The 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2015)
October 28 ∼ 30, 2015 / KINTEX, Goyang city, Korea

Non-holonomic Path Planning of Car-like Robot using RRT*FN

Sotirios Spanogianopoulos [1] and Konstantinos Sirlantzis [2]

[1,2]School of Engineering and Digital Arts, University of Kent, Canterbury, Kent (UK)
(Tel : +44(0)1227 824412 , E-mail: ss976@kent.ac.uk)

Abstract - Path planning of car-like robots can be done using RRT and RRT*. Instead of generating the non-holonomic path between two sampled configurations in RRT, our approach finds a *small incremental step* towards the next configuration. Since the incremental step can be in any direction we use RRT to guide the robot from start configuration to end configuration. Moreover, an effective variant of RRT called as RRT - Fixed Nodes (RRT*FN) is used to show the path planning of non-holonomic car-like robot. The algorithm is further tested with different static environments. The results show that RRT*FN implemented with non-holonomic constraints is able to find a feasible solution with the increased cost of number of iterations of RRT*FN while maintaining fixed number of nodes.

Keywords - Car-like robots, Non-holonomic, path planning, RRT*FN.

1. Introduction

Car-like mobile robot navigation has been a challenging field in the academic research over the last few decades. As these robots are mainly aimed for outdoor activities, corresponding navigation algorithms should be able to account for the constraints imposed by the non-holonomic type of movement allowable for car-like mobile robots. A very popular and successful family of navigation algorithms is based on the Rapidly-exploring Random Tree (RRT) path planning method. In this paper, some variety of modifications are proposed for the basic RRT algorithm that aim to improve the performance with respect to aspects, such as, time, path length, and trajectory smoothness, while observing the non-holonomic kinematic constraints.

Path planning using RRT for car-like robots has been well addressed in literature. In [1] authors extend the RRT algorithm to handle a large class of non-holonomic dynamical systems. While addressing computational complexity and asymptotic optimality of the system motion, the approach seeks connections within bounding boxes and computes the shape and orientation of these boxes for a large class of dynamical systems based on differential geometry using the ball-box theorem, where shapes can be constructed and joined in a fuzzy manner, i.e. without definitive boundary constraints. In [2], a five degrees of freedom dynamic car model is used that considers skidding and sliding. After exploring the configuration space using RRT, the RRT then converges to a uniform coverage of the configuration space by breaking the large Voronoi areas.

As the car-like robots are used outdoors, in [3] a novel RRT algorithm on rough terrains (RRT- RT) has been developed using the Roughness based Navigation Function (RbNF), which is a numerical function that provides the cost-to-go values for each terrain location. Simulation results show that the RRT-RT planner explores the terrain in an efficient manner, and generates final paths that slightly deviate from paths obtained by Dijkstras algorithm. In [4] numerous extensions made to the standard RRT algorithm that enable the on-line use of RRT on robotic vehicles. The sampling is done using the environmental structure to reduce the time in finding trajectories with various maneuvers. Further, in [5] tries to solve the problem of computing a complete motion to the goal within a limited time. There are also other notable approaches (for e.g. [6]) that do path planning in context of non-holonomic constraints.

Making RRT efficient has also been a prime goal of researchers. In [7], the obstacles in the configuration space are taken into account and a general framework for minimizing the effect of inappropriate sampling is developed based on the visibility region of the nodes in the tree. Also, in [8] a new sampling scheme is developed for a variant of the dynamic-domain RRT that iteratively adapts the sampling domain for the Voronoi region of each node during the search process. The boundary domain of a given node (i.e. its associated radius) is adapted as a function of the number of expansion attempts and failures from that node. Also, to ensure the probabilistic completeness of the algorithm, a lower bound on the possible radius values of the nodes is formulated when that nodes are extended.

Addressing the narrow passage problem while path-planning is crucial for RRT to support for non-holonomic constraints. As an example, in [9], the obstacle vector information is used to grow the tree in some nine possible ways in difficult areas of configuration space (C-space). A modification to a greedy algorithm was made for calculating the planner path, such that it would take as big a step length as possible, as long as it is less than some maximum step length specified.

