Balancing State-Space Coverage in Planning with Dynamics

Yanbo Li and Kostas E. Bekris

***Abstract*— Sampling-based kinodynamic planners, such as the popular RRT algorithm, have been proposed as promising solutions to planning for systems with dynamics. Nevertheless, complex systems often raise significant challenges. In particular, the state-space exploration of sampling-based tree planners can be heavily biased towards a specific direction due to the presence of dynamics and underactuation. The premise of this paper is that it is possible to use statistical tools to learn quickly the effects of the constraints in the algorithm's state-space exploration during a training session. Then during the online operation of the algorithm, this information can be utilized so as to counter the undesirable bias due to the dynamics by appropriately adapting the control propagation step. The resulting method achieves a more balanced exploration of the state-space, resulting in faster solutions to planning challenges. The paper provides proof of concept experiments comparing against and improving upon the standard RRT using MATLAB simulations for (a) swinging up different versions of a 3-link Acrobot system with dynamics and (b) a second-order car-like system with significant drift.**

I. INTRODUCTION

Many interesting physical systems exhibit challenging dynamics and underactuation that complicate motion planning. One prototypical problem in this context is swinging up an Acrobot system in the presence of gravity [1], [2], as displayed in Figure 1. This is an important benchmark because many classes of robots, especially walking robots, exhibit similar levels of underactuation and dynamic constraints.

One approach for planning with dynamics is the framework of sampling-based kinodynamic algorithms, such as the popular RRT method [3], [4] and alternatives [5], [6]. These methods aim to cover as quickly as possible the state space through sampling. Specifically the RRT algorithm makes use of an implicit Voronoi bias when selecting states and controls to expand a reachability tree into the state space. This favorable bias, however, is no longer available if there is no good distance metric in the state space or if drift and other dynamic constraints introduce undesired biases. Figure 2 illustrates the challenges in planning with the standard RRT for the Acrobot. The figure provides the six-dimensional states (three angles and three angular velocities) along the reachability tree produced by RRT after 50,000 iterations. The states are projected onto the torus defined by the orientations of the Acrobot's first two links. The state-space exploration is severely biased due to the dynamics and the vicinity of the goal (swinging up the Acrobot: $\theta_1 = \frac{\pi}{2}$, $\theta_2 = \frac{\pi}{2}$) has not even been approached yet.

Yanbo Li and Kostas E. Bekris are with the Computer Science and Engineering department, University of Nevada-Reno, Reno, NV, 89557 email: liy19@unr.nevada.edu, bekris@cse.unr.edu

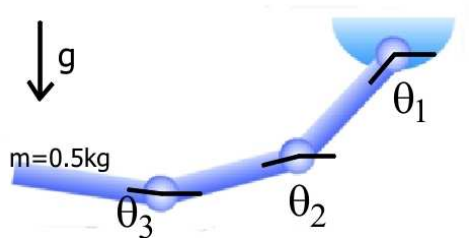

Fig. 1. A 3-link Acrobot. The controls are the joint torques. θ_1, θ_2 and θ_3 are the global orientations of the links. The state of the system also includes angular velocities. In this paper's experiments each link has 0.5Kg weight.

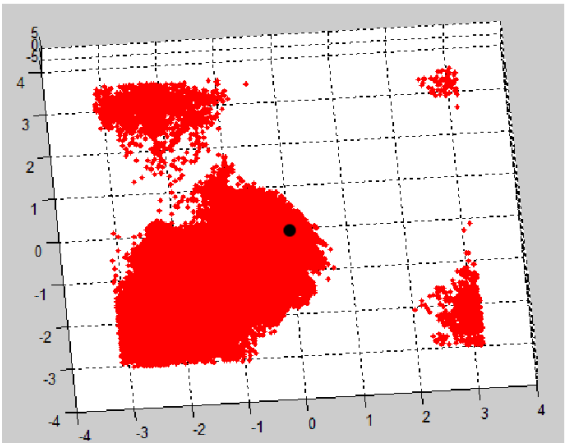

Fig. 2. A projection of the states explored by the standard RRT algorithm after 50,000 iterations to a torus that corresponds to θ_1 and θ_2. The black dot corresponds to the initial state $(0, 0, 0, 0, 0, 0)$, where all the links face horizontally to the right and have zero angular velocity. Due to dynamics, including gravity, the exploration is heavily and undesirably biased.

The main ideas in this paper is to (a) learn the biases introduced by the dynamics in the state space exploration process during an offline training phase and then (b) counteract the effects of these undesired biases during the online operation of the algorithm so as to achieve a more balanced coverage of the state space. As a feasibility study of these ideas, this paper utilizes a classical statistical tool for building predictive models, that of Principal Component Analysis (PCA) [7], [8], [9]. Even though PCA is a linear method and the Acrobot is a non-linear system, it is still possible to acquire useful information about the exploration bias because the system is highly constrained. During the online step, the exploration procedure is modified so as to promote the propagation of the search tree towards the direction of the least significant components. In the context of the RRT, this can be achieved by selecting at each iteration the control which brings the system closer to a modified version of the random state sample.

This paper provides indications that the above procedure is beneficial when planning for highly constrained systems. Experimental results suggest that it is possible to solve planning problems for the Acrobot, as well as for a car-like system, faster, since the state space is covered more efficiently. At the same time, the computational overhead during the online operation of the algorithm is minimal. On average, each iteration of the PCA-based algorithm is only 1.15% more time consuming than an iteration of the standard RRT. Only a few matrix transformations and operations have to be executed to achieve the desired result. Concerning the cost of the offline training procedure there are two issues:

- How large should the trees be during the offline process?
- How many trees should be propagated during training?

