Applied Mechanics and Materials
ISSN: 1662-7482, Vol. 772, pp 471-476
doi:10.4028/www.scientific.net/AMM.772.471

Submitted: 2015-01-30
Revised: 2015-02-15
Accepted: 2015-02-25
Online: 2015-07-08

An Improvement of the Rapidly-Exploring Random Tree Method for Path Planning of a Car-Like Robot in Virtual Environment

GÎRBACIA Teodora [1, a*], MOGAN Gheorghe [1,b]

[1]Transilvania University of Braşov, 29 Eroilor Blvd., RO-500036, Braşov, Romania

[a]teodora.girbacia@unitbv.ro, [b]mogan@unitbv.ro

Keywords: path planning, car-like robot, computational complexity, optimization method

Abstract. In this paper is presented a method of reducing the computational complexity necessary in path planning for a car-like robot in order to generate the optimal path according to the constrains set by the user. The proposed method implies adding the following constrains: setting the maximum and minimum distance between the possible paths and the obstacles placed in the virtual environment in order to reduce the simulation time and to obtain a real-time application and to remove the paths that contain unnecessary turns around the environment without avoiding an obstacle. By applying this method the simulation complexity is reduced and the optimal path is easier to find.

Introduction

Path planning represents an important research area in robotics domain, consisting in selecting the optimal path between the start point and the target, without collision with the obstacles placed in the environment.

In order to obtain the optimal path there are many approaches of determining it and the way in which the autonomous robot follows it. One of these approaches is to use a virtual force field to simulate the path planning in a dynamic environment and a fuzzy controller for the robot navigation [1]. In an environment filled with obstacles, a smooth trajectory can be obtained by using a fuzzy controller and simulating Bezier curves to generate a continuous-curvature path [2]. In an unknown environment it can take a long time to generate an optimal path, so it is advantageous the use of FRIT (Follow and Reconnect with the Ideal Tree) search algorithm because it needs less time to obtain the shortest path [3]. Another possible constraint for planning a path for a car-like robot is to remain in a specified lane on the road and one of the best methods that can be used is the rapidly-exploring random tree (RRT) [4]. Another method recently developed is using genetic algorithms, but requires high complexity algorithms and computation resources. By using the Population-Based Incremental Learning (PBIL) technique the computational complexity decreases and an almost optimal path is obtained [5]. This method is used especially in path planning inside a grid [6].

There are many algorithms developed for path planning, but one of the most commonly used is A* or A* with post processing in discrete domains and Theta* or S-Theta* in continuous ones [7]. The Corridor Map Method (CMM) can be used in order to generate more natural looking and smooth paths [8].

Regardless of the chosen method for path planning, in order to obtain an optimal path, all the possible paths are analyzed and assessed according to a set of constrains initially established. The path planning algorithm must contain a collision detection module in order to create a safe path with no accidents between the car like robot and the obstacles. Such a path planning algorithm implies an increased time simulation in order to perform the computational complexity necessary to determine all possible paths. To obtain a real time simulation of finding the optimal path considering that the virtual environment could change by adding or removing obstacles, the computational complexity must decrease and the simulation time must be minimum.

In this paper is presented a method which establishes the maximum and minimum distance between the calculated path and the set of obstacles in order to reduce the number of all possible paths from which is chosen the optimal path that the car-like robot is going to follow. In this way, if

the dimensions of the virtual environment are too big, the possible paths will remain relatively close to the obstacles depending on the values of the parameters and will not be composed of unnecessary turns in the environment. The obtained paths will be fewer, shorter and, in this case, the shortest path will be easier to be determined and will not have unnecessary turns considering that one of the constrains selected by the user could be the minimum number of turns in order to obtain a continuous and safer path.

Path planning for car-like robots

In order to obtain a continous motion along the planned path for a car-like robot, especially when avoiding obstacles, the transition between a straight segmant and cicle arcs is done through using an additional curve. Considering the advantages presented by the clothoid, this has been chosen to be the additional curve used for steering the car-like robot. The most important characteristic of the clothoid is that its curvature varies linearly with its arclength and this allows the vehicle to travel along the path with constant high speed evan when it is steering.
Since clothoids are defined by complicated transcendental functions (Eq. 1), it requires an increased computational complexity in order to simulate in a virtual environment all the posible paths with a various number of steering.