If RRT has fixed number of nodes, then the memory cost remains constant, thus, making it ideal for use with outdoor car-like robots. The idea of having fixed number of nodes is already presented by RRT*FN [10]. However, for robot with non-holonomic constraints, this paper modifies and tests RRT*FN for path planning of such constrained robots in different static environments. The paper is organized as follows: Section 2.describes the preliminaries for path generation of non-holonomic car-like robot, Section 3. presents RRT*FN and its modification for car-like robot, Section 4. presents experiments and

978-1-4673-7971-7/15/$31.00

results, and Section 5.concludes the paper.

2. Preliminary

The motion of the car-like robot is commonly described by a *control* vector $\mathbf{u} = [\phi\ s]$, where ϕ is the steering angle of the front wheels and s is the motion speed. The path generation of such a robot satisfying non-holonomic constraints is introduced in this section.

2.1 Modelling kinematics of car-like robot

The configuration (or pose) of car-like robot can be expressed as $\mathbf{q} = [x\ y\ \theta]$, where (x, y) is the center position of axle of rear wheels and θ is the orientation of the robot w.r.t. world frame $\{W\}$ (see Figure 1). Let $\mathbf{Q}$ be the configuration space of the robot, which is three dimensional.

Kinematic model of the car-like robot satisfying non-holonomic constraints is widely known in literature [11]. Applying control vector $\mathbf{u_i} = [\phi_i\ s_i]$ to the robot at current configuration $\mathbf{q_i} = [x_i\ y_i\ \theta_i]$ for a small time interval dt_i, the resulting configuration $\mathbf{q_{i+1}} = [x_{i+1}\ y_{i+1}\ \theta_{i+1}]$ can be expressed using following equations:

$$\theta_{i+1} = \theta_i + \frac{s_i \sin\phi_i}{l} \times dt_i \quad (1)$$

$$x_{i+1} = x_i + s_i \cos\theta_{i+1} \cos\phi_i \times dt_i \quad (2)$$

$$y_{i+1} = x_i + s_i \sin\theta_{i+1} \cos\phi_i \times dt_i \quad (3)$$

, where l is the length between two axles.

If the current time is t_i, then $t_{i+1} = t_i + dt_i$ is the time when robot is at configuration $\mathbf{q_{i+1}}$. The small time interval $dt = t_{i+1} - t_i$ enables to make simple assumption such as the speed s_i remains nearly constant from time t_i to t_{i+1} and we can compute $\mathbf{q_{i+1}}$ directly using above differential equations.

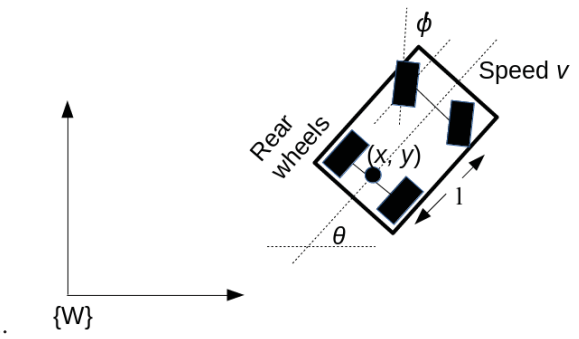


Fig. 1: Symbols used in describing the control vector $\mathbf{u}$ and the configuration $\mathbf{q}$ of the car-like robot.

The small interval dt_i is chosen such that the incremental distance the robot covers can be approximated by a constant straight line euclidean distance d_λ.

2.2 Non-holonomic path generation using small dt

The possible speed s_{i+1} for computing next configuration $\mathbf{q_{i+2}}$ is limited upon acceleration/de-acceleration a applied to the robot within time interval dt_i and the change in speed $ds_{i+1} = s_{i+1} - s_i$ can be expressed as:

$$-adt_i \leq ds_{i+1} \leq adt_i, \text{where } dt_i \leq \frac{d_\lambda}{ds_{i+1}} \quad (4)$$

The Algorithm 1 implements next configuration generation satisfying non-holonomic constraints and constraint expressed in eqn.(4). The algorithm requires presetting distance threshold d_λ, and max. magnitude of acceleration/de-acceleration of car a_{max}. To build a path Γ for car-like robot with n configurations, Algorithm 1 can be iterated for $i = 1$ to n given the starting configuration $\mathbf{q_0}$ and $\mathbf{u_{-1}} = [0\ 0]$.

