited angular resolution.

4. Conclusions

Rapidly exploring random trees based path planning is a reliable and fast method, which proved to be capable of successfully solving path planning problem which includes a number of constraints. The key advantage is quick expansion towards uncovered regions of search space and uniform distribution of RRT nodes over the search space.

RRT can be easily modified to include a number of constraints, as shown on example of walking robot path planning problem, when path was successfully found even for such restrictions which prohibited robot from moving straight and rotate to one side.



$\Delta x = 40,\ N_{nodes} = 14000,\ N_{path} = 89$

$\alpha_{res} = 20^{\circ},\ \alpha_{valid} \in \langle 0^{\circ}, 90^{\circ} \rangle$

$\Delta x = 40,\ N_{nodes} = 31000,\ N_{path} = 156$

$\alpha_{res} = 10^{\circ},\ \alpha_{valid} \in \langle 10^{\circ}, 90^{\circ} \rangle$

Figure 7. Path planning with further restrictions

5. Acknowledgement

Research published in this paper was supported by Czech Academy of Science under pilot project 92028 as part of research project AV0Z20760514.

6. References

LaValle S.M.: Rapidly-exploring random trees: A new tool for path planning, *technical report*, Computer Science Dept., Iowa State University, 1998

Věchet S., Krejsa J.: How to build a robot with no money, merkur, lego and old stepper motor, *in proceedings of Engineering Mechanics 2005*, Svratka