Uniform and Efficient Exploration of State Space using Kinodynamic Sampling-based Planners

Rakhi Motwani, Mukesh Motwani, and Frederick C. Harris Jr.

Abstract Sampling based algorithms such as `RRTs` have laid down the foundation for solving motion planning queries for systems with high number of degrees of freedom and complex constraints. However, lack of balanced state-space exploration of `RRTs` calls for further improvement of these algorithms. Factors such as drift, underactuation, system dynamics and constraints, and the lack of an energy/time based distance metric in state space can cause `RRT` propagation to be uneven. This paper focuses on improving the coverage of the `RRT` algorithm for physical systems that demonstrate a tendency to restrict the growth of `RRT` to certain regions of the state space. A localized principal component analysis based approach is proposed to learn the propagation bias of state-space points sampled on a grid during an offline learning phase. To compensate for this bias, expansion of the `RRT` in real-time is steered in the direction of the least principal component of the propagation of the state-space sample selected for expansion. The algorithm is tested on various systems with high degres of freedom and experimental results indicate improved and uniform state-space coverage.

Key words: uniform state-space coverage, local-PCA based `RRT`

1 Introduction

Motion planning algoirthms can be categorized as Exact planning methods, Analytical solutions, Numerical approaches, and Approximate methods [3]. Exact methods [2, 13], Analytical solutions [11] and Approximate methods such as grid-based search techniques [4] do not scale well beyond systems with few degrees of freedom while Numerical approaches [1] converge to locally optimal solutions. Sampling-based methods such as `RRT` [9], Expansive Spaces [6], and the `PDST` algorithm [8],

University of Nevada, Reno e-mail: rakhi@cse.unr.edu, mukesh@cse.unr.edu, fredh@cse.unr.edu

also fall under the category of approximate methods and are used for high dimensional systems.

This paper presents a technique to improve one of the most widely used kinodynamic sampling-based planners i.e. Rapidly exploring Random Trees (`RRT`) [9]. `RRT` is a popular planner for complex systems with geometric and differential contraints due to its capability of quickly exploring high-dimensional state spaces through sampling. The `RRT` algorithm makes use of an implicit Voronoi bias to evenly explore the state space. However, this Voronoi bias is no longer available if there is no good distance metric in the state space (that determines proximity of points in state-space in terms of time, or energy) or if drift and other dynamic constraints introduce undesired biases. For example, when using a Euclidean distance metric for systems such as Acrobot and Lunar Lander, the `RRT` growth is restricted to certain regions of the configuration-space, as illustrated by figures in Table 1. Uniform state space exploration of `RRT` is desired as it reduces the time to find solution trajectories. Typically, an `RRT` is grown for several thousand nodes for a system under consideration in order to find a solution path. If balanced coverage can be obtained by a relatively smaller sized tree, then we save computational cost and time.

The objective of the work presented here is to make improvements to the standard `RRT` algorithm to result in a tree of similar or smaller size that spans the state-space evenly. The focus of this paper is to address the issue of the lack of a good distance metric for physical systems that incorporates the effects of system dynamics and contraints when determining which states are closer. The contributions of this paper are as follows. The proposed approach provides a uniform state-space coverage for an `RRT` by computing the local exploration bias of the dynamic system at each point in the discretized state-space, and using this information to guide the expansion of the tree out of the biased region into least explored areas of the state-space. Experimental results indicate improvement in `RRT`'s state-space exploration for systems that exhibit a bias in coverage towards a specific direction in the state-space. However, it must be noted that this technique is only effective in situations when `RRTs` fail to evenly explore the state space. If `RRT` for systems grow uniformly in the state-space the proposed approach does not contribute towards further improvement of the state-space coverage. The contribution is unique in that no related work has employed localized approaches to obtain balanced `RRT` exploration.