Algorithm 1 Next configuration $\mathbf{q_i}$ generation

1: Input starting configuration $\mathbf{q_{i-1}}$ and $\mathbf{u_{i-2}}$
2: $r = [0, 1]$ be a random generator
3: $dt_{i-1} = \sqrt{\frac{d_\lambda}{a}}$, where $a = ra_{max}$
4: $s_{i-1} = s_{i-2} + (-1 + 2r)adt_{i-1}$
5:

$$\phi_{i-1} = \begin{cases} \phi_{i-2} & \text{if } r < 0.5 \\ \phi_{i-2} + (-1 + 2r)30^\circ & \text{Otherwise} \end{cases}$$

6: Compute $\mathbf{q_i} = [x_i\ y_i\ \theta_i]$ using eqn.(1– 3) with $\mathbf{u_{i-1}} = [\phi_{i-1}\ s_{i-1}]$ and $\mathbf{q_{i-1}} = [x_{i-1}\ y_{i-1}\ \theta_{i-1}]$

2.3 RRT and its relevant variants

The path generation using algorithm 1 needs a path planner to lead the car from starting configuration to a pre-desired goal configuration. Specifically, the planner can find a sequence of control actions $\mathbf{u}$ to drive the robot from initial configuration $\mathbf{q_s}$ to some final configuration $\mathbf{q_e}$ in presence of constraints imposed by environment and by robot dynamics. The prospects of RRT is cited in [12]. It randomly samples the configuration space of the robot and tries to connect that sampled configuration (node) $\mathbf{q_i}$ to its nearest configuration (node) $\mathbf{q_{near}}$ in the tree τ rooted at node $\mathbf{q_s}$.

For this paper, the relevant variants of RRT to be considered are RRT* [13] and RRT*FN [10]. RRT* introduces heuristics into RRT along-with the concepts of local neighbourhood of newly added node and its re-wiring. Thus, RRT* finds a feasible path Γ_{best} from $\mathbf{q_s}$ to $\mathbf{q_e}$ using a cost function $c(\Gamma)$.

RRT* Fixed Nodes (RRT*FN) minimizes memory requirements of RRT* by removing weak nodes in presence of high-performance node. The tree is grown identical to RRT* until a maximum M number of nodes is reached. If the algorithm does not find a feasible path connecting $\mathbf{q_s}$ to $\mathbf{q_e}$ then algorithm is restarted and the old node is removed whenever a new node is added.

3. Approach

RRT*FN framework can be used to develop path planning of robots with non-holonomic constraints, such as, car-like robot. In this section, the authors show on modifying RRT*FN to accommodate such constraints.

3.1 A review of RRT*FN

Similar to RRT the Algorithm 2 runs for N iterations and finds the probabilistically best path connecting robot configuration $\mathbf{q_s}$ to somewhere near to $\mathbf{q_e}$, using cost function $c(\Gamma)$. Similar to RRT It samples a node in $\mathbf{Q}$, finds the nearest neighbour $\mathbf{q_{near}}$, and then performs collision checking of trajectory of steering the robot from from $\mathbf{q_{near}}$ to $\mathbf{q_i}$. If that trajectory is collision free, then the node is considered to be added.

If the number of nodes exceeds max. number of allowed nodes M in a tree τ it removes a low performance node using force removal strategy or simply restoring the old tree with M nodes. Thus, a new node is only added to the tree τ if the node $\mathbf{q_i}$ at iteration i makes the path to $\mathbf{q_s}$ more cost-effective.

