Linear Dimensionality Reduction in Random Motion Planning

Sébastien Dalibard, Jean-Paul Laumond

Linear Dimensionality Reduction in Random Motion Planning

Sébastien Dalibard and Jean-Paul Laumond

LAAS-CNRS [†]
University of Toulouse
{sdalibar,jpl}@laas.fr

Abstract: The paper presents a method to control probabilistic diffusion in motion planning algorithms. The principle of the method is to use on line the results of a diffusion algorithm to describe the free space in which the planning takes place, by computing a Principal Component Analysis (PCA). This method identifies the locally free directions of the free space. Given that description, our algorithm accelerates the diffusion along these favoured directions. That way, if the free space appears as a small volume around a submanifold of a highly dimensioned configuration space, the method overcomes the usual limitations of diffusion algorithms and finds a solution quickly. The presented method is theoretically analyzed and experimentally compared to known motion planning algorithms.

1 Problem statement, related work and contribution

1.1 General Framework

Motion planning problems have been intensively studied in the last decades, with applications in many diverse areas, such as robotics, part disassembly problems in Product Lifecycle Management (PLM), digital actors in computer animation, or even protein folding and drug design. For comprehensive overviews of motion planning problems and methods, one can refer to (Latombe, 1991), (Choset *et al.*, 2005) and (LaValle, 2006).

In the past fifteen years, two kinds of configuration space ($\mathcal{CS}$) search paradigms have been investigated with success.

- The *sampling* approach, first introduced in (Kavraki *et al.*, 1996) as probabilistic roadmaps (PRM), consists in computing a graph, or a roadmap, whose vertices are collision free configurations, sampled at random in the free space and whose edges reflect the existence of a collision free elementary path between two configurations. It aims at capturing the topology of the collision free space ($\mathcal{CS}_{free}$) in a learning phase in order to handle multiple queries in a solving phase.

[†] This work is partly supported by the French ANR-RNTL project PerfRV2. A preliminary version of this paper appeared in WAFR'08.

- The *diffusion* approach, introduced in both (Hsu *et al.*, 1999) and (Kuffner & LaValle, 2000), which includes RRT planners, consists in solving single queries by growing a tree rooted at the start configuration towards the goal configuration to be reached. That approach can be seen as the exploration of the tangent space of $\mathcal{CS}$: starting from collision free configurations, the algorithm tries directions in $\mathcal{CS}$ which will give collision free paths. In that sense, it is fundamentally different from PRM algorithms, which build a static description of the configuration space.

These methods have been proven to be efficient and suitable for a large class of motion planning problems. Work has been done to analyze and validate theoretically these algorithms. Probabilistic completeness has been studied and proved for RRT (Kuffner & LaValle, 2000), as well as for PRM (Kavraki *et al.*, 1998).

1.2 Narrow Passages

In some environments, however, passing through so-called *narrow passages* is a difficult task for probabilistic motion planners. A lot of work has been done to evaluate this difficulty, as well as to overcome it.

The formalism of expansive spaces was first presented in (Hsu *et al.*, 1999). A good description of it can be found in (Hsu *et al.*, 2006). It quantifies the complexity of a configuration space from a randomized planning point of view. The presence of narrow passages is identified as the main source of complexity for PRM planners. Indeed, if the path to find in $\mathcal{CS}$ goes through a narrow passage, the algorithm has to sample configurations inside the passage, which is difficult because its volume is small compared to $\mathcal{CS}$. Thus, this work demonstrates the difficulty of "finding" the passage, i.e. sampling at least one configuration inside it.

Motion planning problems occur though, when the difficulty is not to find the passage, but to go along it. For industrial disassembling problems for instance, the mechanical part to remove is already mounted and inside a constrained area. The motion planner has to find a path that follows the narrow passage until the part is disassembled. Going along a narrow passage can be a difficulty in itself for diffusion algorithms. Actually, the diffusion velocity strongly depends on the thickness of the narrow passage - i.e. on the radius of the largest sphere included in $\mathcal{CS}_{free}$. We will analyze that effect theoretically in Section 3 then in the experimental section. The purpose of our work is to speed up such constrained diffusion processes, it deals solely with diffusion within a narrow passage, and not with finding the entrance of narrow passages in $\mathcal{CS}$.

Some previous work deals with controlling probabilistic planning algorithms as well. Visibility based PRM is presented in (Siméon *et al.*, 2000). A visibility criteria is used to improve the quality of the sampling and accelerate the algorithm. Dynamic-domain RRTs, presented in (Yershova *et al.*, 2005), control the sampling domain of a RRT algorithm to overcome some bias of classical RRT algorithm. In (Burns & Brock, 2005), a local model of the configuration space shape is built and used to improve the efficiency of configuration sampling. A good review on different sampling techniques was presented in (Geraerts & Overmars, 2004).

More recently, work has been done to use workspace information within $\mathcal{CS}$ probabilistic algorithms. In (Rodriguez *et al.*, 2006) for instance, diffusion uses workspace hints, such as real world obstacle directions, or decouples rotations and translations to accelerate a RRT algorithm that diffuses through a narrow passage.

1.3 Motivations

Motivations for the work presented here come from motion planning for digital actors. In these high-dimensioned problems - we consider whole-body motion planning, diffusion algorithms behave rather well, but can be slowed down by narrow passages.

The fact that the whole-body motion takes place in a narrow passage of $\mathcal{CS}$, however, does not necessarily reflect the difficulty of a motion planning problem from a human point of view, as we will show it with the next example.

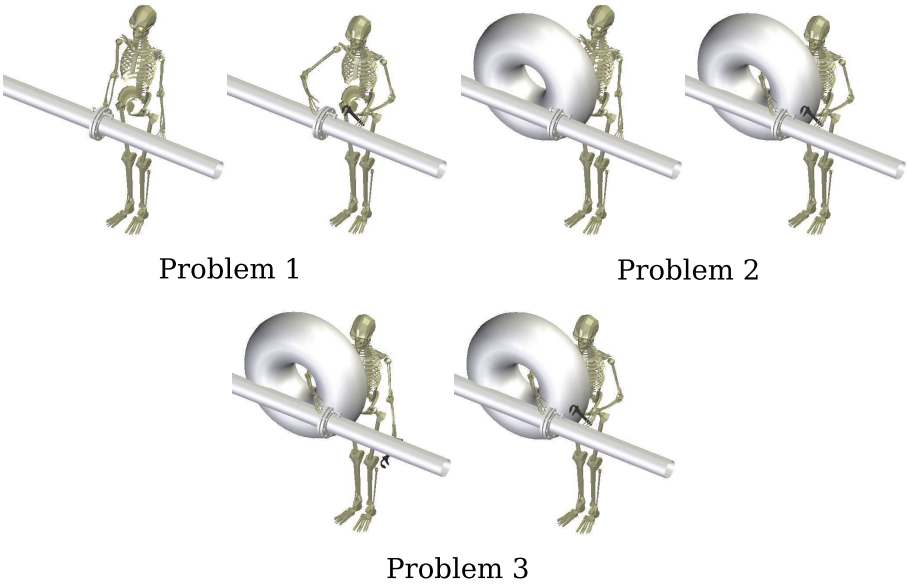

Fig. 1. Three animation motion planning problems. The mannequin has to get into position to fix the pipe. The first problem is unconstrained, we then add an obstacle for problems 2 and 3. In problem 3 the right arm is already in position and the mannequin has to find a way to move its left arm.