2 Related Work

The research community has published a variery of algorithms [10] that enhance the performance of standard `RRT` algorithm by proposing modifications to decrease metric sensitivity, reduce the rate of failed expansion, control the sampling domain, guide tree expansion using local reachability information, bias sampling distributions to search in subspace of complete state space or goal region. There is limited literature [12, 5] that addresses the exploration performance of sampling-based algorithms. Li et. al's [12] work focused on using principal component analysis(PCA) globally to compensate for the undesirable biases introduced by a physical system's dynamic contraints on the exploration of an RRT algorithm. Their approach is composed of two steps: i) an offline learning procedure which constructs an RRT for the physical system and executes a PCA on the entire tree to represent the principal directions that the tree has expanded inside the task space; ii) altering the propagation step for RRT during the online operation by modifying the config-space coordinates of the random state sample in each iteration towards directions in which the variance is lower in the offline generated tree, and choosing the control which takes the system closer to this modified version of the random state sample. As a result, growth of the online tree is promoted towards the least explored direction in the task space. This algorthm has only been tested on a Three-link Acrobot and Car-like systems, and has motivated the authors of this paper to implement this technique on variety of systems to compare it's performance with the proposed algorithm.

Glassman and Tedrake's [5] work is based on control theory to derive an approximation to the exact minimum-time distance pseudometric by adapting the minimum time linear quadratic regulator (LQR) and it's associated cost-to-go function for affine systems. The proposed technique linearizes the system dynamics at the randomly sampled state space point in the RRT framework and defines a cost function based on time and effort which is used as the distance measure. The authors use a finite horizon affine quadratic regulator to compute the optimal cost-to-go functions of linearizations of the physical system for multiple time horizons to locally approximate the optimal distance measure. The proposed affine quadratic regulator-based (AQR) distance metric improves exploration of the state space of double integrator and simple pendulum but proves to be ineffective as the systems' nonlinearity and complexity increases such as the cartpole and torque limited 2-link Acrobot. However, the local-PCA based RRT approach presented in this paper focusses on complex non-linear systems.

3 Approach

The proposed technique is comprised of two steps - i) an offline step to learn the direction of propagation of state-space points sampled on a grid when system dynamics are simulated at these points, and ii) the alteration of the propagation step of basic RRT algorithm during the online construction of the tree. The offline step determines the principal components of the direction of propagation of state-space points on a grid of appropriate resolution, when numerous controls are applied to simulate the system dynamics at that point. These principal components represent the different propagation biases in different parts of the state-space. The online phase utilizes this information during the propagation step of building an RRT to reposition the random sample so as to compensate for the biases and even out the RRT's overall exploration of state-space. Henceforth, the proposed algorithm is referred to as LPCA-RRT.

Algorithm 1 Offline Step - Input: N, M, $\triangle t$

```
for $i = 1$ to $N$ do
    $x \leftarrow$ Sample_Grid_State();
    $u[] \leftarrow$ Sample_Random_Controls($M$);
    $s[] \leftarrow$ Simulate_New_States($x, u[], \triangle t$);
    $s'[] \leftarrow$ Transform($x, s[]$);
    $lpca_x \leftarrow$ PCA($s'_{subset}$);
end for
Return $lpca\_grid$
```

For the offline learning phase, state space points are sampled on a grid of appropriate resolution. Each sampled point represents a region, i.e. bin, in the state-space. For each sampled state space point, the system dynamics are simulated for a specific timestep for a certain amount of controls (i.e. between 50 to 250 depending on the system) to derive a set of new states. The coordinates of the new states are transformed such that the grid point serves as the new origin for these states. Principal Component Analysis (PCA) [7] is executed on a subset of the state-space coordinates of this set of new states. A subset of the state-space dimensions is considered for computational feasibility and also due to the fact that coverage of configuration space is desired as opposed to good coverage in state-space which comprises of derivates of the configuration space parameters. This PCA is referred to as the local PCA and is stored for each state-space point on the grid.

