Application of the RRT Algorithm to Solve Control Theory Problems

Kira Dmitrieva[1] and Majid Abbasov[1][0000-0003-1484-4733]

[1] Universitetskaya Emb., 13B, St Petersburg 199034, Russia

Abstract. This paper presents a modified Rapidly-exploring Random Tree (RRT) algorithm for solving optimal control problems in nonlinear dynamical systems. The key innovation lies in integration of system dynamics direct into the tree expansion process. Unlike the classical RRT, which samples points in the configuration or state space, the proposed approach samples controls from a mixed continuous-discrete probability distribution. Each sampled control is applied to the nearest node in the tree, and the resulting child state is computed by numerically integrating the system's ordinary differential equations over a fixed time step. To explicitly track and minimize the objective functional, the state vector is augmented with the accumulated cost, forming an extended state space. This framework transforms the RRT from a path-planning tool into a stochastic optimizer capable of approaching a control trajectory that satisfies the necessary conditions for optimality while respecting state and control constraints. The method is validated through numerical simulation, demonstrating its effectiveness in generating feasible and near-optimal control policies for complex dynamical systems.

Keywords: optimal control, graph tree, fast growing graph trees, control theory.

1 Introduction

The Rapidly-exploring Random Tree (RRT) algorithm was introduced by LaValle (1998) as a randomized data structure designed for path planning problems with nonholonomic constraints [1]. Unlike grid-based methods, RRT efficiently explores high-dimensional state spaces by incrementally building a tree toward randomly sampled nodes. A major advancement came with RRT-Connect, developed by Kuffner and LaValle (1999), which grows two trees simultaneously—one from the start and one from the goal—significantly improving convergence speed for single-query problems [2]. While the original RRT guarantees probabilistic completeness, it lacks asymptotic optimality. This limitation was addressed by Karaman and Frazzoli (2011), who proposed RRT*, a variant that introduces rewiring of the tree to ensure almost-sure convergence toward the optimal path as the number of iterations increases [3]. Following the introduction of RRT*, numerous modifications have focused on accelerating convergence and improving solution quality. A key challenge in applying RRT to control problems is the incorporation of system dynamics and cost functions.

The classical RRT framework assumes geometric steering, which is insufficient for systems governed by ordinary differential equations (ODEs). LaValle and Kuffner laid the groundwork for kinodynamic planning by extending RRT to systems with differential constraints. Later it was demonstrated how RRTs can be adapted for nonlinear control and hybrid systems, replacing geometric distance metrics with cost-to-go heuristics and integrating local planners for dynamic feasibility. However, despite significant progress, RRT-based algorithms face persistent theoretical and practical challenges.

2 Problem statement

Consider search for optimal control problem with parameters:

$$\begin{cases} \ddot{x} + \omega^2 x = u, t \in [0; T] \\ x(0) = x_0 \\ x(T) = x_* \\ x(t) \in X, t \in [0; T] \\ u(t) \in U, t \in [0; T] \end{cases}$$

$x(t) \in \mathbb{R}^n$- state vector
$u(t) \in \mathbb{R}^m$- control vector
$X \subseteq \mathbb{R}^n$- state space
$U = [u_{min}; u_{max}]$- vector of permissible controls
ω – constant parameter, angular frequency
$T > 0$ – control horizon

The equation can be rewritten as a system: $\begin{cases} \dot{x_1} = x_2 \\ x_2 = -\omega^2 x_1 + u \end{cases}$

The task is to transfer the system from point x_0 to point x_* in time $T > 0$ minimizing the functional J:

$$J = \frac{1}{2}\int_0^T u^2(t)dt \quad (1)$$

Consider the system $z = \begin{pmatrix} x \\ s \end{pmatrix} : \dot{s}(t) = \frac{u^2}{2}$

Then the system can be rewritten as:

$$\dot{z}(t) = \begin{pmatrix} x_2 \\ -\omega^2 x_1 + u \\ \frac{u^2}{2} \end{pmatrix} \quad (2)$$

That is, the problem reduced it to the finding a trajectory in an extended space: $\forall x \in S: S = \{z = \begin{pmatrix} x^* \\ s \end{pmatrix} : s \in \mathbb{R}\}$ with minimizing s(T). Thus, the problem is reduced to finding a trajectory in an extended state space minimizing the terminal cost. The RRT algorithm is based on probabilistic exploration of space and requires stochastic control expansion. A random sample of controls from probability distributions is used to construct a tree of reachable states:

$$U\sim \begin{cases} DUnif(\{x \in \mathbb{Z}: x \in [u_{min}; u_{max}]\}), with\ p\ probability \\ Unif([u_{min}; u_{max}]), with\ 1-p\ probability \end{cases} \quad (3)$$

Stochastic control selection guarantees that the tree will reach the target state with probability 1 as the number of iterations approaches infinity. Random sampling allows the algorithm to avoid local traps in the control space. For complex control systems with hidden spatial geometry, stochastic exploration is often more efficient than deterministic grid methods. With a sufficient number of iterations, the resulting control asymptotically approaches a control satisfying the necessary optimality conditions.

Such hybrid distribution is a heuristic. We assume that discrete part gives “jumps” to control space, giving tree more options and directions to grow, avoiding local “traps”. Continuous part helps with smoothing the found trajectories, locally improving found solutions and making them more precise. Parameter p controls the balance between exploration and refinement. The larger p is, the more continuous samples there are, the better the local improvement, and the worse the global coverage. The smaller p is, the stronger the discrete phase, the faster the coverage of rough directions, but the slower the fine tuning. A practical scheme can be described this way: start with a small pp (almost purely discrete mode) and gradually increase pp as the tree grows, in order to quickly cover the space first and then further optimise the found trajectories. This will be analogous to coarse-to-fine in optimisation tasks.

3 Algorithm

3.1 Algorithm modification description

The original RRT algorithm builds a tree in the state space by sampling random points, finding the nearest node in the tree, and extending it toward the sample via straight-line motion (or a local steering function) over a fixed distance. This process does not consider system dynamics or path costs.

Our modified code incorporates system dynamics by defining an ODE for state evolution. The state is numerically integrated over a fixed time step dt. Each state in the tree is augmented with its accumulated cost, forming an extended state vector [x, s]. This transforms the tree into a cost-aware search structure that explicitly tracks and minimizes the total cost to reach the goal. While extensions like RRT* introduce rewiring for asymptotic optimality, they typically lack this explicit integration of cost propagation within the dynamics.

New nodes are generated not by extending toward a random state, but by applying randomly sampled controls to the nearest tree node. These controls are drawn from a mixed continuous-discrete distribution. The resulting child state is computed by numerically integrating the system ODE over dt under the applied control. This approach ensures that all generated trajectories are dynamically feasible, as they respect the system dynamics exactly. The new state is then validated against state and control constraints, as well as dynamic feasibility checks. Similar to the standard RRT, the proposed method does not guarantee that the found trajectory is optimal. Therefore, claims regarding asymptotic optimality are not made.

3.2 Algorithm code description

The algorithm builds a tree of dynamically feasible trajectories by repeatedly selecting a random state (for nearest neighbor guidance), picking a random control from a mixed distribution, integrating the dynamics forward, and adding the resulting state to the tree if valid. The augmented state with accumulated cost enables direct comparison of trajectory quality when multiple paths reach the goal.

The algorithm takes the following parameters as input:

1. $x \in [-2; 2] \times [-2; 2]$
2. $u \in U$, where $U \sim \begin{cases} DUnif(\{x \in \mathbb{Z}: x \in [u_{min}; u_{max}]\}), with\ p\ probability \\ Unif([u_{min}; u_{max}]), with\ 1 - p\ probability \end{cases}$
3. $T = 100$c
4. $\omega = 1.0$
5. $Integration\ timestep\ dt$

Pseudocode

Algorithm 1 RRT for dynamic system

class `RRT`
  input:
    `dynamics`: function $f(x, u) \to \frac{dx}{dt}$
    `cost_rate`: function $c(x, u) \to$ cost
    `state_bounds`: $(x_{\min}, x_{\max})$ — state space bounds
    `control_bounds`: $(u_{\min}, u_{\max})$ — integer control space bounds
    `prob_contr`: probability of selecting a continuous control
    `goal_tol`: goal reaching tolerance
    dt: integration time step
    $T_{\max}$: time horizon
  initialize:
    `goal_tol`, dt, $T_{\max}$
    `n_state` ⟵ dimension of the state space
    `max_iter` ⟵ $\lfloor \frac{T_{\max}}{dt} \rfloor$
  method `random_control()`:
    `u_rand` ⟵ random number from $U[0, 1]$
    if `u_rand` $<$ `prob_contr`:
      return random control sampled from $U[u_{\min}, u_{\max}]$
    else:
      return random integer control sampled from discrete uniform distribution on $\{u_{\min}, \ldots, u_{\max}\}$
  method `integrate`(z_{near}, u):
    // $z_{\text{near}} = (x_{\text{near}}, s_{\text{near}})$, s — accumulated cost
    define ODE:

$$\begin{cases} \frac{dx}{dt} = \texttt{dynamics}(x, u) \\ \frac{ds}{dt} = \texttt{cost\_rate}(x, u) \end{cases}$$

    solve ODE from $t = 0$ to $t = dt$ with initial conditions $z_0 = (x_{\text{near}}, s_{\text{near}})$
    if integration succeeds:
      return $z_{\text{new}} = (x_{\text{new}}, s_{\text{new}})$
    else:
      return *null*
  method `is_state_valid`(x):
    return *true* **if** $x \in [x_{\min}, x_{\max}]$ **else** *false*
  method `goal_reached`(x, x_{goal}):
    return *true* **if** $\|x - x_{\text{goal}}\| \leqslant$ `goal_tol` **else** *false*
  method `plan`(x_0, x_{goal}):
    z_{start} ⟵ $(x_0, 0)$
    initialize `tree`:
      `states` ⟵ $[z_{\text{start}}]$
      `controls` ⟵ $[null]$
      `parents` ⟵ $[-1]$
    `best_cost` ⟵ $+\infty$
    `best_path` ⟵ *null*
    `best_controls` ⟵ *null*
    for i **from** 0 **to** `max_iter` $- 1$:
      x_{rand} ⟵ random node in $[x_{\min}, x_{\max}]$
      `idx` ⟵ index in `tree.states` which minimizes $\|x_{\text{rand}} - \text{state}.x\|$
      z_{near} ⟵ `tree.states[idx]`
      u ⟵ `random_control()`
      z_{new} ⟵ `integrate`(z_{near}, u)
      if z_{new} is *null*: **continue**
      x_{new} ⟵ first `n_state` components of z_{new}
      s_{new} ⟵ last component of z_{new} (accumulated cost)
      if not `is_state_valid`(x_{new}): **continue**
      `tree.states.add`(z_{new})
      `tree.controls.add`(u)
      `tree.parents.add(idx)`
      if `goal_reached`$(x_{\text{new}}, x_{\text{goal}})$ **and** $s_{\text{new}} <$ `best_cost`:
        `best_cost` ⟵ s_{new}
        `best_path` ⟵ **reconstruct** path from new vertex to root via `tree.parents`
        `best_controls` ⟵ sequence of controls along this path (excluding first *null*)
    return (`tree`, `best_path`, `best_controls`, `best_cost`)

Results

Here you can see the results of the algorithm with parame:

$$x_0 = [0; 0]$$

$$x_{goal} = [1; 1]$$

dt step = 0.1

T = 100

Probability p = 0.5

After 83 iterations, we get such trajectory. Its length is 18 steps. The cost J equals 3.006.

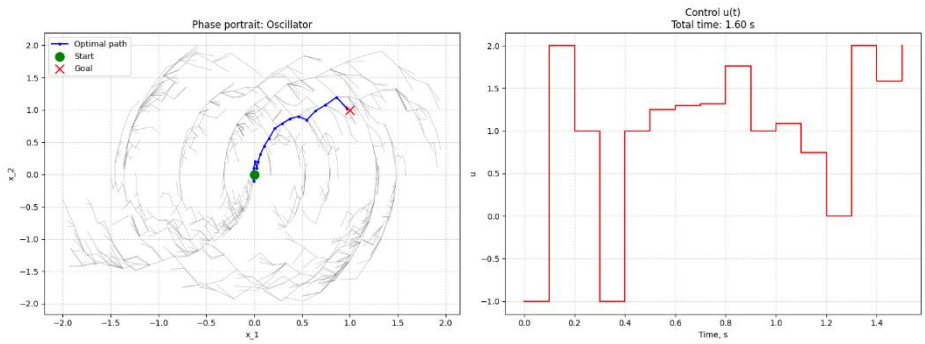

Fig. 2. Oscillator phase portrait and control graphs.

If we run experiments with 30 random starting positions, we obtain stable statistics: 80% of the cases are successful. The average number of iterations is 238.

References

1. S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University. http://janowiec.cs.iastate.edu/papers/rrt.ps, Oct. 1998.
2. J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 2000, pp. 995-1001 vol.2, doi: 10.1109/ROBOT.2000.844730.
3. S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. International Journal of Robotics Research, 30(7):846-894, June 2011.
4. Klemm, Sebastian & Oberlander, Jan & Hermann, Andreas & Roennau, Arne & Schamm, Thomas & Zollner, J. & Dillmann, Rudiger. (2015). RRT*-Connect: Faster, Asymptotically Optimal Motion Planning. 10.1109/ROBIO.2015.7419012.

5. Kang, J.-G.; Lim, D.-W.; Choi, Y.-S.; Jang, W.-J.; Jung, J.-W. Improved RRT-Connect Algorithm Based on Triangular Inequality for Robot Path Planning. Sensors 2021, 21, 333.
6. Abbasov, M. E. O. & Dmitrieva, K. A., 2024, Information Technologies and Their Applications, 2nd International Conference on Information Technologies and Their Applications Baku, ITTA 2024, Part 3.
7. Dmitrieva, K. A. & Abbasov, M. E. O., 2024, Collection of works of the All-Russian conference on natural sciences and humanities with international participation "Science of SPbSU – 2023". pp. 100-102
8. Dmitrieva, K. A., 2023, Control Processes and Stability: Proceedings of the 54th International Scientific Conference of Graduate Students and Students. Publishing House Fedorova G.V., Volume 10(26). pp. 259 (Control Processes and Stability).