Fig. 1 shows the example of a mannequin getting into position to fix a pipe. In the first environment, the mannequin has to move both arms in an unconstrained environment. This problem should be easy to solve for current motion planners. In the second environment, the right hand has to go through a hole to get into position. This makes the problem more difficult, since the path to find in $\mathcal{CS}$ goes through

a narrow passage. The third problem takes place in the same environment, but the right hand is already in position. The difficulty of this last problem, from a human point of view, is more or less the same as the first one: the mannequin has to find a way to move only one of its arms in a free environment. However, the fact that the right hand is in position forces the diffusion to take place in a small volume around a sub-manifold of $\mathcal{CS}$ - the sub-manifold corresponding to a still right hand. This slows down a RRT algorithm (we will quantify it in Section 3) and motivates our work on how to identify on line when a diffusion process takes place within small volumes of $\mathcal{CS}$. Note that we want this identification to be automatic, to keep the generic properties of $\mathcal{CS}$ motion planners. This means that in the third problem of Fig. 1 we do not want an outside operator to input the information about which degrees of freedom the mannequin should use.

1.4 Contribution

This paper presents a study on how going through narrow passages slow down diffusion algorithms, and a new diffusion planner based on this study. Our adaptation of diffusion algorithms analyzes the growing tree on line to estimate if it lies in a small volume around a submanifold of $\mathcal{CS}$. If it does, the search is restrained to that volume. That way, if the tree has to follow a narrow passage, the algorithm does not loose time in trying to expend in the directions orthogonal to the passage. Thanks to the method used to describe the free space from the growing tree, the identified submanifold can be of dimension more than one, while rest of the contributions dealing with narrow passages usually consider tubes, i.e. small volumes around a one-dimensional manifold. This problem is different from the one of sampling for closed kinematic chains -presented for instance in (Han & Amato, 2000) and (Cortes *et al.*, 2002), since in our case, the subspace in which the search should take place is unknown. The new algorithm presented in this paper only uses $\mathcal{CS}$-information, which makes it more generic than algorithms that use workspace information.

Our work uses a long known and classical statistical method: the principal component analysis (PCA). It is used in many fields where subspace identification is needed, and in particular in (Teodoro *et al.*, 2002), in a motion planning context. The use of PCA there is different than ours though, since it is only applied to motion description, while we use it to control the search and generate motion. We found some recent mathematical publications (Zwald & Blanchard, 2005) that helped us understanding some convergence properties of PCA. They are shown and explained in this paper.

Our algorithm has been tried in various examples, and the experimental results are presented and analyzed. We chose to use it on different motion planning classes of problems: an mechanical disassembling problem, the animation problems of Fig. 1, and an animation problem motivated by industrial needs.

1.5 Paper Outline

We start by recalling briefly in Section 2 the structure of a widely used diffusion algorithm: RRT, before analyzing in Section 3 how narrow passages in $\mathcal{CS}$ slow

down this kind of algorithms. Section 4 presents the Principal Component Analysis method and our algorithm that uses it. Section 5 deals with several implementation issues and Section 6 presents our experimental results. We discuss in the final section the possibility of using other dimensionality reduction technique than PCA.

2 Preliminaries: Rapidly exploring Random Trees (RRT)

The method presented in this article can be applied to any diffusion algorithm. To understand what our method changes in a typical diffusion algorithm, let us remind briefly the structure of the RRT algorithm (LaValle, 1998),(Kuffner & LaValle, 2000).

Algorithm 1 shows the pseudo-code of the RRT algorithm. It takes as input an initial configuration q_0 and grows a tree $\mathcal{T}$ in $\mathcal{CS}$. At each step of diffusion, it samples a random configuration q_{rand}, finds the nearest configuration to q_{rand} in $\mathcal{T}$: q_{near}, and extends $\mathcal{T}$ from q_{near} in the direction of q_{rand} as far as possible. The new configuration q_{new} is then added to $\mathcal{T}$.

The version of RRT we consider is RRT-Connect rather than classical RRT. It often works better and is more generic, as there is no need to tune any step parameter.

Algorithm 1 RRT(q_0)

$\mathcal{T}$.Init(q_0)
for i = 1 to K **do**
  $q_{rand} \leftarrow$ Rand(CS)
  $q_{near} \leftarrow$ Nearest($q_{rand}, \mathcal{T}$)
  $q_{new} \leftarrow$ Extend(q_{near}, q_{rand})
  $\mathcal{T}$.AddVertex(q_{new})
  $\mathcal{T}$.AddEdge(q_{near}, q_{new})
end for

3 Analyzing probabilistic diffusion in narrow passages

3.1 Local description of a narrow passage

Most motion planners encounter difficulties when solving problems containing *narrow passages*. In (Hsu *et al.*, 1999), Hsu *et. al* defined the notion of expansive spaces. This formalism describes configuration spaces containing difficult areas, or narrow passages. These difficulties occur when connected areas of $\mathcal{CS}_{free}$ do not *see* large regions outside themselves. Probabilistic planners then have to sample configurations in these small volumes to solve the problem, which make the search longer.

The expansive space formalism describes the global difficulty of a configuration space. What we studied here is the local properties of $\mathcal{CS}_{free}$ that slow down a diffusing algorithm. In other terms, how can we describe locally a narrow passage?

Configuration space probabilistic planners try to find a path between a start and a goal configurations with as few samples as possible. Basically, this means that one would want new edges to be as long as possible - the accurate notion is that one would want to add nodes that expand the roadmap visibility region as much as possible.

Let q be a configuration in $\mathcal{CS}_{free}$. To describe how a diffusing tree will grow from q, one should look at its visibility region $V(q)$. If $V(q)$ is equally constrained in all directions, then diffusing algorithms will behave nicely. A RRT-Connect planner will add edges from q of length the radius of $V(q)$, which is the best one could expect. On the other hand, if $V(q)$ has an elongated shape, we can say that q lies in a narrow passage. The volume of $V(q)$ is small because of constraints in some directions. The tree will grow slowly, while it could expand easily in some other directions.

Thus, a good description of the local difficulty to expand from q is the variances and covariances of $V(q)$ along a basis of $\mathcal{CS}$. Directions accounting for most of the volume of $V(q)$ are the ones along which the tree could be expanded the furthest, while directions accounting for low volume are highly constrained and slow down a diffusing algorithm.

Remark

We use here the notion of $\mathcal{CS}-$visibility, which depends on the local method the planner uses to connect two configurations (see (Laumond & Simeon, 2001) for more comments on this). We try to describe the shape of a configuration visibility region to identify directions in which the local method goes the furthest. The technique we will use - Principal Component Analysis - is only valid for linear trajectories, since it identifies linear subspaces of $\mathcal{CS}$. Thus, the following results are given under the assumption that the local steering method is linear, this is the case in the experiments we will show.

3.2 One step of diffusion in a narrow passage

What characterizes a narrow passage is the elongated shape of a visibility region, i.e. the region has a much higher variance along some directions than along others. To understand how it slows down the expansion of a random tree, it is sufficient to look at a 2D example, where one dimension is more constrained than the other.

Fig. 2 shows a 2D example of a narrow space. Here, the visibility region of q has a rectangular shape. One of the dimension of the rectangle is fixed: L, and the other l approaches 0, which means the space around q gets narrower. The configuration towards which the tree is extended is sampled at random on a circle centered in q of radius half the length of the rectangle. Thus, the diffusion from q is isotropic, and the longest progress the tree could make starting from q is $L/2$.