Algorithm 1 summarizes the offline step of the proposed approach, where N represents the total number of states sampled on the grid. N is determined by the grid bounds, grid resolution, and the dimension of state-space. M denotes the number of controls. M varies from system to system and is experimented with values starting from 50 going up to $50,000$ at increments of 100 to determine at what value does the local PCA converge. The algorithm returns local PCAs for points sampled on the entire grid.

The pseudo-code for the online phase is provided in Algorithm 2. The basic `RRT` algorithm is adapted from [9] where at each iteration a random state x_{rand} is sampled from the state space. For construction of the `RRT`, the node x_{near} on the tree which is nearest to x_{rand} is selected for expansion. A Euclidean distance metric is used to determine the nearest neighbor along the tree. The coordinates of x_{near} are evaluated to calculate the bin from the offline state-space grid that this node belongs to. The offline generated local PCA is then retrieved for the state x_{near}. Since this local PCA is representative of a bin from the offline state-space grid therefore it is only an approximate representation of the direction of propagation of the tree from x_{near}. The configuration space coordinates of the randomly sampled state-space point at the corresponding iteration of `RRT` are then modified to position the random sample in the direction of the least significant components of this local PCA. The controls that extend the tree from the selected node closer to the altered random sample state are chosen for propagation of the tree thereby leading the `RRT` out of the regions, where it would have originally been focussed, into unexplored areas of the state space.

Algorithm 2 Online Step - Input: $x_{init}, N, lpca\_grid$

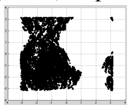
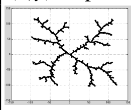
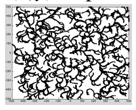
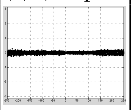
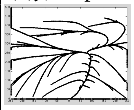
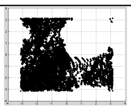
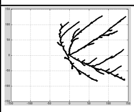
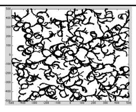
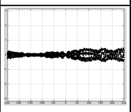
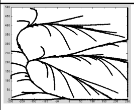
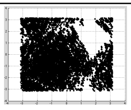
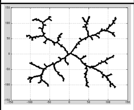
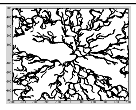
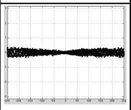
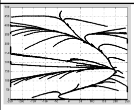
```
$Tree$.init($x_{init}$)
for $i = 1$ to $N$ do
    $x_{rand} \leftarrow$ Sample_Random_State();
    $x_{near} \leftarrow$ Determine_Nearest_Neighbor($x_{rand}, Tree$);
    $bin_{x_{near}} \leftarrow$ Evaluate_Bin($x_{near}, lpca\_grid$);
    $lpca_{x_{near}} \leftarrow$ Retrieve_PCA($bin_{x_{near}}, lpca\_grid$);
    $x'_{rand} \leftarrow$ Modify( $inv(lpca_{x_{near}}), x_{rand}$ );
    $[x_{new}, x_{edge}] \leftarrow$ Propagate($x_{near}, x'_{rand}$ );
    $Tree.Vertex\_Add(x_{new})$
    $Tree.Edge\_Add(x_{near}, x_{edge})$
end for
Return $Tree$

Modify( $inv(lpca_{x_{near}}), x_{rand}$ )
$x_{adj} \leftarrow inv(lpca_{x_{near}}) * x_{rand}$;
for $i = 1$ to $n$ do
    $x'(i)_{adj} \leftarrow \frac{l_1}{l_i} * x(i)_{adj}$;
end for
$x'_{rand} \leftarrow lpca_{x_{near}} * x'_{adj}$;
Return $x'_{rand}$
```

The algorithm then propagates the selected node x_{near} by applying m random controls to obtain new states. The closest new state to x'_{rand}, denoted by x_{new}, and the corresponding control are selected for the expansion step of the algorithm.

4 Experiments

The algorithm was tested on various systems - Three-link Acrobot, Car-like system, Cart Pole, Hovercraft, and Lunar Lander. The results were compared against the basic RRT, and Li et al.'s algorithm henceforth referred to as GPCA−RRT. Performance of these algorithms was measured in terms of the percentage of bins populated by the generated tree on the discretized subset of state-space and execution time, measured in seconds. Trees were grown for sizes spanning from 1000 to 20,000 nodes and the performance results represent an average of ten test runs. The algorithms were implemented in Octave 3.0.5 and executed on the UNR Research Grid. The implementation stores RRT in an array and uses linear search for nearest neighbor search, hence recorded processing times are higher. Therefore, the authors would like to emphasize that this is just a proof-of-concept implementation. For GPCA−RRT, the experiment used the global PCA of the standard RRT of the same size i.e. an RRT was grown for N nodes, the global PCA was computed for this RRT and was used to generate the GPCA−RRT of size N nodes. The figures in Table 1 display a projection of the trees, plotted for various systems, in those configuration space parameters for which the tree exploration was not uniform.

Table 1 Configuration-Space Coverage Plots for trees grown for 20,000 nodes

Plots for →	3-Link Acrobot (θ_1, θ_2) c-space	Hovercraft (x, y) c-space	Car-like System (x, y) c-space	Cart Pole (x, θ) c-space	Lunar Lander (x, y) c-space
RRT					
GPCA					
LPCA					

Three-Link Acrobot: The Acrobot was simulated in Passive-Active-Active mode with torque applied at active joints. The angles θ_i are relative to the global reference frame and do not correspond to the angles between consecutive links. As indicated by Table 2, `LPCA-RRT` outperforms both algorithms by $15-20\%$ in terms of coverage at the expense of spending an average of 2.5% more in time. Moreover, it was observed that the tree generated by `LPCA-RRT` for 5000 nodes covered the config-space more uniformly than `RRT` grown for 20,000 nodes.

		1000 Nodes	3000 Nodes	5000 Nodes	20000 Nodes
RRT	Populated Bins	644	1691	2583	8536
	Time	28.61	203.41	510.59	28210
GPCA	Populated Bins	693	1893	2998	9800
	Time	28.70	202.97	510.98	28471
LPCA	Populated Bins	722	2015	3170	11285
	Time	32.82	209.44	519.30	29329

Table 2 Three-link Acrobot Results: $(\theta_1, \theta_2, \theta_3)$ config-space is divided into 50x50x50 bins to measure coverage

Car-like System: `LPCA-RRT` causes the car to move straight with less turns as in the case of `RRT` or `GPCA-RRT`. Results indicated that neither `GPCA-RRT` nor `LPCA-RRT` provide an improvement in terms of coverage for this system. Results for coverage have not been listed due to space contraints.

Cart Pole: Experiments on Cart Pole system showed that `LPCA-RRT` resulted in an average improvement of 35% in coverage with only 1% increase in time to grow the tree of same size. Space contraints prohibit the authors from sharing the results. From the coverage plots in Table 2, it can be seen that the exploration of θ space (plotted along y-axis) was improved for both `LPCA-RRT` and `GPCA-RRT`.

Hovercraft: `LPCA-RRT` demonstrated a uniform coverage of $(x, y,)$ space as compared to `RRT`. `GPCA-RRT` tends to skew the growth of RRT towards the upward left direction, which is the principal direction of variance represented by the global PCA computed for the basic RRT algorithm. As per Table 3, coverage of LPCA-RRT improved by an average of 25% with an average of 1.5% increase in time cost.

		1000 Nodes	3000 Nodes	5000 Nodes	20000 Nodes
RRT	Populated Bins	530	1581	2588	9969
	Time	19.68	125.11	316.20	4001
GPCA	Populated Bins	477	1113	1772	9234
	Time	21.79	127.48	317.59	4261
LPCA	Populated Bins	568	1655	2704	9875
	Time	53.28	226.26	481.69	4944

Table 3 Hovercraft Results: (x, y, θ) config-space is divided into 50x50x50 bins to measure coverage in terms of number of populated bins

Lunar Lander: The system was simulated in Ascent mode. For RRT and GPCA-RRT, the branches of the tree tend to grow downwards, however LPCA-RRT promotes the growth of the tree sideways. Results from Table 4, show that LPCA-RRT provided an improvement in coverage by 25% at the cost of 4% increase in computational time.

		1000 Nodes	3000 Nodes	5000 Nodes	20000 Nodes
RRT	Populated Bins	361	1089.5	1483	5215
	Time	17.249	112.79	313.281	4231
GPCA	Populated Bins	323	1104	1773	6501
	Time	17.344	113.88	313.289	4247
LPCA	Populated Bins	555	1374	1800	6672
	Time	23.375	133.54	324.123	4398

Table 4 Lunar Lander Results: $(x,\ y,\ \theta)$ config-space is divided into 50x50x50 bins to measure coverage

5 Conclusion

The proposed technique improves RRT exploration by learning the local effects of constraints in a physical system during an offline phase and then counteracts these effects that inhibit the uniform growth of the RRT during the online operation of RRT by adapting the propagation step. This work executes PCA locally and enables better approximation of the underlying non-linear bias by decomposing the state-space into regions where the bias may be varying. The approach is tested on various systems and experimental results demonstrate that this technique works on systems that exhibit a consistent exploration bias regardless of the size of the tree, and that exploration performance is improved by $15-35\%$ thereby reducing the cost of finding a solution to specific motion planning queries. One key conclusion is that the algorithm is only effective in scenarios where the standard `RRT` algorithm fails to uniformly and quickly explore the state-space, as seen in the case of Car-like system. Overall, this technique compensates for the lack of good distance metric in the state-space and can be adopted by various sampling-based planning algorithms.

Acknowledgements

The authors would like to thank their colleagues Yanbo Li and Kostas Bekris for providing guidance and assisting with replicting results from [12].

References

[1] Betts, J.T.: Survey of numerical methods for trajectory optimization. AIAA Journal of Guidance, Control and Dynamics **21**(2), 193–207 (1998)
[2] Canny, J., Rege, A., Reif, J.: An exact algorithm for kinodynamic planning in the plane. Discrete and Computational Geometry **6**, 461–484 (1991)
[3] Choset, H.M.: Principles of robot motion: theory, algorithms, and implementation Intelligent robotics and autonomous agents. MIT Press (2005)
[4] Donald, B., Xavier, P., Canny, J., Reif, J.: Kinodynamic motion planning. J. ACM **40**(5), 1048–1066 (1993)
[5] Glassman, E.L.: A quadratic regulator-based heuristic for rapidly exploring state space. Master's thesis, Massachusetts Institute of Technology (2010)
[6] Hsu, D., Kindel, R., Latombe, J.C., Rock, S.: Randomized kinodynamic motion planning with moving obstacles (2000)
[7] Jolliffe, I.: Principal Component Analysis, second edn. Springer Series in Statistics. Springer (2010)
[8] Ladd, A.M., Kavraki, L.E.: Motion planning in the presence of drift, underactuation and discrete system changes. In: Robotics: Science and Systems I, pp. 233–241. MIT Press (2005)
[9] LaValle, S., Kuffner, J.: Rapidly exploring random trees: Progress and prospects. In: WAFR, pp. 293–308 (2001)
[10] LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
[11] Lewis, F.L.: Applied optimal control & estimation: digital design & implementation. Prentice Hall (1992)
[12] Li, Y., Bekris, K.E.: Balancing state-space coverage in planning with dynamics. In: IEEE International Conference on Robotics and Automatio (ICRA), pp. 3246–3253 (2010)
[13] Schwartz, J.T., Sharir, M.: On the piano movers' problem: Ii. general techniqies for computing topological properties of algebraic manifolds. Communications on Pure and Applied Mathematics **36**, 345–398 (1983)