Experiments indicate that a small number, even just one, of small size trees is sufficient. Overall, the experiments show that the combined cost of the offline step and the online step is smaller than the cost of a solution with the standard RRT.

An important advantage of the methodology is that it does not heavily depend upon the underlying system. As sampling-based planners are general methods, similarly the proposed approach does not depend on the properties of specific systems. Nevertheless, the more constrained the system is, the more beneficial the modification is expected to be. If the system does not suffer from any bias, then the technique is automatically equivalent to a standard RRT.

Furthermore, while the idea is described in the context of the RRT, it could be also paired with alternative methods. In particular, there are techniques for adapting the sampling process in sampling-based tree planners [10], [11] and methods for producing motion primitives [12], [13]. The current work is complementary to many of these existing methodologies.

Another important advantage of the proposed method is that it is not necessary to study the coverage of the complete state space but instead to focus on a task space that is important for the problem's solution. For instance, the dynamical version of the Acrobot has a six-dimensional state space but it is fairly straightforward to apply the PCA on a subspace. For example, if the goal is specified only in terms of the configuration that has to be achieved (i.e., the three orientations of the links so that the Acrobot is vertical), then PCA can be run only in the configuration space (C-space) and not the complete state space, since this is the space that it is important to be covered to solve the problem.

Last but not least, this work, as a proof of concept, opens the door to many exciting extensions, such as the online computation of the principal components and the online adaptation of the exploration procedure. It can also lead to a hierarchical and local learning of the principal components so as to deal with different biases in different parts of the state space. A challenging variant of the method is to automatically select a projection of the complete state space where the PCA should be executed. It is similarly interesting to study the application of non-linear dimensionality reduction [14], more complex models of dynamical systems that make use of physics-based simulation, as well as the effects of obstacles and collisions to the computation of the biases.

II. RELATED WORK

Direct or kinodynamic planning searches directly the entire state space ($\mathcal{X}$) of a dynamical system. There are many ways to approach such problems in the related literature:

- Optimal control can be applied for direct planning [15] but handles only simple systems. Algebraic solutions are known only for 2D point masses [16], [17].
- Numerical optimization [18] is expensive and suffers from local minima.
- Search-based methods compute optimal paths with grid-based approximations in $\mathcal{X}$ but depend exponentially to the resolution [19], [20].

A polynomial-time, search-based approximation algorithm solves the problem efficiently for a point mass with dynamics [21] and extends to more complicated systems [22], [23]. Sampling-based tree planners, such as RRT [3], [4], Expansive Spaces [5], and the PDST algorithm [6], can be seen as extensions of such search-based methods that employ heuristics to evenly explore $\mathcal{X}$.

These methods follow a selection-propagation scheme to construct the tree data structure in the state space. At each iteration they first select a node/state along the tree that is already connected with a path to the start state. Then they apply a control from the selected state. In particular, the RRT algorithm selects the initial state for expansion (x_{near}) by randomly sampling a state x_{rand} and then finding the closest state along the tree to x_{rand}. If the metric used for finding the closest state is appropriate and if the propagation step is not biased then the algorithm has a Voronoi bias. This implies that the larger unexplored sections of the state space have higher probability of being explored.

There are also bidirectional versions of the algorithm that considerably improve performance [24]. The bidirectional RRT, however, requires a "steering method" to connect two specific states or a way to address gaps in the merging process of the two trees, which for certain systems is possible to achieve [25], [26]. Nevertheless, the application of the bidirectional tree approach is challenging for the type of systems considered in this paper.

Alternative schemes to RRT use different mechanisms to implement the selection-propagation step. For example, the PDST algorithm uses an adaptive subdivision scheme of $\mathcal{X}$ and a scoring system over edges for guaranteeing probabilistic completeness. This is an approach that has been tested on the Acrobot system [27].

Many recent methods focus on applying RRT-like solutions to challenging problems that involve manipulators or grasping, or high-dimensional systems, or underactuated and dynamical systems [28], [29], [30], [31], [32], [33], [34]. Many of these techniques make use of the idea that the search can be focused in a subspace of the complete state space [28], [35], [32], [34], [33]. Then this subspace corresponds to the task space, the space defined by the specific task and goal that has to be achieved. Recent work applies task-space control tools to the problem of planning for the Acrobot [29], [28], and together with the work on PDST, it has motivated the focus of this paper on this specific benchmark.

There have also been previous efforts to utilize tools such as Principal Component Analysis (PCA) in motion planning from which this work is inspired. The idea has been to use PCA in an online fashion in order to accelerate the search procedure of RRT-like algorithms once the tree has already reached the narrow passage [36]. In this online PCA approach for geometric planning, the idea is to bias the exploration of the tree so as to favour the directions in which the variance of the growing tree is high. There have also been efforts that utilize PCA in the study of protein motion [37], [38]. To the best of the authors' knowledge there has not been an application of PCA in motion planning so as to express the effects of dynamic constraints in the exploration performance of sampling-based algorithms.

III. USING PCA TO IMPROVE COVERAGE

The approach proposed in this paper is split into two steps, an offline learning procedure and a modification of the online operation of the RRT algorithm. For the offline part, the approach first applies a standard sampling-based algorithm, such as `RRT-Connect`, to grow a tree with a certain number of nodes. Once the desired tree has been expanded, a PCA is executed on the coordinates of all the nodes/states to calculate a new basis. This basis can be used to represent the principal directions that the tree has expanded inside the state space. For the online part, the approach alters the random state (x_{rand}) towards which the tree is extended. The modification encourages the tree to favor directions in which the variance is low in the offline tree.

A. Principal Component Analysis and Offline Step

Principal Component Analysis (PCA) is a statistical tool to investigate the underlying dimensions of a dataset [7], [8]. It provides a transformation from a number of possible interrelated data to a smaller number of independent variables, which are called principal components. The first component found accounts for the direction of the highest variance in the dataset. Subsequent components account for decreasing levels of variance. PCA returns a new basis for the original data as well as the corresponding variance in each direction. These correspond to the eigenvectors and the eigenvalues produced by the method respectively. PCA assumes that the underlying operation of the system can be expressed by a linear process.

In the general case PCA operates over a matrix/dataset M with dimension $n \times d$, where each observation of d values corresponds to a row. There are n total observations. In the first step, the basic version of the algorithm subtracts the mean from each dimension of the data. Then, it computes the covariance matrix C with dimension $d \times d$ and its eigenvectors b, as well as the corresponding eigenvalues l. Finally, the method transforms each observation data point by the new basis b. In this work, the observations correspond to n states extracted from the tree computed by `RRT-Connect`. These states correspond to the last state of each propagation step in the algorithm. For the Acrobot system the dimensionality of each state is 6, the three joint angles and three angular velocities. For the current implementation only the orientations of the three links are used for each state, since the modification is applied only in the configuration space where the goal state and the metric is defined. Algorithm 1 summarizes the offline step of the proposed approach.

Algorithm 1 Offline Step - INPUT: n

$T \leftarrow$ `RRT-Connect`(n)
$M \leftarrow$ coordinates of T's nodes
$(b, l) \leftarrow$ PCA(M)

B. Online Step

For the online step, the pseudo-code is provided in Algorithm 2. As it can be seen, the proposed approach follows the basic structure of the RRT algorithm but modifies the sampled state x_{rand} before calling the Extend step.

Algorithm 2 Online Step - Input: $x_0 K, b, l$

T.init(x_0)
for $i = 1$ *to* K **do**
  $x_{rand} \leftarrow$ sample $\mathcal{X}$
  $x_{near} \leftarrow$ find nearest $x \in T$ to x_{rand}
  $x'_{rand} \leftarrow$ Transform($inv(b), x_{rand}$)
  $x'_{new} \leftarrow$ Modify(l, x'_{rand})
  $x_{new} \leftarrow$ Transform(b, x'_{new})
  $x_{edge} \leftarrow$ Extend(x_{near}, x_{new})
  T.AddVertex(x_{edge})
  T.AddEdge(x_{near}, x_{edge})

The algorithm first samples a random state x_{rand} in the state space $\mathcal{X}$. Then the nearest state along the tree is found according to a metric $\rho(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$. Once the nearest neighbor is found, then the coordinates of the randomly sampled state are transformed into the basis b computed by the offline part. Then for each principle component, the algorithm modifies the corresponding coordinate according to the eigenvalues of the corresponding matrix.

More specifically the function "Modify" executes the following steps. Let $l_1 > \ldots > l_n > 0$ be the eigenvalues of the covariance matrix, and $b_1, \ldots, b_n$ the corresponding eigenvectors. The coordinates of the adjusted random state x'_{new} is given by:

$$\forall i \in [1, n], x'^{i}_{new} = \frac{l_1}{l_i} x'(i)_{rand} \quad (1)$$

For each principal dimension, the algorithm rescales the sample coordinate. The adjustments keep the direction with the most variance unchanged, while it projects low variance directions outwards depending on their eigenvalues. The next step transforms x'_{new} back into the coordinates of the original state space $\mathcal{X}$, resulting to the state x_{new}, and then attempts to connect the nearest node x_{near} to the modified sample x_{new} by calling function Extend.

Algorithm 3 details the operation of Extend. The function tests k random controls from the state x_{near} and integrates

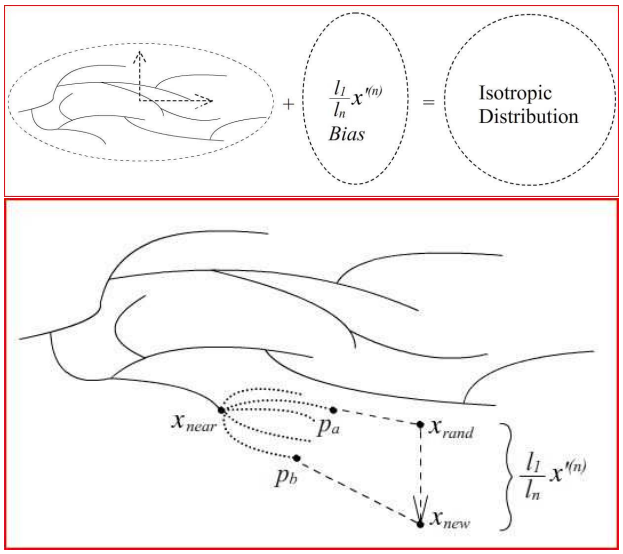

Fig. 3. Trees created through sampling in highly constrained spaces tend to explore certain directions in the state space much more than others. Algorithm 2 alters the sampling process so as to bias the expansion of the tree towards the least explored dimensions.

them for time t. The resulting state of each integration is denoted as x_{ext}. For each x_{ext} the function computes the distance to x_{new}. The closest x_{ext} to x_{new} and the corresponding control are selected for the expansion step of the algorithm.

Algorithm 3 Extend(x_{near}, x_{new})

$\rho_{min} \leftarrow \infty$
for $i = 1$ *to* k **do**
  $u_{rand} \leftarrow$ sample $\mathcal{U}$
  $x_{ext} \leftarrow$ Integrate(x_{near}, u_{rand}, t)
  $\rho \leftarrow \rho(x_{ext}, x_{new})$
  if $\rho_{min} > \rho$ **then**
    $\rho_{min} = \rho$
    $x_{new} = x_{ext}$
return x_{new}

Figure 3 illustrates the difference in the growth of a sampling-based tree data structure computed by a standard RRT in highly-constrained spaces and the effects of the proposed version of RRT that uses PCA. The oval in the figure illustrates the probability of growth along different directions. The dynamic constraints of real systems, such as the Acrobot, distort the expected isotropic distribution of the state-space exploration into an ellipsoidal one. The PCA adjustment proposed here biases the expansion of the tree towards the principal dimension that the standard RRT algorithm would least explore.

IV. EXPERIMENTAL RESULTS

A. Acrobot Model and Setup

The algorithm is tested on a 3-link Acrobot robot system, shown in Figure 1. All joints of the robot can be both actuated and passive. The control input corresponds to torques at

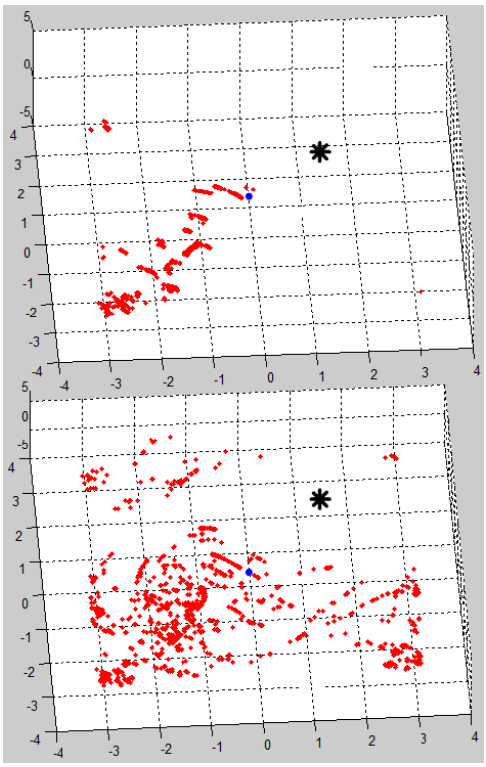

Fig. 4. (up) The nodes/configurations reached by the RRT algorithm after 1000 iterations for an APP Acrobot. (down) The nodes/configurations reached by the proposed PCA-based variation after 1000 iterations.

the joints. Each link has 0.5m in length and 0.5 kilograms in mass and there is an assumption of uniform density. There are no angle limits but the maximum absolute value for the torque at each joint is 20 Nm, unless specified otherwise. Tests were conducted for different modes of the system. For example, all joints can be actuated, a mode that is referred to as AAA. The mode APP indicates that only the first joint, that is connected to the fixed base, is actuated and the subsequent two are passive. Consequently, this is a highly underactuated mode. Similar naming conventions apply to AAP, PAA and PPA.

For the distance measure the implementation ignored the effect of velocities, and the distance between two states is defined to be equal to the distance between two configurations. The topology of the configuration space for the Acrobot is a three dimensional torus, corresponding to the three joint angles. The distance measure between two configurations is defined as the sum of the distances of the corresponding angles. The distance between two angles θ_i and θ'_i for the i-th joint is defined as:

$$d(\theta_i, \theta'_i) = min_{\forall k,k'} |2k\pi + \theta_i, 2k'\pi + \theta'_i|$$

where k, k' are integers. Consequently, the overall distance metric between two states $x = (\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3)$ and $x' = (\theta'_1, \theta'_2, \theta'_3, \dot{\theta}'_1, \dot{\theta}'_2, \dot{\theta}'_3)$ = is:

$$\rho(x, x') = \sum_{i=1}^{3} d(\theta_i, \theta'_i).$$

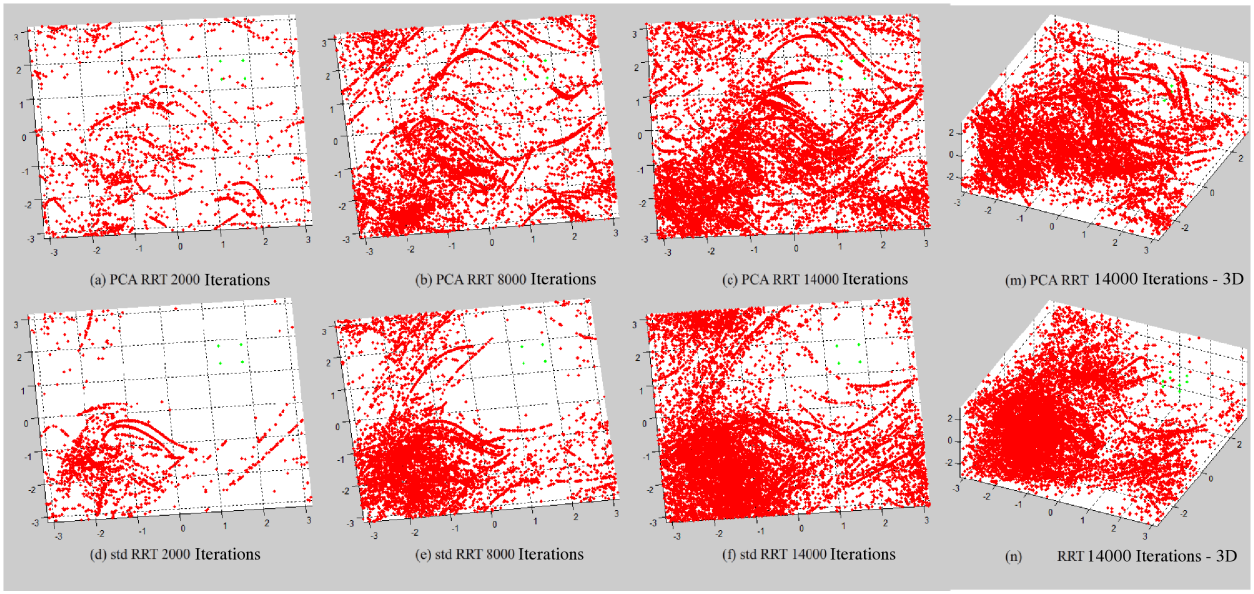

Fig. 5. APP mode: (up) The coverage of the configuration space by the PCA -based approach for incrementally larger number of iterations. (down) The standard RRT. The last column corresponds to a view from a different 3D viewpoint for both algorithms.

Unless otherwise specified, the figures in this paper display a projection of the trees in the C-space ($q = (\theta_1, \theta_2, \theta_3)$). Note that the angles θ_i are relative the global reference frame and do not correspond to the angles between consecutive links. In the configuration $(0, 0, 0)$ all the links of the Acrobot are horizontal, pointing towards the right. An Acrobot in the up-swing configuration has a configuration of $(\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2})$. Furthermore, the figures display only the nodes of the resulting trees. For the Extend function, the values $k = 10$ and $t = 0.2sec$ were used in the experiments. The algorithms have been implemented on MATLAB and tested on a machine with Intel 2.0 Dual Core CPU and 2GB RAM.

B. Evaluation

Coverage in APP mode: The algorithms were tested on different modes of the Acrobot starting with the most underactuated case (APP mode). Figure 4(up) shows the performance of RRT-Connect with 1000 nodes, where the exploration is severely constrained to one direction. In Figure 4(down), the exploration is better distributed when using PCA for the same number of nodes (1000). Figure 5 provides an incremental comparison between the standard RRT and the PCA-based solution. Moreover, the computational overhead for the transformation and modification step is very low and does not exceed on average 1.15% of the time spent by the standard RRT. The table below shows the average duration and the associated variance of building trees with 1000 nodes with the standard RRT and with the PCA-based RRT, over 100 experiments on MATLAB.

Computation time	Average duration	Variance
Standard RRT (seconds)	5.731	0.13
PCA- based RRT (seconds)	5.797	0.137

TABLE I

TIME TO COMPUTE A TREE WITH 1000 NODES IN SECONDS.

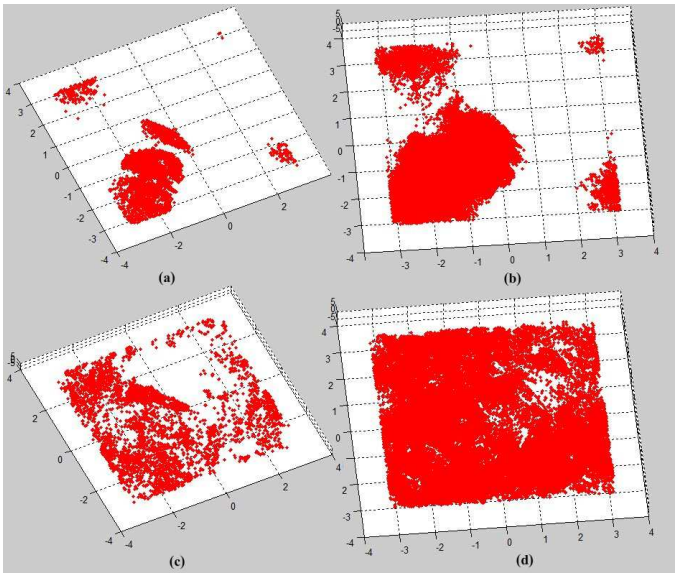

Fig. 6. AAA mode: (up) Coverage for the standard RRT algorithm along the θ_1 and θ_2 orientations after 5,000 nodes (left) and after 50,000 nodes (right) - (down) Similar results after the PCA-based modification.

Coverage in AAA mode: The second test case involved the Acrobot in AAA mode. The results are shown in Figure 6, where the top figures correspond to the coverage performance of the standard RRT and the bottom ones to the PCA-based modification. As it can be seen, after 50,000 nodes the proposed modification manages to cover most of this projection of $\mathcal{X}$.

Comparison: Table II is a summary of the above and similar experiments. It displays the number of nodes necessary for different algorithms and different modes of the Acrobot to solve a planning challenge. The planning problem required

	Standard RRT (iterations)	PCA-based RRT (online iterations)
AAA	46.8 K	17.3 K
AAP	52.1 K	26.3 K
APP	Failed (>120,000)	96.9 K

TABLE II

AVERAGE PERFORMANCE: STANDARD RRTVS. PCA-BASED RRT.

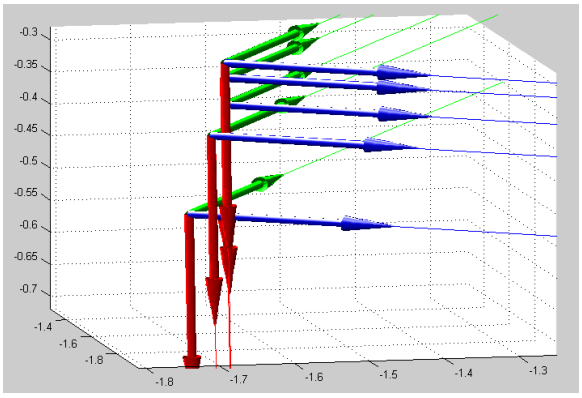

Fig. 7. The three principal components for different sizes of the same tree as it expands using the standard RRT algorithm.

the robot to move from an initial configuration of $(0, 0, 0)$ with zero velocities, to a vertical position of $(\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2})$, denoted by the black asterisk in Figure 4. The goal was considered achieved regardless of the velocity of the system at the vertical position and if the planner was able to produce a configuration within five degrees of all the angles. As figure 6 exhibits, this is a hard problem for sampling-based planners as it requires extensive exploration in order to cover the space. The part of the configuration space which corresponds to the goal is the last to be reached by the algorithm. For the PCA-based solution, an offline tree was extended using the standard RRT algorithm for 2,000 iterations.

Note that for these problems, the more the links of the acrobat weigh the more difficult the problem is. For the AAA mode, the absolute value of the input torque was limited to 20Nm, while for the APP mode it was limited to 30Nm. For the AAP mode, the first joint has a limit of 30Nm and the second one has a limit of 20Nm. As the table shows, more difficult problems (i.e., APP is much harder than AAA) require more iterations to be solved. In every case, however, the PCA-based solution performs always better than the standard RRT even if the offline cost is included to the cost of the PCA-based solution (2,000 iterations). This provides some evidence that we can take advantage of such offline information to considerably improve state-space coverage performance.

How large offline trees are needed? Experiments indicate that it is possible to grasp the effects of the dynamic constraints at the beginning of the exploration process with a small number of nodes. In order to investigate this issue, the following experiment was executed. A tree was expanded using the RRT algorithm up to 5,000 iterations. Every 1,000 iterations the PCA algorithm was executed to compute the principal components given all the nodes produced up to that point. Figure 7 visualizes the principal

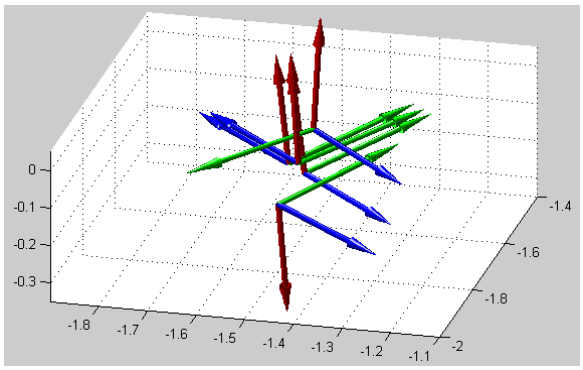

Fig. 8. The directions of the principal component after running PCA on different trees computed with the standard RRT(blue for least principal component). In all of the above cases, the vector for the least principal component points in a similar or in a negative direction. Overall, there is not significant deviation between the bias detected by the algorithm.

components from this experiment. Red lines indicate the first principle components for each PCA run. Green and blue lines indicate the rest of the principal components. While the mean of the nodes changes as the tree grows, the direction of the eigenvectors does not change significantly. And even the mean changes only along the direction of the principal component, which does not effect the proposed algorithm. Similar results have been achieved over multiple experiments, which due to space limitations are not presented here. Overall, there are indications that for dynamically constrained systems a small exploration tree may be sufficient to grasp the major constraint influence.

How many offline trees are needed? There are also indications that running PCA over a small number of trees offline is enough to grasp the principal components. Figure 8 shows an example with the principal component computed for different trees using the standard RRT on the constrained version of the problem (APP). The resulting vectors are quite similar meaning that the differences on the online operation of the technique would not be significant. Similar results are acquired for experiments with a much larger number of trees than the one depicted on Figure 8. Furthermore, note that in all of the previous experiments and comparisons only one offline tree was used during the training session. This was sufficient in order to get the improved coverage and performance in planning.

C. Second System: Car-Like Vehicle

The proposed approach has also been tested on a different system other than the Acrobot. In particular, a second-order car-like system has been chosen with state-update equations:

$$\begin{aligned}
\dot{x} &= w\ cos\zeta\ cos\theta & x &\in [-150, 150] \\
\dot{y} &= w\ cos\zeta\ sin\theta & y &\in [-150, 150] \\
\dot{\theta} &= w\ sin\zeta & \theta &\in [-\pi, \pi] \\
\dot{w} &= u_1 & w &\in [0, 4] \\
\dot{\zeta} &= u_2 & \zeta &\in [-\frac{\pi}{6}, \frac{\pi}{6}]
\end{aligned}$$

$w = 0$	500 nodes	1K nodes	3K nodes	5K nodes
Standard RRT	0.01887	0.045	0.22328	0.53474
PCA on same size	0.01453	0.04786	0.12898	0.28111
PCA on 1K nodes			0.10724	0.21055
$w = 2$	500 nodes	1K nodes	3K nodes	5K nodes
Standard RRT	0.01368	0.03482	0.21007	0.49757
PCA on same size	0.0137	0.02837	0.12754	0.32466
PCA on 1K nodes			0.13136	0.26946
$w = 4$	500 nodes	1K nodes	3K nodes	5K nodes
Standard RRT	0.012807	0.03034	0.17893	0.44894
PCA on same size	0.01077	0.02698	0.12814	0.32617
PCA on 1K nodes			0.13404	0.32699

TABLE III

VARIANCE OF DENSITY IN CELLS IN THE C-SPACE.

where u_1 and u_2 are the control inputs (acceleration and derivative of the steering angle correspondingly), x, y are the Cartesian coordinates of a point on the robot, θ is orientation, w is the forward velocity and ζ the steering angle. The bounds for the state parameters are also provided in the above equation, from which it becomes apparent that the car can only move forward. The bounds for the control parameters are as follows: $-0.03 \leq u_1 \leq 0.03$ and $-0.06 \leq u_2 \leq 0.06$. This is a system that is not small-time locally controllable everywhere and given the above bounds has significant drift.

The distance measure used was the following: $s_1 * (x_1 - x_2) + s_2 * (y_1 - y_2) + s_3 * (\theta_1 - \theta_2) + s_4 * (w_1 - w_2) + s_5 * (\zeta_1 - \zeta_2)$, where s_i is a scaling factor for each state parameter and depends on the bounds for the corresponding parameters ($s_1 = s_2 = \frac{1}{300}, s_3 = \frac{1}{2\pi}, s_4 = \frac{1}{4}, s_5 = \frac{\pi}{3}$). As with the Acrobot, special care has to be taken when considering the difference in orientation. The PCA was run on a projection of the states onto the C-space of the system, which in this case corresponds to (x, y, θ).

Figure 9 provides a comparison of the different kind of trees that result after the application of the standard RRT and the PCA-based RRT on the car-like system. Both algorithms were applied for 2,000 iterations. The initial state corresponds to the car-like system having its maximum velocity ($w = 4$). The result of the PCA-based outcome is using the principal components learned from a tree of size 2,000 nodes expanded by standard RRT.

In order to quantitatively estimate the coverage, a discretization of the C-space was employed. Each dimension of the C-space (x, y, θ) is divided into 50 intervals. This results into the definition of $50 \times 50 \times 50$ cells into the C-space. Initially the density of each cell is zero and increases every time that a node of the tree lies on the cell. A tree that covers nicely the state space should result into cells with relatively equal densities. Thus, a metric of the C-space coverage is the variance of the density over the cells. For the same size tree, the lower the variance the better. For the example in Figure 9, the variance of the density achieved by RRT is 0.15, while the PCA alternative achieves 0.07.

Table III provides the variance of the density of cells for trees of different sizes (500, 1000, 3000, 5000 nodes), for different initial conditions (velocity $w = 0, w = 2, w = 4$) and for different algorithms (standard RRT, PCA-RRT using learning on a tree of the same size and PCA-RRT using learning on a tree of 1,000 nodes). Each value is the average variance metric achieved over 10 experiments. The standard deviation for the variance metric is quite low. Notice that in the majority of the cases the PCA-RRT achieves lower variance for the same tree size. This means that it explores the C-space more evenly. The comparison becomes increasingly more favorable for larger trees. Furthermore, if the tree used for offline learning is smaller than the constructed tree, there is no degradation in performance. On the contrary, it appears to be beneficial in terms of coverage.

V. DISCUSSION

This paper presents a method that utilizes Principal Component Analysis (PCA) to improve the coverage efficiency of sampling-based methods in the presence of dynamic constraints and underactuation. An offline step is executed first so as to learn the major influence from the constraints using PCA. A modification to the online step of the popular RRT algorithm is also presented that introduces a counter-measure to the bias and balances the overall exploration of the state-space. The approach is tested on a typical 3-link Acrobot system with varying levels of actuation and a second-order, not small-time locally controllable car-like system. The experimental results indicate that it is possible to benefit from the application of the PCA. The coverage of the state-space is improved and the cost of finding a solution to specific planning challenges is reduced. There are also experimental indications that for the tested systems a single small tree is sufficient for offline learning.

In the presented work PCA is used in a global fashion and only once. Future work will focus on executing PCA online, as well as locally. An online variant would hopefully be able to adapt to the changes caused by the modifications to the algorithm, since once PCA is used, the algorithm is no longer exhibiting the same kind of bias as the offline tree. A local approach brings the promise of being able to better approximate the underlying non-linear bias by decomposing the space into regions where the bias may be varying. Another way to address the underlying non-linearity of the physical system is to consider more sophisticated methods, such as Isomap/LLE, and in general non-linear dimensionality reduction algorithms. An important issue, especially for higher-dimensional challenges, is to use the PCA process so as to identify a projection of the complete state space that should be used for more effective planning. For example, in the current experiments the distance metric and the PCA are computed on a projection space that ignores the velocities, since the goal is specified in terms of the joint angles. In the general case, however, it might be possible to automatically identify the appropriate projection of the complete state space that is sufficient to cover using the PCA adaptation. There has been recent work on the performance of random linear projections for sampling-based motion planning [39]. Finally, future work will also deal with how the information collected through a PCA procedure can assist the online solution of problems that involve obstacles and collisions.

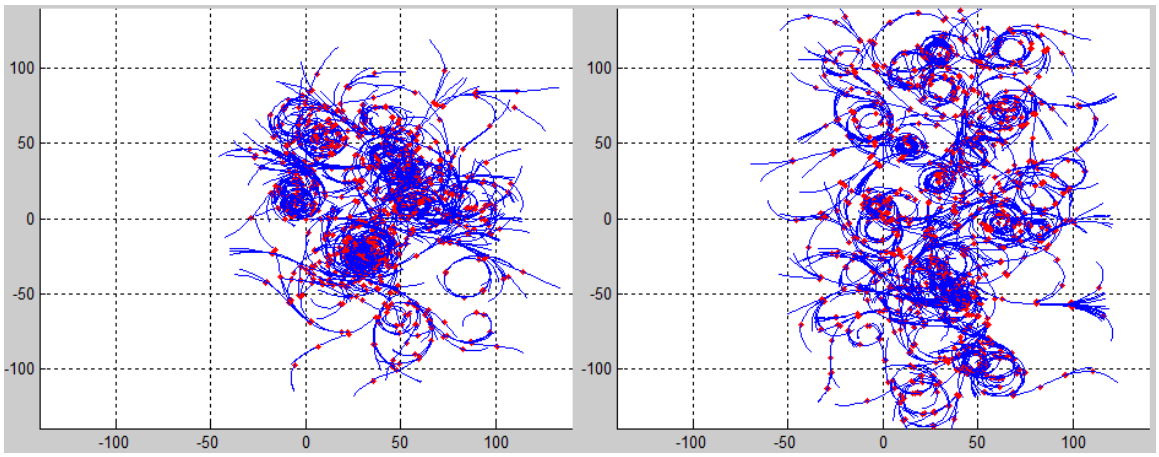

Fig. 9. Trees with 2000 nodes projected on the x, y plane. They are computed with the standard RRT (left) and the PCA-based solution (right) for the car-like system. The initial state is $(x, y, \theta, w, \zeta) = (0, 0, 0, 4, 0)$, which means that the car has its highest velocity in the initial state.