$$F(s) = C_0 + \sqrt{\frac{\pi}{a}}\, e^{i(\varphi_0 - \frac{\rho_0}{2a})}\left[FC(\frac{\rho_0 + a\,s}{\sqrt{\pi a}}) - FC(\frac{\rho_0}{\sqrt{\pi a}})\right] + i\sqrt{\frac{\pi}{a}}\, e^{i(\varphi_0 - \frac{\rho_0}{2a})}\left[FS(\frac{\rho_0 + a\,s}{\sqrt{\pi a}}) - FS(\frac{\rho_0}{\sqrt{\pi a}})\right] \quad (1)$$

where: C_0 is the starting point of the curve, a- the smoothness of the curve, expressed by the derivative of the bend (a> 0), φ_0- clothoids steering angle in the starting point, ρ_0- the clothoid bend in the starting point, FC and FS are the Fresnel cosine and sine integrals (Eq. 2).

$$FC(x) = \int_0^x \cos(\frac{\pi}{2}u^2)\,du, \qquad FS(x) = \int_0^x \sin(\frac{\pi}{2}u^2)\,du. \quad (2)$$

The main repercussion of this high computational complexity is the greater length of time necessary to simulate all the possible paths that could contain a various number of steering depending on the positions of the obstacles and the dimensions of the virtual environment. Such being the case, the simulation may have a considerable delay and it would be difficult to follow the real time events.

One method to decrease the computational complexity in order to obtain a real time simulation is to reduce the number of possible paths initially calculated from which the user chooses the optimal path considering the constrains that he takes into account. Taking into account that the most commonly used constraints are finding the shortest path or the path with the minimum number of turns, the number of possible paths could be shrunk by adding an supplementary constraint that consists in establishing the minimum and maximum distance between the paths and the obstacles.

The virtual reality simulator

Recently many robot simulation platforms have been developed, offering various functionalities, but they are not fully portable. The Virtual Robot Experimentation Platform (V-REP) [9] offers many functionalities in order to create complex virtual environments and simulation scenarios. Also, this platform allows the user to add other programming techniques through plug-ins, remote API clients or ROS nods. In the virtual scene can be inserted many kinds of objects such as: joints, shapes, proximity, vision and force sensors, graphs, paths, dummies. Paths allow the definition of complex movement through the virtual environment especially for different kinds of robots.

This platform offers many calculation modules such as: kinematics module, dynamics module, collision detection module, mesh-mesh distance calculation module, path/motion planning module. The collision detection module allows checking the interference between any type of shapes or collection of shapes. This module uses data based on a binary tree of oriented bounding boxes for acceleration. This method is optimized by using a temporal coherency caching technique. The path planning module handles both holonomic and non-holonomic (for car-like vehicles) path planning tasks by using a derivation of the Rapidly-exploring Random Tree (RTT) algorithm[10].

The simulation in V-REP can be done using the bullet dynamics engine, ODE dynamics engine or real time simulation. By executing the script, the simulation time is incremented by the simulation time step, that is a parameter which can be set by the user. If the time step is too big, the simulation is fast, but inaccurate, but if the parameter is too small, the simulation takes longer. While running a real-time simulation, the simulation time may not be able to follow the real time due to high complexity computation.

The virtual environment created in V-REP initially consists in a 25 sq.m. smooth platform in the middle of which is placed a virtual car. In the virtual environment are placed two dummies that represent the start and the stop point of the path followed by the car. The start point is placed in the same position as the virtual car and the stop point is placed randomly in the virtual environment. At this moment, if the path planning module is accessed, the resulted path is represented by a strait line between the two points. Furthermore, in the environment are placed a collection of obstacles that have cuboid forms with the dimensions established by the user. All the objects from the collection have the following properties:

- collidable
- measurable
- detectable
- taken into consideration during path planning

The user can place as many obstacles as he wants in the virtual environment by adding a new object in the collection and establishing its value for translation and orientation in this way creating as many tipes of virtual scenes as he wants as illustrated in Fig.1.