Algorithm 2 RRT*FN

Input $\mathbf{q_s}$, $\mathbf{q_e}$, N and M
*I*nitialize tree τ with one node $\mathbf{q_s}$
for i = 1 to N **do**
  if the number of nodes added to τ exceeds M **then**
    $\tau_{old} \leftarrow \tau$
  end if
  *S*ample a node $\mathbf{q_i}$ randomly in $\mathbf{Q}$
  $\mathbf{q_{near}} \leftarrow$ *n*earest node to $\mathbf{q_i}$ in τ
  $[\mathbf{q_i}\ u_{i-1}] \leftarrow$ *S*teer the robot from $\mathbf{q_{near}}$ to $\mathbf{q_i}$
  if Steering the robot from $\mathbf{q_{near}}$ to $\mathbf{q_i}$ is *c*ollision-free **then**
    *I*nsert node $\mathbf{q_i}$ in τ using control vector $\mathbf{u_{i-1}}$
    Using local neighbourhood *r*e-wire the node $\mathbf{q_i}$ to some other parent that minimizes path cost to $\mathbf{q_s}$
    if $\exists\ \tau_{old}$ and new node $\mathbf{q_i}$ is cost-effective **then**
      if $\exists$ a node $\mathbf{q_\emptyset}$ with one or no child **then**
        Force remove that node $\mathbf{q_\emptyset}$
      end if
    end if
    if no node can be removed or new node $\mathbf{q_i}$ is not cost-effective **then**
      $\tau \leftarrow \tau_{old}$ when *no removal performed*
    end if
  end if
end for
$c(\Gamma_{best}) = \min\{c(\Gamma), \forall$ feasible Γ from $\mathbf{q_s}$ to near $\mathbf{q_e}$
*r*eturn Γ_{best}

3.2 RRT*FN with non-holonomic constraints

Algorithm 3 (called as RRT*FN-NH) proposes on making RRT*FN work for robot with non-holonomic constraints. Similar to RRT it samples a node $\mathbf{q}$ in the configuration space $\mathbf{Q}$ of the robot but does not uses that node to connect to nearest node $\mathbf{q_{near}}$ in tree τ. Instead, using Algorithm 1 it generates the next configuration $\mathbf{q_j}$ using $\mathbf{q_{near}}$ and control vectors $\mathbf{u_{i-2}}$ and $\mathbf{u_{i-1}}$. The parent control vector $\mathbf{u_{i-2}}$ is needed in generating the next configuration $\mathbf{q_j}$ (Algorithm 1). Using the output control vector $\mathbf{u_{i-1}}$ of Algorithm 1 the robot is steered to the next incremental configuration. Thus, RRT*FN-NH while satisfying non-holonomic constraints explores the configuration space and tries to connect to the goal using similar principle of RRT*FN.

There are few additional differences between RRT*FN and RRT*FN-NH. The collision checking of robot configuration is done instead of a trajectory while steering the robot from $\mathbf{q_{near}}$ to $\mathbf{q_j}$. As the next incremental step is small, possibility of finding a feasible configuration is much higher than a trajectory steering the robot. This also provides simple computationally less-intensive collision checking of a robot configuration instead of continuous collision checking of a trajectory. Also, another difference is the re-wiring strategy is removed from the algorithm as it may not be effective for non-holonomic systems.

Algorithm 3 RRT*FN-NH

*I*nitialize tree τ with one node $\mathbf{q_s}$ and $\mathbf{u_{-1}} = [0\ 0]$
for j = 1 to N **do**
  if the number of nodes added to τ exceeds M **then**
    $\tau_{old} \leftarrow \tau$
  end if
  *S*ample a node $\mathbf{q}$ randomly in $\mathbf{Q}$
  $\mathbf{q_{near}} \leftarrow$ *n*earest node to $\mathbf{q}$ in τ
  $\mathbf{u_{i-2}} \leftarrow$ the control vector applied at parent of $\mathbf{q_{near}}$
  $[\mathbf{q_i}\ u_{i-1}] \leftarrow$ Using Algorithm 1 *g*enerate $\mathbf{q_i}$ with $\mathbf{q_{i-1}} = \mathbf{q_{near}}$ and $\mathbf{u_{i-2}}$
  $\mathbf{q_j} = \mathbf{q_i}$ and store u_{i-1} with node $\mathbf{q_j}$
  if robot at $\mathbf{q_i}$ is *c*ollision-free **then**
    *I*nsert node $\mathbf{q_i}$ in τ using control vector $\mathbf{u_{i-1}}$
    if $\exists\ \tau_{old}$ and new node $\mathbf{q_j}$ is cost-effective **then**
      if $\exists$ a node $\mathbf{q_\emptyset}$ with one or no child **then**
        Force remove that node $\mathbf{q_\emptyset}$
      end if
    end if
    if no node can be removed or new node $\mathbf{q_j}$ is not cost-effective **then**
      $\tau \leftarrow \tau_{old}$ when *no removal performed*
    end if
  end if
end for
$c(\Gamma_{best}) = \min\{c(\Gamma), \forall$ feasible Γ from $\mathbf{q_s}$ to near $\mathbf{q_e}$
*r*eturn Γ_{best}

4. Experiments and Results

Experiments and Results were conducted on modified RRT*FN toolbox [10] in Matlab. We tested the RRT*FN with non-holonomic constraints in five different environments. The car robot tested had dimension 0.58×0.38 unit^2. The minimum and maximum velocity of the car was set to 0.001 and 0.05 unit per seconds respectively. The length between the two axles of the car was 0.38 units. The maximum acceleration of car was 0.04 units/sec^2.

The incremental or discritization step d_λ was set to a

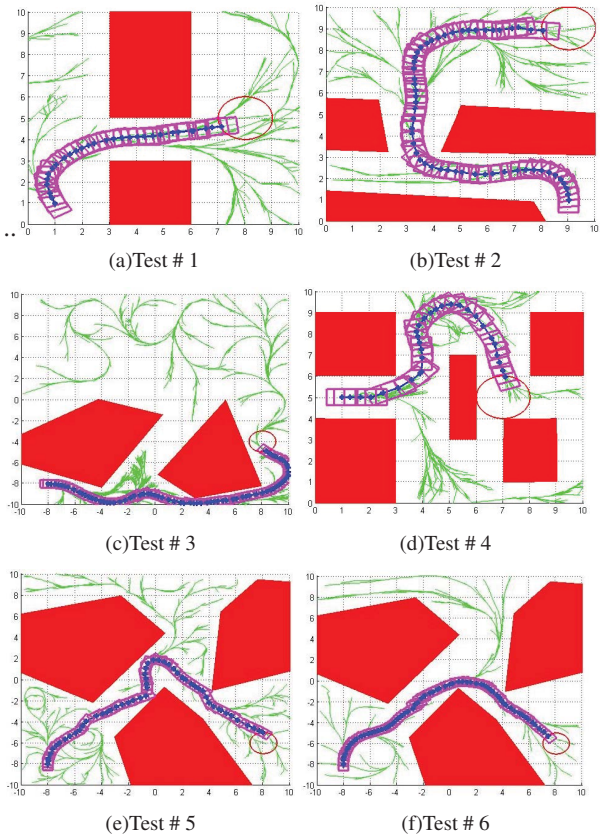
constant of 0.1 units. Table 1 shows the results of six tests with following output parameters: L as path length and T as total time taken for robot to reach the goal. The input parameters to RRT*FN for non-holonomic constraints are N (the number of iterations in RRT*FN) and M (the max. number of nodes in tree τ).

Test # 1 shows a traditional example of narrow passage (Fig. 2(a)) in path planning problem. Test # 2 enforces the car to take multiple turns (Fig. 2(b)) without stopping the car as min. velocity of car is greater than 0. Test # 3 checks if the car finds the shortest route (Fig. 2(c)) compared to longer route; hence, the number of iterations are set to 50,000. Test # 4 checks if the car can take sharper turn (U-turn) to reach the goal (Fig. 2(e)). Test # 5 (Fig. 2(e)) and # 6 (Fig. 2(f)) are for the same environments but with different set accelerations that result in different total time.

As seen in Figure 2 RRT*FN for non-holonomic constraints always find a solution but requires iterations of at least 10,000. Two factors affect the number of iterations, the discritization step and the non-holonmic constraints of the robot. As seen in results L and T is crucial parameters and relevant to environment and the constraints of the car-like robot movement. Also, for Test # 4, the acceleration was reduced to 0.004 units/sec^2 to obtain feasible solution in 10,000 iterations, but total time increased. So we conducted test # 6 with the same acceleration as original and with 10,000 iterations no feasible solution was obtained. So we had to increase the iterations upto 50,000. Thus, if the speed of the car is high most of the time then the chances of it hitting the obstacles are higher and more iterations will be needed.