In this example, we can compute the mean length $\overline{d}$ of the new edge returned by one step of diffusion. The exact value of $d(\theta)$ is indicated in Fig. 2.

$$\begin{aligned} \overline{d} &= \frac{1}{2\pi}\int_{\theta=0}^{2\pi} d(\theta)d\theta \\ &= \frac{1}{\pi}\left(L\arcsin(l/L) - l\log(\tan(\arcsin(l/L)/2))\right) \\ &\simeq_{l\to 0} -\frac{l}{\pi}\log(l) \end{aligned}$$

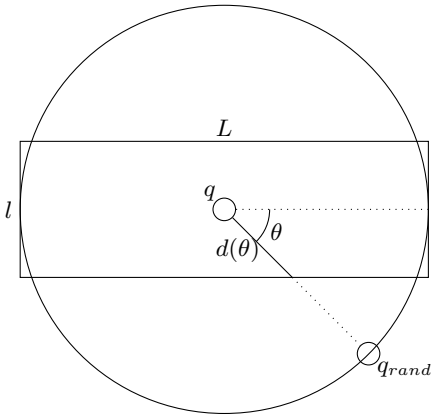

Fig. 2. One step of RRT expansion in a 2D narrow passage. L is fixed and l approaches 0. We are interested in the expected length $\overline{d}$ of the new vertex of the tree. For $|\theta[\pi]| \leq \arcsin(l/L), d(\theta) = L/2$, otherwise $d(\theta) = l/2\sin(\theta)$

As the width of the rectangle approaches 0, the mean distance the tree is extended converges to 0, even though q sees configurations far away in one of the two dimensions. This example shows how diffusion is slowed down in a narrow passage: if some directions are more constrained than others, the diffusion process does not go as fast as possible in the free directions. We used a 2D example to describe the effect of one dimension getting narrower, but the result is still valid in higher dimension. If more dimensions are constrained the diffusion gets even slower.

The important point is that what slows down a diffusion tree inside a narrow passage is the elongation of the visibility region, i.e. the difference between variances along different directions of $\mathcal{CS}$.

4 A new Planner using dimensional reduction tools

According to the previous description of narrow passages, a good way to quantify the local difficulty of $\mathcal{CS}$ in terms of probabilistic motion planning is to describe the shape of its points visibility region. Following the sampling motion planning paradigm, we do not want to describe effectively $\mathcal{CS}_{free}$. The only information that we have and will use are the vertices of the random tree already obtained.

Given a node we are about to extend the tree from, we will assume its nearest neighbours in the tree capture the local shape of the free space. We will then describe that shape by computing its variances and covariances along the canonical basis of $\mathcal{CS}$. A statistical tool, the Principal Component Analysis, is then used to determine directions in $\mathcal{CS}$ accounting for most of the variance of this region of $\mathcal{CS}_{free}$. Given that description, we change the random direction in which the tree is about to be extended to follow the directions of the passage in which the node lies.

The technical way we change the RRT algorithm is the same as in (Rodriguez *et al.*, 2006): changing q_{rand} just before extending the tree. The methods used to change the random direction, however, are completely different, since we only use $\mathcal{CS}$ information.

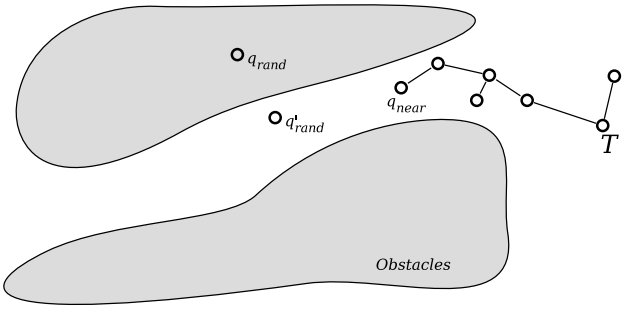

Fig. 3. Changing the random direction of extension according to $\mathcal{CS}_{free}$ local shape. Instead of extending from q_{near} to q_{rand}, we use $\mathcal{CS}$ information and extend to q'_{rand}.

Note that while ideally we would like to describe the visibility region of q_{near}, we compute the variances and covariances of a neighbourhood of q_{near} in $\mathcal{CS}_{free}$. There is no reason for those two regions of $\mathcal{CS}$ to be the same, but we assume that the later one is a good approximation of the first one. Using only nodes that q_{near} can actually "see" would take more iterations of classical RRT before being able to compute the PCA, as there is a minimal number of nodes required to compute a valid PCA.

4.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical dimensionality reduction technique. Its goal is to determine the underlying dimensionality of a set of p points in an $n-$dimensional space. PCA is one of the most commonly used technique for dimensionality reduction. It was first proposed by Pearson (Pearson, 1901) and further developed by Hotelling(Hotelling, 1933). A good description can be found in Jolliffe (Jolliffe, 2002).

This method involves a mathematical procedure that, given p points, computes an orthogonal change of basis such that the first vectors of the new basis are the directions that account for most of the variance of the input set. The underlying structure found by PCA is therefore linear.

The new basis and the corresponding variances are obtained by diagonalizing the covariance matrix of the input points. They correspond respectively to its eigenvectors and eigenvalues. The PCA Algorithm takes a set of p points in an $n-$dimensional space, computes its covariance matrix C and returns the matrices Λ and U, containing respectively the eigenvalues and the eigenvectors of C.

Algorithm 2 PCA(Points$[n][p]$)

matrix C
for i = 1 to n **do**
  $C_{ii} \leftarrow$ Variance(Points$[i]$)
end for
for i, j = 1 to n, $i \neq j$ **do**
  $C_{ij} \leftarrow$ Covariance(Points$[i]$, Points$[j]$)
  $C_{ji} \leftarrow$ Covariance(Points$[i]$, Points$[j]$)
end for
matrix Λ, matrix U
$\Lambda, U \leftarrow$ EigenSolve(C)
return Λ, U

4.2 Applying PCA to Diffusing Algorithms

The way we use PCA for diffusion algorithms does not change their structure. We only reimplemented the extension step. At each step of diffusion, we find the p nearest neighbours of q_{near} by performing a Breadth-First Search in the tree. We then compute a PCA on those points. The random direction towards which the tree should be extended (q_{rand}) is changed according to the results of the PCA, in order to favour the directions in which the variance of the growing tree is high.

More precisely, let us note $\lambda_1 > \cdots > \lambda_n > 0$ the eigenvalues of the covariance matrix, and $\mathbf{u_1}, \ldots, \mathbf{u_n}$ the corresponding eigenvectors. Placing the origin on q_{near}, the coordinates of the new random direction q'_{rand} , in the eigen basis, are given by:

$$\forall i \in [1, n], q'^{(i)}_{rand} = \frac{\lambda_i}{\lambda_1} q^{(i)}_{rand}$$

Thus, U being the eigen change of basis matrix, we get in $\mathcal{CS}$ canonical basis :

$$q'_{rand} = q_{near} + \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1} (q_{rand} - q_{near}).U_i \right) U_i$$

This transformation projects the directions with low variances towards q_{near}, while keeping the directions with high variances unchanged.

By projecting directions with low variances, the algorithm makes the diffusion process go faster in narrow spaces. Recalling the example of Fig. 2, we can show the difference between classical diffusion and PCA-controlled diffusion. Fig.4 shows how an isotropic distribution around q is transformed into an ellipsoidal distribution. We indicated in red where half of the probability stands, before and after transforming q_{rand}. One can see that with PCA-Extend, the measure is concentrated on the ends of the ellipse. That way, when the space gets narrower, the expected length of one step of diffusion does not converge to 0, but to a positive limit. The respective expected length of diffusion are plotted in Fig. 5.

Algorithm 3 PCA-Extend($q_{near}, q_{rand}, \mathcal{T}$)

vector table Points$[n][p]$
Points$\leftarrow$ BreadthFirstSearch($\mathcal{T}, q_{near}, p$)
matrix Λ, matrix U
$\Lambda, U \leftarrow$ PCA(Points)
$\lambda_{max} \leftarrow \max(\Lambda_{ii})$
$q_{rand} \leftarrow q_{near} + \sum_{i=1}^{n} \left(\frac{\Lambda_{ii}}{\lambda_{max}} (q_{rand} - q_{near}).U_i \right) U_i$
$q_{new} \leftarrow$ RRT-Extend(q_{near}, q_{rand})
return q_{new}

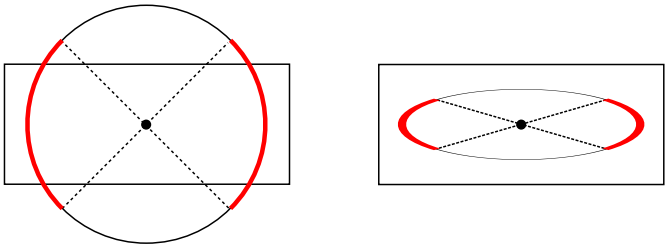

Fig. 4. One step of diffusion in a narrow passage. One the left, RRT isotropic diffusion and on the right PCA-RRT diffusion, concentrated along the direction of the passage. The bold red part shows where half of the measure is concentrated.

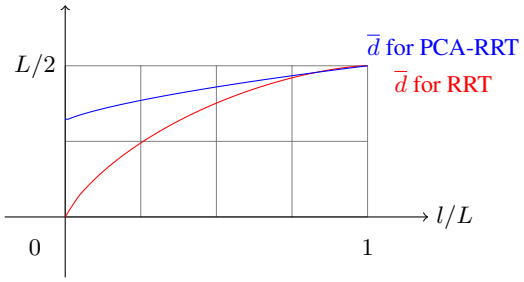

Fig. 5. Expected length of diffusion in a narrow passage. The curves have to be looked at from right to left. As the space gets narrower, the RRT (in red) converges to a null extension, while PCA-RRT (in blue) converges to a positive limit.

4.3 Accuracy of the PCA description

The previous sections explained how knowledge about the local shape of $\mathcal{CS}_{free}$ could help improving the performance of a diffusion algorithm. However, a key point is the accuracy of this knowledge. We said that we assume the nearest neighbours of a configuration catch the shape of the free space. Two questions remain: how true is this assumption and how many neighbours are necessary to describe locally $\mathcal{CS}_{free}$?

The following result answer these questions. It is a simplified version of mathematical results found in (Zwald & Blanchard, 2005). The accuracy of PCA descrip-

tion is expressed as the distance δ between the high variance subspace found by PCA and the real high variance subspace $\mathcal{CS}_{free}$ lies around. The distance between the two subspaces is expressed as the distance between the projectors on the subspaces, using the Frobenius norm. Let D be the dimension of the high variance subspace $\mathcal{CS}_{free}$ lies around, and $\lambda_1 > \cdots > \lambda_n > 0$ the eigen values of the local covariance matrix of $\mathcal{CS}_{free}$.

Theorem 1. *For p points lying in a ball of radius r, for $\xi \in [0,1]$, the following bound holds with probability at least $1 - e^{-\xi}$:*

$$\delta \leq \frac{4r^2}{\sqrt{p}(\lambda_D - \lambda_{D+1})}\left(1 + \sqrt{\frac{\xi}{2}}\right) \tag{1}$$

This theorem is a simpler rewriting of Theorem 4 in (Zwald & Blanchard, 2005)[3]. It illustrates two points: the speed of convergence of the PCA analysis with respect to the number of points is $1/\sqrt{p}$, and the narrower the free space is (i.e. the higher $(\lambda_D - \lambda_{D+1})$ is), the more accurate the PCA description will be.

Note that if the space is unconstrained (all the λ_i are close to each other), this bound is not interesting. In that case, PCA may return imprecise results, but then again, traditional diffusion algorithms behave properly. We will discuss in the next section how not to suffer from PCA computation in a locally unconstrained space.

5 Implementation details

5.1 PCA Bias in unconstrained spaces

As shown earlier, the computation of PCA will give accurate results in narrow passages. However, if the free space is locally unconstrained - or equally constrained in all directions - the description returned by PCA may not be accurate enough. This could induce a bias in the algorithm: if the space is free, the algorithm may favour expansion in directions that have previously been sampled at random. This is a main drawback since the new diffusion algorithm loses one of the key properties of the RRT algorithm: the convergence of the RRT vertices distribution towards the sampling distribution.

To overcome this bias, our implementation of PCA-RRT uses, at each step of expansion, PCA-Extend with probability 0.5 and classical RRT Extend with probability 0.5. That way, the tree still covers densely the free space, while the PCA-Extend function accelerates the search in narrow passages.

[3] Note to the reviewers: we did not consider as useful to develop the underlying mathematics. However, you can find, for review purpose, the original theorem in Annex 1.

5.2 Recursive PCA

The previous section presented ideas about the necessary number of points to correctly describe the space. It depends on each region of the space, narrow passages needing less points than free regions. As we do not want that tuning to be done by a user, but *automatically* by the algorithm, we need a version of the algorithm where the number p of nearest neighbours used to compute the PCA is no longer a parameter. It is done by evaluating the PCA result each time we add a new neighbour: we start by computing the PCA on $(n+1)$ points, so that all the eigenvalues of the covariance matrix are positive, then we update that computation by adding the points one by one.

We chose to use the bound given by Theorem 1 as an indicator on the accuracy of the successive computations of PCA. At one step of the computation of recursive PCA, given the estimate on the eigenvalues $\lambda = (\lambda_1, \ldots, \lambda_n)$, the number of points used p, and the maximum distance between two points r we can compute the bound of Theorem 1 for every D between 1 and $(n-1)$. Let $f_D(\lambda, p, r) = \frac{4r^2}{\sqrt{p}(\lambda_D - \lambda_{D+1})}$ (see Eq. 1).

The Frobenius norm of the projector on the high variance subspace of dimension D is D, so the relative error made on evaluating that subspace is δ/D.

For a given ϵ, if $f_D(\lambda, p, r)/D \leq \epsilon/1.707$ (choosing $\xi = 1$ in Eq. 1), we get, with probability at least $1 - e^{-1}$, $\delta/D \leq \epsilon$. In our implementation, we chose $\epsilon = 0.1$ as a convergence criterion: if for some $D \in [1, n-1]$, $f_D(\lambda, p, r)/D \leq 0.059$, we stop the recursive PCA computation and use the results as they are.

Remark

The Theorem 1 uses the real eigenvalues of the local covariance matrix of $\mathcal{CS}_{free}$, while we only have access to an estimate of it. Therefore, we can not guarantee that the bound is valid. We still found that this convergence criterion was both theoretically founded and experimentally efficient. Fig. 6 shows an example of the successive evaluation of the convergence criterion for a robot with 20 degrees of freedom. The criterion is met when the PCA uses 50 points to describe $\mathcal{CS}_{free}$. This data is actually taken from a trial of the PCA-RRT algorithm on the example shown on Fig. 1, Problem 3.

This whole computation needs an incremental version of the PCA, in order not to recompute the diagonalization of the covariance matrix at each step. A good description of incremental PCA can be found in (Erdogmus *et al.*, 2004). It uses matrix perturbation theory, and basically computes a rank one update on the PCA computation each time we add a point[4].

[4] Note to the reviewers: the algorithm behind recursive PCA is well explained in the cited paper, and we were able to implement it without trouble. Therefore, we did not find relevant to go over it here. However, you can find it, for review purpose, in Annex 2.

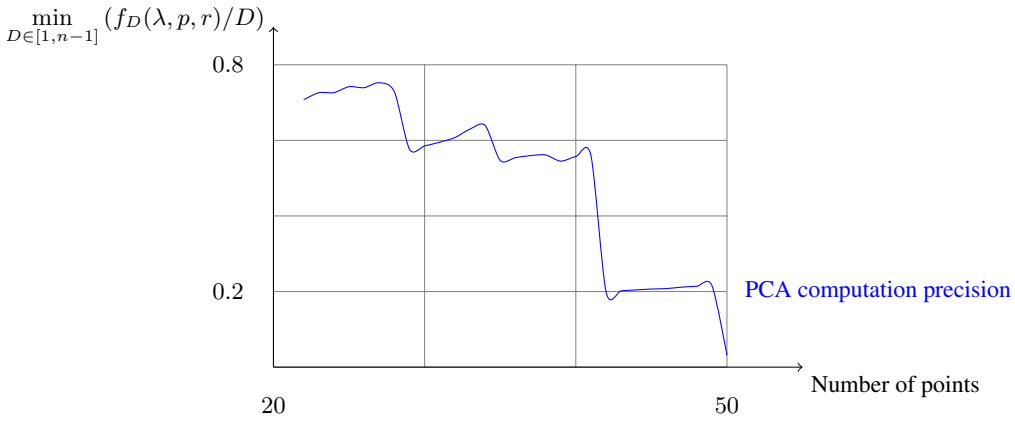

Fig. 6. Convergence of the estimate of the distance between the real high variance subspace of $\mathcal{CS}_{free}$ and the subspace identified by the PCA. The computation stops when that estimate reaches a lower threshold, which happens for 50 points. The data is taken from a trial of the PCA-RRT algorithm on the example shown on Fig. 1, Problem 3.

5.3 Complexity

The complexity of the algorithm at each step of diffusion depends only on two parameters : the dimension of the space n and the number p of nearest neighbours used to do the PCA.

The computation of the original PCA, with $(n+1)$ points is done in $O(n^3)$ operations. Afterwards, each step of recursive PCA involves one $(n \times n)$-matrix product, so the overall complexity is $O(n^3 \times p)$. Unfortunately, we have no way of predicting the value of p. We will show in the experimental section its average value for the considered problems, as well as the relative costs of PCA computation and collision detection.

6 Experimental Results

In this section, we compare our PCA-RRT algorithm with classical RRT. The algorithm has been implemented in KineoWorks^{TM}(Laumond, 2006), we used its implementation of RRT as a reference. The default expansion method used in KineoWorks is RRT-Connect. Since our expansion method only uses CS information, it can be useful in any motion planing problem, as long as the configuration space contains narrow passages that slow down classical diffusion methods. To show the range of uses of our method, we present results in various motion planning problems: an industrial disassembly problem, the three computer animation problems presented in introduction, and an industrial animation problem. In a previous version of this paper (Dalibard & Laumond, 2009), the PCA-RRT algorithm was compared to OBRRT (Rodriguez *et al.*, 2006) on two motion planning benchmarks. Because of differences

in the RRT implementations and in the computers used to do the tests, computation time comparison is not relevant. (Dalibard & Laumond, 2009) presented a comparison based only on the number of iterations needed by the motion planner to solve the benchmark. PCA-RRT generally required more iterations than OBRRT, however, one should notice that PCA-RRT *does not need any tuning*, while the results of OBRRT taken from (Rodriguez *et al.*, 2006) correspond to a tuned version, adapted to each problem. We chose to focus here on the computation gain between classical RRT and PCA-RRT.

All the experiments presented here were performed on a 2.13 GHz Inter Core 2 Duo PC with 2 GB RAM. Each method was tested a hundred times on each environment and we show the average results of all test runs. The initial and final configurations were fixed for each problem. The results report the time taken by each method, the number of expansion steps and the number of nodes needed. We have also indicated the average length of each expansion step, in an arbitrary unit, to show the benefit of PCA-RRT in narrow passages.

For each trial, we stopped the motion planner after one million iterations if it had not found a path. The results indicate the percentage of failed trials. One should note that for a given problem and planner, the higher this percentage is, the more we underestimate the computation time average.

6.1 Industrial Disassembly Problem

This environment (Fig. 7) is the back of a car. The robot, the exhaust, must be disassembled. This industrial environment was provided as a motion planning benchmark with KineoWorks^{TM}. The initial configuration is the mounted position, and the random tree must get out of the narrow passage to the unmounted position. For that problem both RRT and PCA-RRT found a solution every time. Fig. 7 shows the problem, and a solution path.

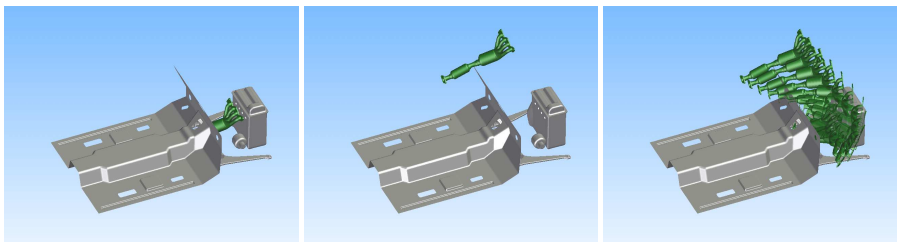

Fig. 7. Exhaust disassembly problem. Start and goal configurations and solution path.

Tab. 1 shows the average results over one hundred trials. The PCA used in average 14.5 nodes to describe locally $\mathcal{CS}_{free}$. For the PCA-RRT algorithm, the PCA cost 1% of the computation time, and collision detection 93%. Because the dimension of $\mathcal{CS}$ is small (6), the PCA does not cost a lot compared to collision detection.

The gain of using PCA, however is not very significant in terms of number of iterations, since dimensionality reduction does help much on a 6-dimensional space. On the following examples, PCA will cost much more relatively to collision detection, but will help in finding solution paths in very few iterations.

	Iterations	Nodes	Average Computation Time	Standard Deviation	Distance
RRT	7748.3	5241.2	272.239 s	87.855 s	96.5
PCA-RRT	4725.2	3362.2	196.405 s	82.978 s	118.3

Table 1. Experimental results for the exhaust disassembling problem.

6.2 Animation Problems

This algorithm was first designed to overcome the limitations of traditional motion planing algorithms. Among others, highly dimensioned problems are hard to solve for diffusion algorithms. The problems presented in Fig. 1 deal with a rather highly dimensioned robot: the upper body of a virtual character. The robot can move its torso, head and both arms. These adds up to 20 degrees of freedom. It should get into position to fix a pipe. Its left hand holds a wrench and its right hand should hold a bolt that lies into a hole, inside a large torus shaped obstacle. The obstacle is represented by 1,000 triangles. The difficulties in that environment come from the narrow space the right hand has to go through, the dimension of $\mathcal{CS}$ and the collisions caused by the arms and wrench. We considered the two problems presented in the introduction in the following order:

- One where the character's right hand is already inside the hole. The motion planner has to find a path around a linear submanifold of $\mathcal{CS}$.
- One where the character has to move both hands to get into position. The most difficult part is finding a way for the right hand. Here, the narrow passage to find in $\mathcal{CS}_{free}$ is non-linear.

Right hand already positioned

For the first set of problems, the right hand is already inside the hole. The motion planner has to find a path to get the right hand in position. To do so, the random diffusion algorithm has to find a path around a linear submanifold of $\mathcal{CS}$.

To evaluate how the advantage of PCA is related to the narrowness of the passage, we varied the size of the hole the right hand is in. The first experiment is done without an obstacle, which means the right hand degrees of freedom hardly slow down the path finding process for the left hand, than the size of the hole is successively 24 cm, 16 cm, 12 cm and 8 cm. The initial and final configurations do not change for the different problems. Fig. 8 shows the different environments, from the easiest to the hardest.

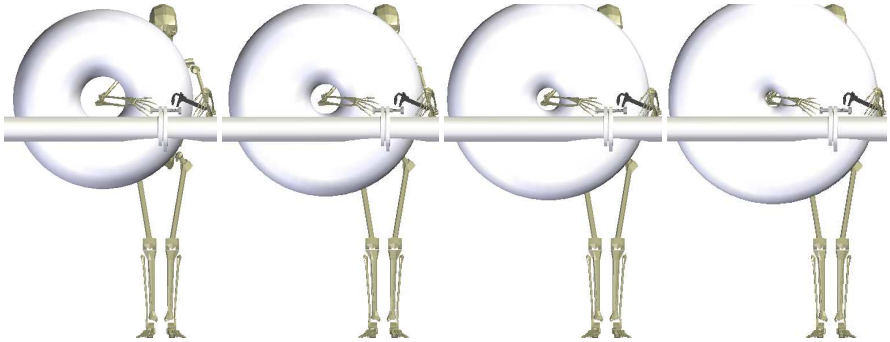

Fig. 8. Different scenarios where the diffusion process gets more difficult because of the narrowness of the hole in which the right hand is.

Fig. 9 shows a solution path for this problem. The results of PCA-RRT and RRT on those problems are indicated in Table 2 and 3. The average number of nodes used by the PCA is 32.2. It varies slightly between 31.6 for the easiest problem and 33.1 for the most difficult. For these problems, the PCA computation represents 80% of the total computation time, the rest of it being mainly collision detection. That rate does not vary significantly over the different problems. The PCA-RRT algorithm always found a path for these problems. On the other hand, classical RRT often failed to find a path in less than 1,000,000 iterations. We indicated the failure rate of the algorithm for each problem. One should keep in mind that the higher that rate is, the more the estimate on the average computation time is an underestimate of the real average computation time.

One can see that PCA-RRT outperforms classical RRT for every problem, as expected on those benchmarks. An interesting point is that as the hole gets narrower,

Fig. 9. Solution path for the first problem. The character only has to move its left hand.

Size of the hole	Iterations	Nodes	Average Computation Time	Standard Deviation	Distance
No obstacle	2027.2	119.7	11.823 s	16.298 s	8.86
24 cm	11,643	206.8	78.138 s	124.667 s	2.62
16 cm	12,801	184.7	87.276 s	150.817 s	2.59
12 cm	17,714	198.5	124.376 s	278.344 s	1.57
8 cm	29,393	217.5	222.273 s	417.551 s	1.27

Table 2. Results of PCA-RRT for the first set of animation problems. The average number of nodes used by the PCA is 32.2.

Size of the hole	Iterations	Nodes	Average Computation Time	Standard Deviation	Distance	Failure Rate
No obstacle	54,345	190.3	137.696 s	463.852 s	1.09	3%
24 cm	199,816	186.5	432.112 s	799.452 s	0.352	16%
16 cm	213,819	173.9	458.903 s	798.752 s	0.254	16%
12 cm	296,811	180.0	609.035 s	854.029 s	0.152	23%
8 cm	317,740	157.5	639.031 s	874.861 s	0.0982	26%

Table 3. Results of RRT for the first set of animation problems

the average gain of using the PCA increases. Fig. 10 shows the evolution of the ratio between the average distance between two nodes with PCA-RRT and classical RRT on this set of problems. Recalling the analysis summarized by Fig. 5 we should expect that ratio to diverge to infinity when the thickness of the narrow passage approaches 0 in $\mathcal{CS}$.

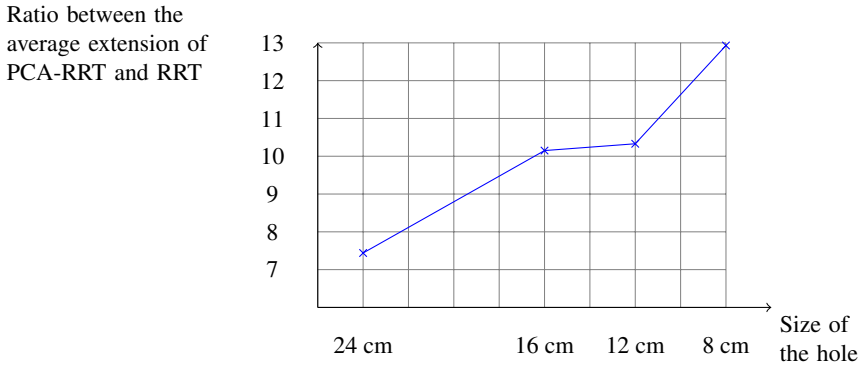

Fig. 10. Evolution of the ratio between the average distance between two nodes with PCA-RRT and classical RRT. As the hole gets smaller, the passage in $\mathcal{CS}$ gets narrower and the gain of using PCA for diffusion is higher.

Two hand motion

The environment is the same as before, but this time, the character has to get both hands into position. For this problem, we used a bi-RRT -i.e. one tree is rooted at the initial configuration and an other one at the goal configuration. That way, the difficulty is not to find the entrance of the narrow passage - the final configuration is inside the passage, but to follow it. The path to find in $\mathcal{CS}$ is non linear. Fig. 11 shows a solution path for this problem.

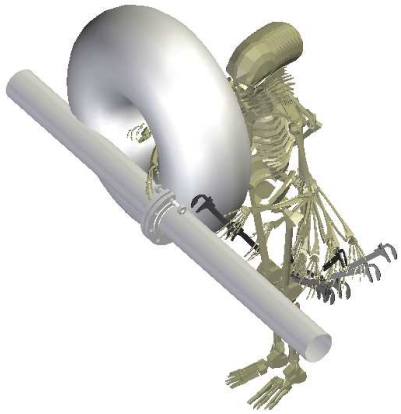

Fig. 11. Solution path for the second problem. The character has to move both hands.

We ran the classical RRT algorithm a hundred times on the problem where the hole size is 24 cm and it never found a solution in less than 1,000,000 iterations. We did not find relevant to make the hole smaller, as we did on the previous problem. The results of both algorithms are shown on Tab. 4 and Tab. 5. The average number

Size of the hole	Iterations	Nodes	Average Computation Time	Standard Deviation	Distance	Failure Rate
No obstacle	1,167	136.8	5.535 s	3.192 s	1.01	0%
24 cm	35,981	4,938	541.437 s	546.243 s	0.56	6%

Table 4. Results of PCA-RRT for the second set of animation problems. The average number of points used by the PCA is 33.6.

Size of the hole	Iterations	Nodes	Average Computation Time	Standard Deviation	Distance	Failure Rate
No obstacle	4,774	244.8	10.074 s	6.104 s	0.13	0%
24 cm	1,000,000	653.7	2,312.744 s	146.540 s	0.012	100%

Table 5. Results of RRT for the second set of animation problems

of points used by the PCA is 33.6. Its computation cost is 70% of the total computation time. Note that for the problem without the torus, the algorithms perform better than when the right hand was already in position thanks to the bi-RRT. The standard deviation of the computation time for the classical RRT is irrelevant, since the algorithm was always stopped at the same number of iterations, before finding a solution. For that problem, the ratio between the extension length of classical RRT and PCA-RRT jumps to almost 50.

Industrial animation problem

This problem was given to us in the scope of the French ANR-RNTL project PerfRV2 by Dassault Aviation. The picture on Fig. 12 and the corresponding 3D model of the plain hold is their courtesy.

In that problem, two characters have to cooperate to fix a part of a plain hold. Getting into position is easy for the character above the hold, but difficult for the character underneath it. We considered the motion planning problem consisting in moving the upper part of the body of the character beneath the hold. As in the previous animation benchmarks, the robot has 20 degrees of freedom. Fig. 12 shows the 3D model of the hold and of the virtual characters. On the left is the initial configuration of the mannequin underneath the hold and on the right its final configuration. Fig. 13 shows a solution path for that problem.

The results of PCA-RRT and classical RRT on that problem are shown in Tab. 6 and 7. As in the last animation benchmark, the path to follow is not linear, but the use of PCA speeds up the diffusion process a lot, because of the presence of narrow passages. The PCA used in average 32.4 points to describe CS_{free}. The computation of PCA represented 88% of the total computation time.

Iterations	Nodes	Average Computation Time	Standard Deviation	Distance	Failure Rate
186,769	2602	1,406.818 s	868.773 s	1.07	34%

Table 6. Results of PCA-RRT for the industrial animation problem. The average number of points used by the PCA is 32.4.

Iterations	Nodes	Average Computation Time	Standard Deviation	Distance	Failure Rate
870,852	1424	2,484.212 s	786.087 s	0.0402	72%

Table 7. Results of RRT for the industrial animation problem.

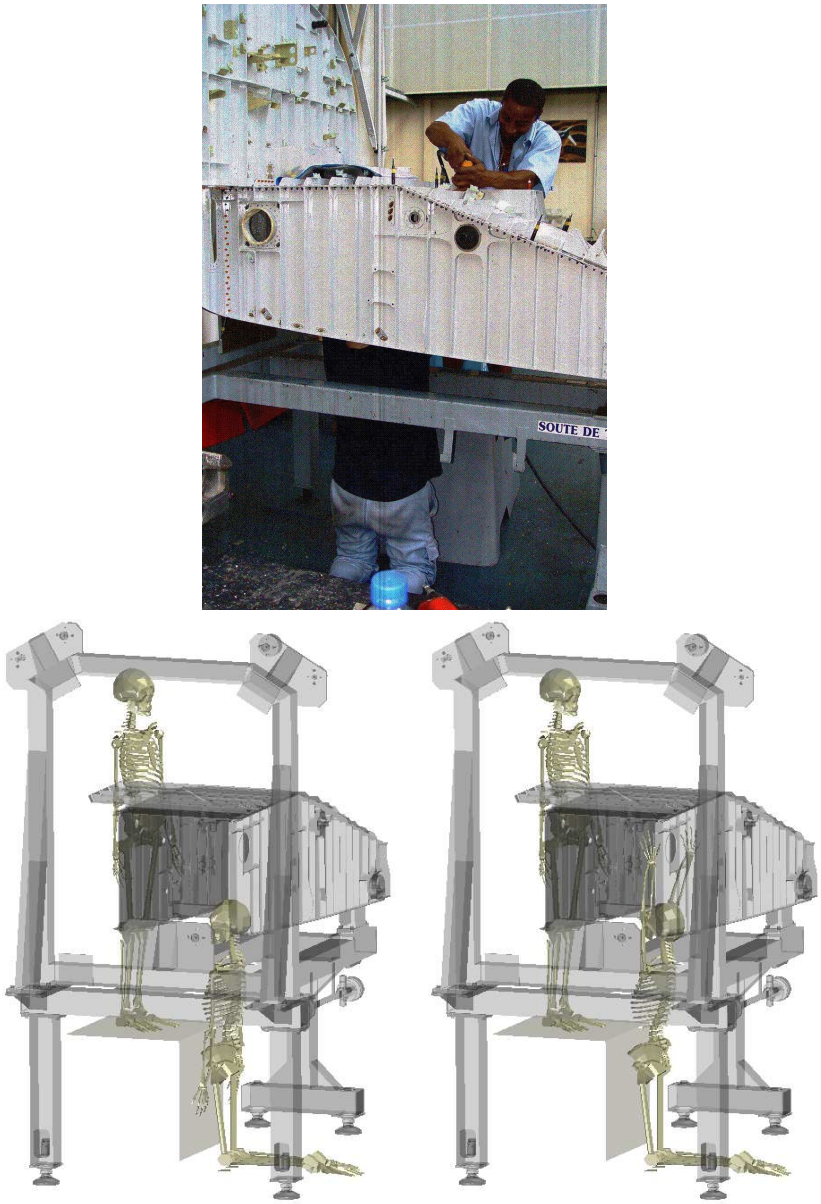

Fig. 12. Motion planning problem for a virtual actor. The main difficulty is to place both hands of the character beneath the plain hold. Picture and 3D model of the plain hold are under a Copyright of Dassault Aviation. They were given to us in the scope of the French ANR-RNTL project PerfRV2.

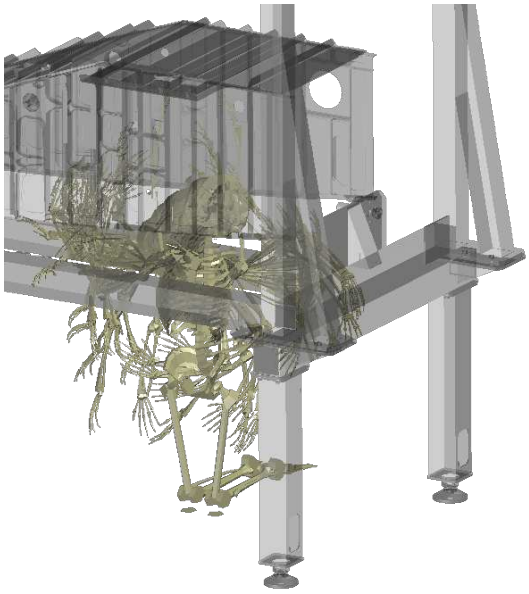

Fig. 13. Solution path for the plain hold problem.

7 Discussion on the choice of the dimensionality reduction technique

7.1 Non-linear steering methods

As we said in the introduction, the method presented earlier is only valid for linear local methods. If the motion planner does not connect configurations by straight lines - for instance if we are dealing with car-like robots, the sets of free trajectories starting from a given point of $\mathcal{CS}$ can not be well approximated by linear subspaces.

In order to use the same algorithm, we need to do the linearization in a mapping of $\mathcal{CS}$, in which the local paths are mapped into straight lines. This can be done by using Kernel-PCA. The classical computation of PCA can be done using only the euclidean inner product rather than the variance and covariance matrix. If the steering method we are using defines a distance - as it is the case for Reeds and Shepp curves (Reeds & Shepp, 1990), we can define what would be the inner product corresponding to that distance in the mapping of $\mathcal{CS}$ and use it to compute the KPCA. This method is valid if there is actually a space where our operator is an inner product. Intuitively, that means that the notions of orthogonality and alignment are valid for the local method we consider. For a good description of Kernel-PCA, one can refer to (Schoelkopf *et al.*, 1997).

We have not shown that Reeds and Shepp curves actually define an inner product, but we have used KPCA with what would be the Reeds and Shepp inner product, and it showed better results that naive PCA on some examples. We chose not to

present these results because for car-like robots, i.e. devices with only three degrees of freedom, there is no need for dimensionality reduction techniques and classical RRT outperformed PCA-RRT as well as KPCA-RRT.

7.2 Non-linear dimensionality description of the space

In the work presented here, we chose PCA as our dimensionality reduction technique because it is easy to implement and provides good results. The main limitation of this method is that it is globally linear. The subspaces found by PCA are therefore linear, and could be of higher dimension than necessary if the true underlying structure of the data is not linear.

Some methods for non-linear dimensional reduction have been proposed to overcome this limitation. Some of them are widely used in fields where dimensionality reduction is a critical issue, for instance the isomap method, proposed by Tenenbaum *et. al* (Tenenbaum *et al.*, 2000), or the locally linear embedding method, proposed by Roweis and Saul (Roweis & Saul, 2000).

Several cases can appear while using PCA to control random diffusion in motion planning:

1. The sub-manifold to find is actually linear. This is the case for the animation benchmark where the right hand of the character was in the right position in the initial configuration: some of the degrees of freedom should not be moved (the ones corresponding to the right arm), which forces the diffusion to take place in an intersection of hyperplans, i.e a linear submanifold of $\mathcal{CS}$. In that case, using PCA as our dimensionality reduction technique is appropriate. These are the cases we first had in mind while designing the algorithm.
2. The sub-manifold $\mathcal{CS}$ lies around is non-linear but is smooth. Because our use of the PCA is local, the result it will give can be seen as the tangent space of that manifold rather than the manifold itself. PCA then helps solving the motion planning problem, as some of our experiments showed: the exhaust disassembly problem, and more remarkably, the animation benchmarks with the torus shaped obstacle. In these last examples, we even gained more with PCA in the second problem, where the passage to follow was non-linear than in the first one, where the free space lied around a linear submanifold of $\mathcal{CS}$.
3. The free space makes a right angle (i.e. goes from a direction to an other orthogonal one). In that case, the PCA description does not help at all. However, the fact we use classical RRT every other step guaranties us that we will find the turn. Once the turn is found the new points will be taken into account in the PCA description which will make our algorithm deal equally with the two orthogonal directions at the point where the free space makes a right angle. Thus, our technique does not slow down the diffusion -except for the useless PCA computations- in terms of number of steps of diffusion.
4. The free space is a T-shape tunnel. The use of the PCA induces an inertia in the straight line direction, and the exploration will go faster in that direction than in the orthogonal one. In a sense, our method could slow down the RRT algorithm

here if the path to find takes the turn and the straight line is a dead-end. This is an other reason why we use classical RRT half of the iterations: the turn will be found with probability one and not that slower than the classical RRT would do.

8 Conclusion

An adaptation of randomized diffusion motion planning algorithms is proposed. The statistical method we used, PCA, is well-known and widely used in literature, but not yet to control motion planning algorithms. Using dimensional reduction tool in diffusion planning, and analyzing the result of it is the contribution of this work. As expected, we observed that this technique could significantly improve diffusion algorithm performances in constrained environments. Note that it does not solve the problem of finding the entrance of a narrow passage, but only accelerate the diffusion within the passage. Our method has been used on several motion planning problems, coming from different domains of application, and has shown good results. Its main strength is the fact that it does not need any tuning and adapts itself automatically to the difficulty and dimension of the problem. The class of problems where we found it to be the most useful are highly dimensioned animation problems in confined and complex environments.

References

[Burns & Brock, 2005]Burns, B., & Brock, O. 2005. Toward optimal configuration space sampling. *In: Robotics: Science and Systems I*.

[Choset *et al.*, 2005]Choset, H., Lynch, KM, Hutchinson, S., Kantor, GA, Burgard, W., Kavraki, LE, & Thrun, S. 2005. *Principles of Robot Motion: theory, algorithms, and implementation*. MIT Press.

[Cortes *et al.*, 2002]Cortes, J., Simeon, T., & Laumond, JP. 2002. A random loop generator for planning the motions of closed kinematic chains using PRM methods. *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, **2**.

[Dalibard & Laumond, 2009]Dalibard, S., & Laumond, J.P. 2009. Control of probabilistic diffusion in motion planning. *Algorithmic Foundation of Robotics VIII*, 467–481.

[Erdogmus *et al.*, 2004]Erdogmus, Deniz, Rao, Yadunandana N., Peddaneni, Hemanth, Hegde, Anant, & Principe, Jose C. 2004. Recursive Principal Components Analysis Using Eigenvector Matrix Perturbation. *EURASIP Journal on Applied Signal Processing*, **2004**(13), 2034–2041.

[Geraerts & Overmars, 2004]Geraerts, R., & Overmars, M.H. 2004. Sampling techniques for probabilistic roadmap planners. *In: Proceedings International Conference on Intelligent Autonomous Systems*.

[Han & Amato, 2000]Han, L., & Amato, N.M. 2000. A kinematics-based probabilistic roadmap method for closed chain systems. *Pages 233–245 of: Algorithmic and Computational Robotics: New Directions. The Fourth Workshop on the Algorithmic Foundations of Robotics*.

[Hotelling, 1933]Hotelling, H. 1933. Analysis of a Complex of Statistical Variables into principal components, J. Ed. *Psychology*, **24**, 417–441.

[Hsu *et al.*, 1999]Hsu, D., Latombe, J.C., & Motwani, R. 1999. Path planning in expansive configuration spaces. *Int. J. Computational Geometry & Applications*, **9**(4-5), 495–512.

[Hsu *et al.*, 2006]Hsu, D., Latombe, J.C., & Kurniawati, H. 2006. On the Probabilistic Foundations of Probabilistic Roadmap Planning. *The International Journal of Robotics Research*, **25**(7), 627.

[Jolliffe, 2002]Jolliffe, I. T. 2002. *Principal Component Analysis*. Springer.

[Kavraki *et al.*, 1996]Kavraki, LE, Svestka, P., Latombe, J.C., & Overmars, MH. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, **12**(4), 566–580.

[Kavraki *et al.*, 1998]Kavraki, LE, Kolountzakis, MN, & Latombe, J.C. 1998. Analysis of probabilistic roadmaps for path planning. *Robotics and Automation, IEEE Transactions on*, **14**(1), 166–171.

[Kuffner & LaValle, 2000]Kuffner, J., & LaValle, S. 2000 (Apr.). RRT-Connect: An efficient approach to single-query path planning. *In: Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'2000), San Francisco, CA*.

[Latombe, 1991]Latombe, J.C. 1991. *Robot Motion Planning*. Kluwer Academic Publishers.

[Laumond, 2006]Laumond, J.P. 2006. Kineo CAM: a success story of motion planning algorithms. *Robotics & Automation Magazine, IEEE*, **13**(2), 90–93.

[Laumond & Simeon, 2001]Laumond, JP, & Simeon, T. 2001. Notes on Visibility Roadmaps and Path Planning. *Algorithmic and Computational Robotics: New Directions: the Fourth Workshop on the Algorithmic Foundations of Robotics*.

[LaValle, 1998]LaValle, S. M. 1998 (Oct.). *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rept. 98-11. Computer Science Dept., Iowa State University.

[LaValle, 2006]LaValle, S. M. 2006. *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press. Available at http://planning.cs.uiuc.edu/.

[Pearson, 1901]Pearson, K. 1901. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, **2**(6), 559–572.

[Reeds & Shepp, 1990]Reeds, JA, & Shepp, LA. 1990. Optimal pathsfor a car that goesboth forwards and backwards. *Pacific Journal of Mathematics*, **145**(2).

[Rodriguez *et al.*, 2006]Rodriguez, S., Tang, X., Lien, J.M., & Amato, NM. 2006. An obstacle-based rapidly-exploring random tree. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 895–900.

[Roweis & Saul, 2000]Roweis, Sam T., & Saul, Lawrence K. 2000. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, **290**(5500), 2323–2326.

[Schoelkopf *et al.*, 1997]Schoelkopf, B., Smola, A.J., & Mueller, K.R. 1997. Kernel Principal Component Analysis. *Lecture notes in Computer Science*, 583–588.

[Siméon *et al.*, 2000]Siméon, T., Laumond, J.P., & Nissoux, C. 2000. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, **14**(6), 477–493.

[Tenenbaum *et al.*, 2000]Tenenbaum, Joshua B., Silva, Vin de, & Langford, John C. 2000. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, **290**(5500), 2319–2323.

[Teodoro *et al.*, 2002]Teodoro, Miguel L., George N. Phillips, Jr., & Kavraki, