Published in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 420-425, 2012

Low-Dimensional Projections for SyCLoP

Matthew R. Maly and Lydia E. Kavraki

Abstract— **This paper presents an extension to SyCLoP, a multilayered motion planning framework that has been shown to successfully solve high-dimensional problems with differential constraints. SyCLoP combines traditional sampling-based planning with a high-level decomposition of the workspace through which it attempts to guide a low-level tree of motions. We investigate a generalization of SyCLoP in which the high-level decomposition is defined over a given low-dimensional projected subspace of the state space. We begin with a manually-chosen projection to demonstrate that projections other than the workspace can potentially work well. We then evaluate SyCLoP's performance with random projections and projections determined from linear dimensionality reduction over elements of the state space, for which the results are mixed. As we will see, finding a useful projection is a difficult problem, and we conclude this paper by discussing the merits and drawbacks of various types of projections.**

I. Introduction

The classic motion planning problem for a robot requires computing a trajectory that takes the robot from a start state to a goal region and is free of collisions. Early work on this problem includes a proof of PSPACE-completeness for the motion planning problem with respect to the number of degrees of freedom of the robot [1]. In response, much of the motion planning research community shifted its focus to sampling-based approaches, which trade completeness guarantees for tractable time and space complexity [2], [3]. Sampling-based algorithms include the roadmap-based PRM planner [4], and the tree-based RRT [5], EST [6], SBL [7], PDST [8], and KPIECE [9] planners, among many others. Such sampling-based algorithms offer *probabilistic completeness*, which means that the probability that such an algorithm will find a solution (assuming one exists) approaches 1 as the algorithm spends more time on the problem. A probabilistically complete motion planning algorithm cannot in general detect if a solution does not exist.

The success of sampling-based motion planning algorithms has prompted researchers to apply them to increasingly difficult problems. One class of such problems includes robotic systems with differential constraints. In these systems, robots can only exhibit motions that are realized by the application of a sequence of controls. The classic motion planning problem can be generalized to incorporate robotic dynamics by including the additional requirement that the computed trajectory satisfies the differential constraints imposed by the robot's equations of motion. Many sampling-based planners can easily be generalized to solve such problems, where a tree state q not only holds a pointer to its parent state $p(q)$ but also stores the necessary controls to realize a motion from $p(q)$ to q.

This paper investigates the use of low-dimensional projections for SyCLoP, a synergistic multilayered motion planning framework that has been successfully used to solve high-dimensional motion planning problems with differential constraints [10]. SyCLoP is a meta-planner that outperforms classic sampling-based planners for high-dimensional problems with differential constraints by combining high-level discrete search with a low-level sampling-based motion planner. On the high-level side, SyCLoP uses a projection (called the *workspace projection*) that maps from the robot's state space into the robot's workspace. Using this projection, SyCLoP partitions the workspace into cells and creates an adjacency graph of neighboring cells. SyCLoP then searches this adjacency graph to determine a contiguous sequence of regions (called a *lead*) through which to guide a low-level tree of motions defined in the state space. As the motions in the tree are constrained by the dynamics of the robotic system, SyCLoP cannot easily force the tree to grow exactly through a given lead. Instead, information on where the tree is able to grow is passed to SyCLoP's high-level layer which affects the generation of future leads, creating a two-way channel of information between the layers, as illustrated in Figure 1. SyCLoP has been shown to yield significant speedups of up to two orders of magnitude when compared to classic tree-based motion planners such as RRT and EST [10].

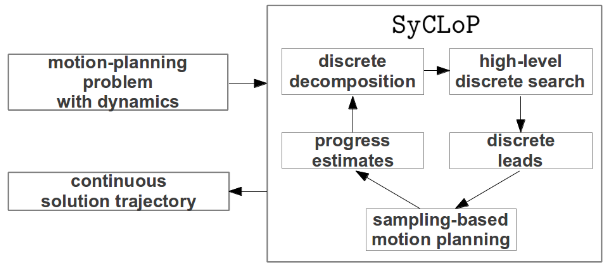
SyCLoP
motion-planning problem with dynamics
discrete decomposition
high-level discrete search
progress estimates
discrete leads
continuous solution trajectory
sampling-based motion planning

Fig. 1. The SyCLoP architecture.

We have reimplemented SyCLoP as part of the Open Motion Planning Library (OMPL) [11]. In the process, we have generalized SyCLoP's high-level layer to accept an arbitrary linear projection as an additional input parameter. As a result, the subspace in which SyCLoP computes leads is customizable, determined by the linear projection given as an input parameter to our software. The original implementation of SyCLoP computed leads exclusively in the workspace, which is one example of such a subspace, determined by the workspace projection which maps the robot's state to its (x, y) location.

In this paper we will investigate the use of various projections with `SyCLoP` and compare their performance to the workspace projection that has been previously used. First, we will show that some nonworkspace projections can improve `SyCLoP`'s performance. Then we will investigate the use of random projections and projections obtained by linear dimensionality reduction. We will show that in general, finding the best projection for `SyCLoP` is problem-specific and often difficult.

Section II describes related approaches on guiding tree-based motion planners through low-dimensional projections. Section III formally introduces the motion-planning problem with dynamics and the original formulation of `SyCLoP`. Section IV describes our extensions to `SyCLoP`. We have tested our generalized reimplementation of `SyCLoP` on multiple control-based robotic systems with various projections; these experimental results are given in Section V. Finally, concluding remarks and a discussion of future work are given in Section VI.

II. Related Work

The use of low-dimensional projections in sampling-based motion planning is not new. As an example, tree-based planners such as `PDST` and `KPIECE` use low-dimensional projections to guide the tree of motions. `PDST` dynamically subdivides a projected subspace of the state space in order to estimate coverage without the use of a metric, and has been shown to benefit from the use of a nonworkspace projection [8]. `KPIECE` chooses where to expand its tree of motions by considering the tree's coverage of a space determined by some low-dimensional projection [9]. `KPIECE` has been shown to benefit from random linear projections when planning for high-dimensional systems [12]. This motivates our investigation of nonworkspace projections for `SyCLoP`.

III. Framework

We first introduce the motion-planning problem with dynamics and then describe `SyCLoP`, a hybrid motion planner designed to solve it.

A. Problem Statement

A motion-planning problem with dynamics consists of

1) $\mathcal{Q} \subset \mathbb{R}^n$, a bounded n-dimensional state space, an element of which completely determines the system's state,
2) $\mathcal{U} \subset \mathbb{R}^c$, a bounded c-dimensional control space consisting of control variables that can be applied to the system to change its state,
3) $\texttt{ctrl} : \mathcal{Q} \times \mathcal{U} \rightarrow \dot{\mathcal{Q}}$, a differential equation that captures the system's constraints,
4) $\texttt{valid} : \mathcal{Q} \rightarrow \{0, 1\}$, a boolean function describing whether a state is valid (used for collision avoidance),
5) $q_{\text{init}} \in \mathcal{Q}$, a start state for the system, and
6) $Q_{\text{goal}} \subseteq \mathcal{Q}$, a goal region in which the system should achieve a state.

A solution to a given motion-planning problem with dynamics is a control function $u : [0, T] \rightarrow \mathcal{U}$ that moves the system from the start state q_{init} to a state $q \in Q_{\text{goal}}$, with the requirement that $\texttt{valid}(q) = 1$ for all states q along the trajectory realized by u.

B. `SyCLoP`

The pseudocode for `SyCLoP` is given in Algorithm 1. Lines 1 and 8-11 comprise the behavior of `SyCLoP`'s low level, in which the expansion of a tree is promoted within a given decomposition region. `SyCLoP`'s high-level behavior is seen in lines 3-6 and 12-15, in which high-level leads are computed, and aggregate information regarding states and decomposition regions is collected to affect the computation of future leads. Lines 5 and 7 feature two of `SyCLoP`'s tunable parameters ("number of region expansions" and "number of tree selections"), which in this work are set to 100 and 50, respectively. For complete details of the `SyCLoP` algorithm, we refer the reader to [10].