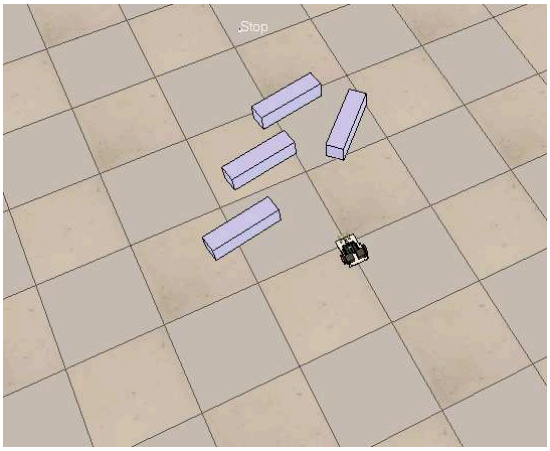

Fig.1 The virtual environment created with V-REP

Setting the minimum and maximum distance between the path and the obstacles

Theoretically the minimum distance between the path and the obstacles should be at least half the width of the virtual car because the start point is situated initially in the center of the car and the vehicle follows the path by situating itself simetrically on the path. But in reality, due to its kinematics, the vehicle will drift slightly from the path, generating errors in following it. This is the reason why the minimum distance between the path and the obstacles shoul be more than half the width of the virtual vehicle.

By not mentioning the maximum distance between the path and the obstacles, the path planning module will calculate all the possible paths in the virtual platform with the dimension of 25 sq.m. By setting this parameter, the searching area will be reduced according to the distribution of obstacles in the virtual environment. If the obstacles are distributed over the entire surface of the virtual platform, this parameter will not reduce the computational complexity of generating all the possible paths. But if the obstacles are grouped in a certain area of the virtual environment, as illustrated in Fig.1, by setting this parameter, the search area of the possible paths will be reduced. In this case, the computational complexity will be also decreased and the simulation time will closely follow the real time. Also, by setting this parameter, the possible paths will not be composed from unnecessary turns in the environment.

The experimental evaluation

The experiment consists in using the path planning module for different values atributed to the parameters representing the minimum and maximum distances between the generated paths and the obstacles placed in the virtual environment.After simulating these sets of values for the parameters, the user obtains the simulation time for each case and he can analize the complexity and numbers of possible paths.

In order to establish the values of the parameters, the user needs to know the dimensions of the virtual vehicle. In this case, after scaling the virtual vehicle, its length is 0.19m and its width is 0.09m. In Table 1 is indicated the simulation time for five different cases: without establishing the minimum and maximum distance between the path and the obstacle, case in which the simulator searches paths throughout the environment and four cases with different values for these parameters.

Table 1. The simulation time for different values atributed to the minimum and maximum distance

Minimum/Maximum Distance[m]	Simulation time[s]
Without parameters	0.74
Dmin=0.05 Dmax=0.5	0.59
Dmin=0.05 Dmax=0.7	0.56
Dmin=0.07 Dmax=0.7	0.63
Dmin=0.07 Dmax=0.5	0.67

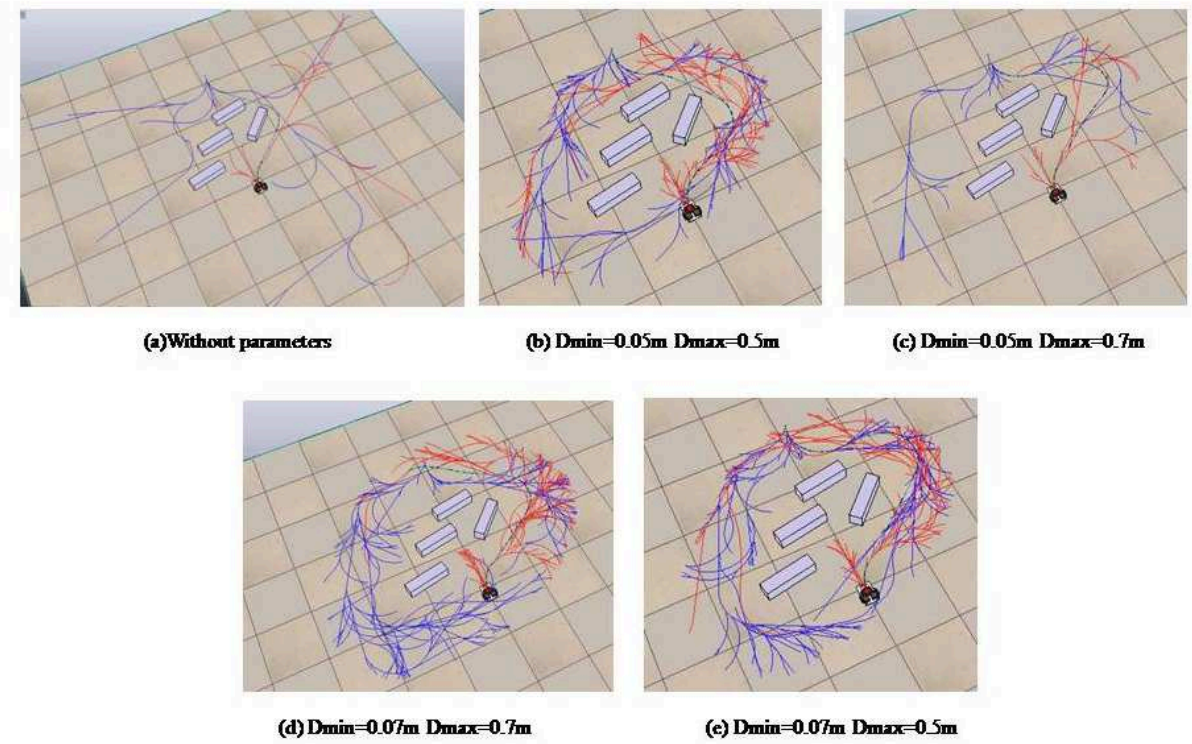

Fig.2. The posible paths or partial paths in the five experimental cases that emphasize the complexity of the simulation.

The user can also visualize the complexity of this simulation by analyzing all the possible paths or partial paths in the virtual environment as can be seen in Fig.2. Also in each case in the virtual environment is emphasized the shortest path By a green dot line.

After analyzing the simulations of this five cases the following conclusions were drawn:

- without setting this parameters, the possible paths may include unnecessary turns in the virtual environment that do not aim at avoiding obstacles;
- if the maximum distance between the path and the obstacles is too small, it may be difficult to realize the necessary turns in order to avoid the obstacles;
- if the minimm distance between the path and the obstacles is too big, the shortest possible path could be avoided;
- the optimal values for these parameters are: the minimum distance should be slightly bigger than half the width of the virtual car and the maximum distance should be big enough to allow the car to make the necessary steering in order to avoid the obstacles while restricting the search area for the optimal path.

Conclusions

In this paper is presented a method for reducing the computational complexity while using the Rapidly-exploring Random Tree method for path planning for a car-like robot in a virtual environment by adding the following constrains: setting the minimum and maximum distance between the possible paths and the obstacles placed in the virtual environment. After applying this method, the paths composed from unnecessary turns around the environment were removed from all the posible paths from which the optimal one is chosen. Also, by reducing the computational complexity, the simulation time is reduced and is able to follow the real time. By applying these constrains, the optimal path that may represent the shortest one or the path with a minimum number of turns is easyer to find.

Acknowledgements

This work was partially supported by the strategic grant POSDRU/159/1.5/S/137070 (2014) of the Ministry of National Education, Romania, co-financed by the European Social Fund – Investing in People, within the Sectoral Operational Programme Human Resources Development 2007-2013.

References

[1] J. Ni, W. Wu, J. Shen and X. Fan, An improved VFF approach for robot path planning in unknown and dynamic environments, Mathematical Problems in Engineering (2014), 1-10.

[2] J. Pérez, J. Godoy, J. Villagra and E. Onieva, Trajectory generator for autonomous vehicles in urban environments, IEEE International Conference Robotics and Automation (2013),409-414.

[3] N. Rivera, L. Illanes and J.A. Baier, Real-Time Pathfinding in Unknown Terrain via Reconnection with an Ideal Tree, Advances in Artificial Intelligence, (2014), 69-80.

[4] R.R. Radaelli, C. Badue, M.A. Gonçalves, T. Oliveira-Santos and A.F. De Souza, A Motion Planner for Car-Like Robots Based on Rapidly-Exploring Random Trees, Advances in Artificial Intelligence (2014), 469-480.

[5] M. Xu, J. Lee, S.K. Bahn and B.Y. Kang, Path planning for robot using Population-Based Incremental Learningm, Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), IEEE 4th Annual International Conference (2014), 474-479.

[6] B.Y. Kang, M.Xu, J. Lee and D.W. Kim, ROBIL: Robot Path Planning Based on a PBIL Algorithm, submitted to International Journal of Advanced Robotic Systems, 11 (2014), 1-14.

[7] P. Muñoz, D.F. Barrero and M.D. R-Moreno in: A Statistically Rigorous Analysis of 2D Path-Planning Algorithms, submitted to The Computer Journal (2014).

[8] R. Geraerts and M.H. Overmars, The corridor map method: a general framework for real-time high-quality path planning, Computer Animation and Virtual Worlds 18 (2007) 107-119.

[9] Information on http://www.coppeliarobotics.com.

[10] E. Rohmer, S.P. Singh and M. Freese V-REP: A versatile and scalable robot simulation framework, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), (2013),1321-1326.