ACKNOWLEDGMENTS

Work on this paper has been supported by NSF CNS 0932423 and the University of Nevada, Reno. The authors would also like to thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] R. Murray and J. Hauser, "A case study in approximate linearization: The acrobot example," 1990.

[2] M. W. Spong, "Underactuated mechanical systems," in *Control Problems in Robotics and Automation, Lecture Notes in Control and Information Sciences*, B. Siciliano and K. P. Valavanis, Eds., 1997.

[3] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *IJRR*, vol. 20, no. 5, pp. 378–400, May 2001.

[4] S. LaValle and J. Kuffner, "Rapidly exploring random trees: Progress and prospects," in *WAFR*, 2001, pp. 293–308.

[5] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," April 2000.

[6] A. M. Ladd and L. E. Kavraki, "Motion planning in the presence of drift, underactuation and discrete system changes," in *Robotics: Science and Systems I*. MIT Press, 2005, pp. 233–241.

[7] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 559-572, 1901.

[8] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, no. 417-441, 1933.

[9] L. Zwald and G. Blanchard, "On the convergence of eigenspace in kernel principal component analysis," in *NIPS*, 2005.

[10] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato, "An obstacle-based rapidly-exploring random tree," in *ICRA*, Orlando, FL, May 2006, pp. 895–900.

[11] M. Rickert, O. Brock, and A. Knoll, "Balancing exploration and exploitation in motion planning," in *ICRA*, 2008.

[12] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 116–129, 2002.

[13] K. Hauser, T. Bretl, and J.-C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," in *WAFR*, 2006.

[14] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[15] F. L. Lewis and V. L. Syrmos, *Optimal Control*. John Wiley and Sons Inc., 1995.

[16] C. O'Dunlaing, "Motion planning with inertial constraints," *Algorithmica*, vol. 4, no. 2, pp. 431–475, 1987.

[17] J. Canny, A. Rege, and J. Reif, "An exact algorithm for kinodynamic planning in the plane," *Discrete and Computational Geometry*, vol. 6, pp. 461–484, 1991.

[18] J. T. Betts, "Survey of numerical methods for trajectory optimization," *AIAA Journal of Guidance, Control and Dynamics*, vol. 21, no. 2, pp. 193–207, March-April 1998.

[19] G. Sahar and J. Hollerbach, "Planning of minimum-time trajectories for robot arms," in *ICRA*, St. Louis, 1985.

[20] Z. Shiller and S. Dubowsky, "Global time-optimal motions of robotic manipulators in the presence of obstacles," in *ICRA*, 1988.

[21] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," vol. 40, no. 5, pp. 1048–1066, 1993.

[22] B. R. Donald and P. G. Xavier, "Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open chain manipulators." *Algorithmica*, vol. 4, no. 6, pp. 480–530, 1995.

[23] G. Heinzinger, P. Jacobs, J. Canny, and B. Paden, "Time-optimal trajectories for a robot manipulator: A provably good approximation algorithms," in *ICRA*, 1989, pp. 150–156.

[24] J. Kuffner and S. LaValle, "An efficient approach to path planning using balanced bidirectional rrt search," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., August 2005.

[25] P. Cheng, E. Frazzoli, and S. M. LaValle, "Improving the performance of sampling-based planners by using a symmetry-exploiting gap reduction algorithm," in *ICRA*, 2004.

[26] F. Lamiraux, E. Ferre, and E. Vallee, "Kinodynamic motion planning: Connecting exploration trees using trajectory optimization methods," in *ICRA*, 2004, pp. 3987–3992.

[27] A. M. Ladd, "Motion planning for physical simulation," Ph.D. dissertation, Computer Science Department, Rice University, 2007.

[28] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space voronoi bias," in *ICRA*, 2009.

[29] ——, "High-dimensional underactuated motion planning via task space control," in *IROS*, 2008.

[30] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *ICRA*, 2006, pp. 1874–1879.

[31] W. V. Weghe, D. Ferguson, and Srinivasan, "Randomized path planning for redundant manipulators without inverse kinematics," in *IEEE International Conference on Humanoid Robots*, November 2007.

[32] R. Diankov, N. Ratliff, D. Ferguson, S. S., and J. Kuffner, "Bispace planning: Concurrent multi-space exploration," in *Robotics: Science and Systems IV*, June 2008.

[33] M. Stilman, "Task constrained motion planning in robot joint space," in *IROS*, 2007, pp. 3074–3081.

[34] Z. Yao and K. Gupta, "Path planning with general end-effector constraints: using task space to guide configuration space search," in *IROS*, 2005, pp. 1875–1880.

[35] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *ICRA*, 2009, pp. 625–632.

[36] S. Dalibard and J.-P. Laumond, "Control of probabilistic diffusion in motion planning," in *WAFR*, 2008.

[37] M. L. Teodoro, J. G PN, and L. Kavraki, "A dimensionality reduction approach to modeling protein flexibility," in *RECOMB*, New York, NY, USA, 2002, pp. 299–308.

[38] L. Tapia, S. Thomas, and N. M. Amato, "Using dimensionality reduction to better capture rna and protein folding motions," Computer Science, Texas A&M University, Tech. Rep., 2008.

[39] I. A. Sucan and L. E. Kavraki, "On the performance of random linear projectinos for sampling-based motion planning," in *IROS*, 2009.