Algorithm 1 `SyCLoP`

Input: A motion-planning problem with dynamics $(\mathcal{Q}, \mathcal{U}, \texttt{ctrl}, \texttt{valid}, q_{\text{init}}, Q_{\text{goal}})$, a workspace decomposition $\mathcal{D}$, and a time bound $t_{\max}$.
Output: A solution to the given problem, or `NULL` if one cannot be found within time $t_{\max}$

```
1:  $\mathcal{T} \leftarrow$ InitializeTree($q_{\text{init}}$)
2:  while Time Elapsed $< t_{\max}$ do
3:      $(R_{i_1}, \ldots, R_{i_k}) \leftarrow$ ComputeLead($\mathcal{D}$)
4:      $R_{\text{avail}} \leftarrow$ ComputeAvailableRegions($(R_{i_1}, \ldots, R_{i_k})$)
5:      for number of region expansions do
6:          $R_s \leftarrow$ SelectRegion($R_{\text{avail}}$)
7:          for number of tree selections do
8:              SelectAndExtend($\mathcal{T}, R_s$)
9:              for each new state $s$ added to $\mathcal{T}$ in line 8 do
10:                 if $s \in Q_{\text{goal}}$ then
11:                     return trajectory from $q_{\text{init}}$ to $s$
12:                 $R_n \leftarrow$ LocateRegion($s$)
13:                 if $R_n \notin R_{\text{avail}}$ then
14:                     $R_{\text{avail}} \leftarrow R_{\text{avail}} \cup \{R_n\}$
15:                 UpdateEstimates($R_n, s$)
16:         if no improvement to high-level estimates then
17:             abandon current lead with probability $p$
18:             return to line 3 to compute new lead
19: return NULL
```

`SyCLoP` is intended to be used with a low-level tree-based planner. Notice that the low-level tree planner used in `SyCLoP` is not specified. Any tree planner that supports planning with differential constraints can be used as the low-level tree planner for `SyCLoP`. In this work, we restrict our attention to `SyCLoP` with `RRT` as its low-level tree planner, which for simplicity we refer to as `SyCLoP` [10].

IV. Generalizing `SyCLoP`

We have generalized the discrete layer of `SyCLoP` so that high-level guides can be computed through any subspace of the state space $\mathcal{Q}$ determined by some low-dimensional projection. Specifically, `SyCLoP` has been extended to accept as inputs a projection

$$\text{Proj} : \mathcal{Q} \rightarrow \mathbb{R}^k,$$

where $k \leq \dim(\mathcal{Q})$, a lift function

$$\textsc{Lift} : \mathbb{R}^k \to \mathcal{Q}$$

which approximates $\textsc{Proj}^{-1}$, and a decomposition

$$\mathcal{D} = \{R_1, \dots, R_m\},$$

where $\mathcal{D}$ is a geometric partition of the projected k-dimensional subspace $\textsc{Proj}(\mathcal{Q})$, so that $\cup_{i=1}^{m} R_i = \textsc{Proj}(\mathcal{Q})$ and $R_a \cap R_b = \varnothing$ if $a \neq b$. For this work, we restrict our attention to the case in which $\textsc{Proj}$ is a linear projection; i.e., for each state $q \in \mathcal{Q}$,

$$\textsc{Proj}(q) = \mathbf{M}q,$$

for some fixed $k \times n$ matrix $\mathbf{M}$. Additionally, we restrict our attention to two-dimensional decompositions ($k = 2$); in general, some systems may benefit from a three-dimensional decomposition.

Computing the bounds of a given decomposition and sampling a full state from a decomposition region have stood out as the most challenging changes to `SyCLoP`'s generalization to accept arbitrary linear projections such as the ones described above.

A. Computing Decomposition Bounds

Knowing the bounds of $\textsc{Proj}(\mathcal{Q})$ is necessary when sampling states from a given decomposition region, a crucial step in the `SyCLoP` algorithm. In `SyCLoP`'s original formulation, computing the bounds of the decomposition and each of its regions was trivial for vehicular exploration problems, as the decomposition space was equivalent to the two-dimensional workspace, and the workspace could easily be extracted as two dimensions of the state space.

When the decomposition space follows from an arbitrary linear projection, its bounds must be computed differently. Recall $n = \dim(\mathcal{Q})$, and let $q_{j,\texttt{low}}$ and $q_{j,\texttt{high}}$ denote the lower and upper scalar bounds, respectively, of the jth dimensional axis of the state space $\mathcal{Q}$ for each $j \in \{1, \dots, n\}$. Further suppose that $\textsc{Proj}$ is a linear projection defined by some $k \times n$ matrix $\mathbf{M}$. For each $i \in \{1, \dots, k\}$, we compute the lower and upper scalar bounds $d_{i,\texttt{low}}$ and $d_{i,\texttt{high}}$ of each ith dimensional axis of the decomposition space $\textsc{Proj}(\mathcal{Q})$ as follows.

$$d_{i,\texttt{low}} = \sum_{j=1}^{n} \min\left\{\mathbf{M}_{i,j} \cdot q_{j,\texttt{low}}, \mathbf{M}_{i,j} \cdot q_{j,\texttt{high}}\right\}$$

$$d_{i,\texttt{high}} = \sum_{j=1}^{n} \max\left\{\mathbf{M}_{i,j} \cdot q_{j,\texttt{low}}, \mathbf{M}_{i,j} \cdot q_{j,\texttt{high}}\right\}.$$

The bounds of $\textsc{Proj}(\mathcal{Q})$ are used to sample states from $\mathcal{Q}$ whose projections reside in specific decomposition regions, a process which we describe in the following section.

B. Sampling States from Regions

The $\textsc{SelectAndExtend}(\mathcal{T}, R_s)$ step of the `SyCLoP` algorithm, which extends the low-level tree planner within a given nonempty region R_s, becomes more difficult to implement given an arbitrary projection. Some planners, such as `RRT` and its many variants, sample a random state $q \in \mathcal{Q}$ and then attempt to grow the tree towards q. The `SyCLoP`-guided variant of such a planner should then sample a random state from region R_s, i.e., a state $q \in \mathcal{Q}$ so that $\textsc{Proj}(q) \in R_s$. In `SyCLoP`'s original formulation, accomplishing this task was simple. For example, if we were planning for a planar vehicle with state $q = (x, y, \theta, v)$, where (x, y) is the vehicle's location, θ is its heading, and v is its forward velocity, then an workspace projection is defined so that $\textsc{Proj}(x, y, \theta, v) = (x, y)$. To sample a state q so that $\textsc{Proj}(q) \in R_s$, we would sample a point $(x_r, y_r) \in R_s$, and then return the state $q_r = (x_r, y_r, \theta_r, v_r)$, where θ_r and v_r are sampled at random.

With an arbitrary projection, we must do some extra work to accomplish this task. This is where the input $\textsc{Lift}$ comes into play. To obtain a random state q so that $\textsc{Proj}(q) \in R_s$, we uniformly sample a random point $p \in R_s$ and use the state $\textsc{Lift}(p) \in \mathcal{Q}$. In general, $p \in R_s$ does not necessarily imply that $\textsc{Proj}(\textsc{Lift}(p)) \in R_s$. In addition, following a $\textsc{Lift}$ operation to obtain a state q, it is possible that q is outside of the bounds of $\mathcal{Q}$. Hence we follow each $\textsc{Lift}$ operation with an operation to bring the obtained state within the bounds of the state space. Specifically, if $\textsc{Lift}$ returns a state $q = (q_1, \dots, q_n)$ whose ith dimensional component q_i is outside of the bounds of the ith dimensional component $\mathcal{Q}_i$ of the state space $\mathcal{Q}$, then the value of q_i is set to the lower or upper bound of $\mathcal{Q}_i$, whichever is closer. As long as the returned state q is within the bounds of the state space, it is acceptable if q is in collision. This is analogous to the sampling step in the `RRT` algorithm, in which the sampled state toward which the tree is expanded does not need to be collision-free [5]. Hence we do not follow this step with collision checking.

V. Experiments

A. Dynamic Vehicles Used

Our experiments involve second-order models of two dynamic vehicles, the car and the tractor-trailer.

1) Car (adapted from [10]): A dynamic car has state $q = (x, y, \theta, v, \psi)$, consisting of the planar position $(x, y) \in \mathbb{R}^2$, planar orientation $\theta \in [-\pi, \pi]$, forward velocity $v \in \mathbb{R}$, and steering angle $\psi \in [-\pi, \pi]$. We include the bounds $|x|, |y| \leq 55\ m$, $|v| \leq 1\ m/s$, and $|\psi| \leq 30°$. The car is controlled with the acceleration u_0 and the steering angle velocity u_1, with the bounds $|u_0| \leq 5\ m/s^2$ and $|u_1| \leq\ 2°/s$. The motions of the car are constrained by the differential equations $\dot{x} = v\cos(\theta)$, $\dot{y} = v\sin(\theta)$, $\dot{\theta} = v\tan(\psi)$, $\dot{v} = u_0$, and $\dot{\psi} = u_1$.

2) Tractor-Trailer (adapted from [10]): A dynamic tractor-trailer is modeled as a car that pulls behind it some number t of trailers. A tractor-trailer has state $q = (x, y, \theta_0, v, \psi, \theta_1, \dots, \theta_t)$, where $(x, y, \theta_0, v, \psi)$ are constrained the same as with the dynamic car in Section V-A.1. Each trailer i has planar orientation $\theta_i \in [-\pi, \pi]$. The equations of motion for the tractor-trailer include those for the car as well as

$$\dot{\theta}_i = \frac{v}{d}\left(\prod_{j=1}^{i-1}\cos(\theta_{j-1}-\theta_j)\right)(\sin(\theta_{i-1}-\theta_i))$$

for each $i \in \{1,\ldots,t\}$.

B. Projections Used

We restrict our attention to three types of two-dimensional linear projections. Let $n = \dim(\mathcal{Q})$.

1) Velocity Projection: Under the assumption that $\mathcal{Q}$ contains dimensions corresponding to a planar (x, y) location and a forward velocity v, we define a two-dimensional *velocity projection* so that for each state $q \in \mathcal{Q}$,

$$\text{PROJ}(q) = (x + cv, y + cv),$$

where c is the *velocity coefficient*. Notice that when $c = 0$, PROJ is simply the workspace projection, used by the original `SyCLoP` algorithm.

We pair with the velocity projection the same LIFT operation that was described for the workspace projection in Section IV-B. This will be a reasonable approximation in our experiments, as we will keep both the velocity bounds and the coefficient c much smaller than the bounds on x and y.

2) Random Projections: We define a two-dimensional random projection as the matrix

$$\mathbf{R} = \begin{bmatrix} a_{1,1} & \ldots & a_{1,n} \\ a_{2,n} & \ldots & a_{2,n} \end{bmatrix},$$

where each entry $a_{i,j}$ is sampled from a Gaussian distribution with mean 0 and variance 1, and the rows of $\mathbf{R}$ are made orthonormal. We define a two-dimensional random projection so that for each state $q \in \mathcal{Q}$,

$$\text{PROJ}(q) = \mathbf{R}q.$$

Since $\mathbf{R}$ is a real-valued matrix with orthonormal rows, its pseudoinverse is equivalent to its transpose $\mathbf{R}^T$. Hence we pair with each random projection the LIFT operation defined so that for each $p \in \mathbb{R}^k$,

$$\text{LIFT}(p) = \mathbf{R}^T p + (I - \mathbf{R}^T\mathbf{R})w,$$

where w is a randomly sampled vector. This is to capture all possible solutions to the equation $\text{PROJ}(q) = p$.

3) PCA-induced Projections: We also consider projections taken from applying linear dimensionality reduction to elements of a given state space. Specifically, we use principal component analysis (PCA) with two approaches:

1) Run a sampling-based tree planner until a sample solution path is found. Output just the states from this path to PCA.
2) Run a sampling-based tree planner for 30 s to cover the state space. Output all states to PCA.

Following each of the above two approaches, we take the first two principal components (in descending order of variance) from PCA to obtain a two-dimensional projection, the rows of which are normalized. We call these two projections the *path-induced* and *space-induced* projections, respectively. Similar to the case of the random projections, we use the transpose matrices to define LIFT operations.

TABLE I
DYNAMIC CAR EXPERIMENTS USING VELOCITY PROJECTIONS

Environment	Vel. Coef.	Solved %	Avg. Soln. Time
Free	0.0	100%	15.88 s
	0.2	100%	14.1 s
	0.4	100%	15.12 s
	0.6	100%	14.8 s
	0.8	100%	13.54 s
	1.0	100%	13.01 s
Maze	0.0	100%	31.59 s
	0.2	100%	30.5 s
	0.4	100%	28.88 s
	0.6	100%	29.5 s
	0.8	100%	28.81 s
	1.0	100%	29.27 s

C. Results

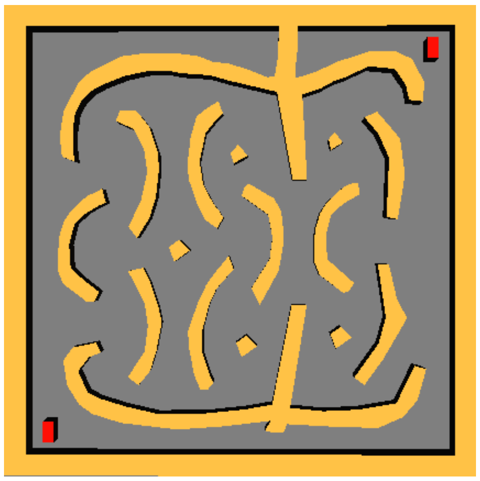

Fig. 2. A maze environment with a second-order car: the start state is located at the bottom-left of the environment, and the goal region is located at the top-right of the environment.

We have run our generalized version of `SyCLoP` on a second-order car model and a tractor-trailer model with 3 trailers. For the car, we consider a free space environment (no obstacles) and a maze environment, which is pictured in Figure 2. For the tractor-trailer model, we consider only the maze environment. We have tested these models and environments with all of the projections described in Section V-B. All experiments were written in C++ using `OMPL` [11] and were run on an Intel Core 2 Quad machine running at 2.83 GHz with 8 GB of RAM.

Table I contains planner performance data for a car traveling through free space and a maze environment using velocity projections. Each experiment ran 50 times per velocity projection. Notice that when the velocity coefficient is 0, the velocity projection is equivalent to the workspace projection,

TABLE II

DYNAMIC CAR EXPERIMENTS USING RANDOM PROJECTIONS

Environment	Random Proj. #	Solved %	Avg. Soln. Time
Free	1	93%	34.17 s
	2	77%	63.81 s
	3	100%	21.94 s
	4	100%	20.23 s
	5	100%	34.59 s
Maze	6	73%	90.04 s
	7	20%	115.44 s
	8	27%	112.27 s
	9	0%	120 s
	10	67%	87.82 s

TABLE III

DYNAMIC CAR EXPERIMENTS USING PCA PROJECTIONS

Environment	Projection	Solved %	Avg. Soln. Time
Free	Path-induced	98%	27.66 s
	Space-induced	100%	13.89 s
Maze	Path-induced	100%	28.45 s
	Space-induced	100%	31.58 s

TABLE IV

TRACTOR-TRAILER EXPERIMENTS USING VELOCITY PROJECTIONS

Environment	Vel. Coef.	Solved %	Avg. Soln. Time
Maze	0.0	97%	73.93 s
	0.2	100%	62.1 s
	0.4	100%	72.78 s
	0.6	100%	61.82 s
	0.8	100%	55.38 s
	1.0	100%	53.39 s

TABLE V

TRACTOR-TRAILER EXPERIMENTS USING RANDOM PROJECTIONS

Environment	Random Proj. #	Solved %	Avg. Soln. Time
Maze	1	13%	228.87 s
	2	30%	216.35 s
	3	37%	210.84 s
	4	20%	221.04 s
	5	53%	185.05 s

TABLE VI

TRACTOR-TRAILER EXPERIMENTS USING PCA PROJECTIONS

Environment	Projection	Solved %	Avg. Soln. Time
Maze	Path-induced	100%	74.95 s
	Space-induced	100%	74.02 s

TABLE VII

EXPERIMENTS USING RRT

Vehicle	Environment	Solved %	Avg. Soln. Time
Car	Free	100%	19.15 s
	Maze	24%	112.73 s
Tractor	Maze	23%	195.35 s

which is our baseline of comparison (the original version of SyCLoP). Table II contains planner performance data for a car traveling through free space and a maze environment using random projections. For each environment, five random projections were used, with averages taken over 30 runs. Table III contains planner performance data for a car traveling through free space and a maze environment using the path-induced and space-induced PCA projections, with averages taken over 50 runs. For each run of each car experiment, a 120 s timeout is enforced.

Table IV contains performance data for the tractor-trailer model using velocity projections, with averages taken over 30 runs. As with the car, our baseline of comparison is the velocity projection with coefficient 0. Table V contains performance data for the tractor-trailer model using random projections, and Table VI contains performance data for the tractor-trailer model using the path-induced and space-induced PCA projections, with averages taken over 30 runs. For each run of each tractor-trailer experiment, a 240 s timeout is enforced.

Table VII contains performance data for the car and tractor-trailer models using RRT. Although the focus of our work is an optimization within SyCLoP, this table is meant as a brief demonstration of SyCLoP's superior performance over other sampling-based planning techniques. For a detailed comparison of SyCLoP to other methods, we refer the reader to [10].

D. Analysis

The velocity projection has improved performance of SyCLoP. This is likely due to the fact that when we incorporate velocity into the projection, the same physical region in the workspace is potentially mapped to multiple decomposition regions, differentiated by the velocity of states that exist within that physical workspace region. This enables SyCLoP's high-level layer to compare the successes of different velocities in the same region of the workspace, which allows it to value paths of higher velocity in open regions of the workspace and similarly paths of lower velocity in cluttered regions of the workspace. The more SyCLoP can take advantage of high velocity paths, the faster the simulated paths in its tree of motions will be, leading more quickly to a solution path.

In Tables III and VI, we see that the projections generated from PCA yield mixed results. For the car, PCA projections slightly improve performance compared to the workspace projection. For the tractor-trailer, PCA projections slightly degrade performance.

Random projections degrade performance of SyCLoP in all experiments, and they become almost unusable in the case of the maze environment.

There are many ways in which a complex linear projection (such as a random projection or PCA projection) can potentially degrade SyCLoP's performance. For one, SyCLoP requires the ability to sample a state from a given decomposition region by sampling a point p from the region and then computing $\text{LIFT}(p)$ as defined in Section IV. Given a sufficiently complex projection PROJ, for a state q sampled by SyCLoP given a decomposition region R_i, linearity ensures that $q \in \text{LIFT}(R_i)$. However, by the definition of LIFT, the state q may not even be within bounds of $\mathcal{Q}$. In such cases, our algorithm brings q within the bounds of $\mathcal{Q}$. SyCLoP will attempt to grow its low-level tree toward q, likely at the edge of $\mathcal{Q}$, where further collision-free operations may be impossible. This issue distorts the information passed between SyCLoP's high-level and low-level layers: SyCLoP's high-level reacts to the failure to promote low-level tree expansion in region R_i as if the failure was due to differential constraints or obstacles in the environment. Instead, it is likely that the true culprit was that inaccuracies with LIFT caused the low-level layer to attempt expansion from some other region R_j, and the high-level layer was not aware of this issue.

A second issue with complex linear projections is that they challenge one of SyCLoP's core tasks, which is to create high-level sequences of adjacent workspace regions through which to guide a tree of motions. Under the workspace projection, adjacency of decomposition regions corresponds to physical adjacency of regions in the robot's environment. Given the vehicular models used in these experiments, motions between adjacent environment regions are typically realizable as long as enough free space exists in which to move. Under other low-dimensional projections, we cannot guarantee correspondence between decomposition regions and environment regions, and motions between adjacent decomposition regions (without first traveling through other decomposition regions) may often be impossible. Ultimately, low-dimensional projections can fundamentally alter the topology of the state space. This issue challenges the motivating feature of SyCLoP, which is to create a sequence of adjacent regions as a helpful lead to guide the low-level planner. Given these potential drawbacks combined with the merely slight performance advantages obtained from using some non-workspace projections, we advocate the use of the original workspace projection for SyCLoP in general.

VI. CONCLUSION AND FUTURE WORK

We have presented a generalization of the SyCLoP planner to accept any arbitrary linear projection through which it guides the low-level tree planner. With algorithms such as SyCLoP that guide a low-level tree planner through a low-dimensional projection and make assumptions based on the planner's ability to explore certain areas of that projection, the projection used should likely be closely tied to the dynamics of both the robot and its environment.

For future work, we hope to improve SyCLoP's handling of high-dimensional goals and obstacles. All of the experimental setups presented in this paper use two-dimensional goal regions which are defined in terms of the workspace. More realistic motion planning problems require goals of high dimension, and it may prove advantageous to incorporate goal information into a low-dimensional projection for SyCLoP, in a similar approach to the velocity projection. The problem of how to manage high-dimensional goals and obstacles is not unique to SyCLoP but instead could have applications to motion planning in general.

ACKNOWLEDGEMENTS

Work on this paper has been supported in part by NSF CCF 1018798, NSF Expeditions 1139011 and U. S. Army Research Laboratory and the U. S. Army Research Office under grant number W911NF-09-1-0383.

In addition, the authors would like to thank the members of the Kavraki group at Rice University for their helpful comments and suggestions.

REFERENCES

[1] J. Canny, "Some algebraic and geometric computations in PSPACE," in *Annual ACM Symposium on Theory of Computing*. Chicago, Illinois, United States: ACM Press, 1988, pp. 460–469.

[2] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: http://msl.cs.uiuc.edu/planning/

[4] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

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

[6] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Intl. J. of Computational Geometry and Applications*, vol. 9, no. 4-5, pp. 495–512, 1999.

[7] G. Sánchez and J.-C. Latombe, "On delaying collision checking in PRM planning: Application to multi-robot coordination," *Intl. J. of Robotics Research*, vol. 21, no. 1, pp. 5–26, Jan. 2002.

[8] A. M. Ladd and L. E. Kavraki, "Fast tree-based exploration of state space for robots with dynamics," in *Algorithmic Foundations of Robotics VI*, M. Erdmann, D. Hsu, M. Overmars, and A. F. van der Stappen, Eds. Springer, STAR 17, 2005, pp. 297–312.

[9] I. A. Şucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Trans. on Robotics*, vol. 28, no. 1, pp. 116–131, 2012.

[10] E. Plaku, L. Kavraki, and M. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Trans. on Robotics*, vol. 26, no. 3, pp. 469–482, jun. 2010.

[11] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, 2012, to appear. [Online]. Available: http://ompl.kavrakilab.org

[12] I. A. Şucan and L. E. Kavraki, "On the performance of random linear projections for sampling-based motion planning," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Oct. 2009, pp. 2434–2439.