Table 1: Experimental data of RRT*FN with Nonholonomic constraints

Test #	**N** ($\times 10^3$)	**M** ($\times 10^3$)	**L** (units)	**T** (secs)
1	10	1	9.02	67.27
2	10	1	18.25	124.1
3	40	15	24.08	162.6
4	10	1	13.12	221.29
5	10	1	27.40	395.01
6	50	1	22.08	156.085



(a)Test # 1 (b)Test # 2

(c)Test # 3 (d)Test # 4

(e)Test # 5 (f)Test # 6

Fig. 2: A path Γ_{best} returned by RRT*FN NH

5. Conclusion

This paper was focused on adapting and testing path planning of car-like robots using an existing variant of RRT, known as RRT*FN. By using incremental small next step and the modified path planning algorithm RRT*FN-NH for robots with non-holonomic constraints, the paper demonstrated experiments of car-like robot in different challenging static environments with additional constraint of having minimum velocity of car. The tests showed that to obtain feasible solution, the number of iterations required are in magnitude of 10^3. Future work includes testing RRT*FN-NH in real-world static environments and to deal with motion uncertainty of car-like robot.

Acknowledgement

This work was part of the SAVEMORE project co-funded by the European Regional Development Fund and the School of Engineering and Digital Arts, University of Kent, UK. SAVEMORE was selected for funding under the Interreg IVA France (Channel) England programme.

References

[1] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *IEEE International Conference on Robotics and Automation (ICRA),*, pp. 5041–5047, May 2013.

[2] R. Pepy, A. Lambert, and H. Mounier, “Path planning using a dynamic vehicle model,” in *Information and Communication Technologies, 2006. ICTTA ’06. 2nd*, vol. 1, pp. 781–786, 2006.

[3] A. Tahirovic and G. Magnani, “A roughness-based rrt for mobile robot navigation planning,” in *In IFAC World Congress*, vol. 18, pp. 5944–5949, 2001.

[4] Y. Kuwata, G. Fiore, J. Teo, E. Frazzoli, and J. How, “Motion planning for urban driving using rrt,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1681–1686, Sept 2008.

[5] S. Petti and T. Fraichard, “Safe navigation of a car-like robot in a dynamic environment,” in *Proceedings of the European Conference on Mobile Robots*, 2005.

[6] S. Balakirsky and D. Dimitrov, “Single-query, bi-directional, lazy roadmap planner applied to car-like robots,” in *IEEE International Conference on Robotics and Automation*, pp. 5015–5020, May 2010.

[7] A. Yershova, L. Jaillet, T. Simeon, and S. LaValle, “Dynamic-domain rrts: Efficient exploration by controlling the sampling domain,” in *IEEE International Conference on Robotics and Automation*, pp. 3856–3861, April 2005.

[8] L. Jaillet, A. Yershova, S. La Valle, and T. Simeon, “Adaptive tuning of the sampling domain for dynamic-domain rrts,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2851–2856, Aug 2005.

[9] S. Rodriguez, X. Tang, J.-M. Lien, and N. Amato, “An obstacle-based rapidly-exploring random tree,” in *Proceedings IEEE International Conference on Robotics and Automation*, pp. 895–900, May 2006.

[10] O. Adiyatov and H. Varol, “Rapidly-exploring random tree based memory efficient motion planning,” in *IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 354–359, 2013.

[11] L. O. Noureddine Ouadah and F. Boudjema, “Car-like mobile robot oriented positioning by fuzzy controllers,” in *International Journal of Advanced Robotics System*, vol. 5, pp. 249–256, 2008.

[12] S. M. LaValle and J. J. Kuffner Jr, “Rapidly-exploring random trees: Progress and prospects,” 2000.

[13] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, “Anytime motion planning using the rrt*,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1478–1483, May 2011.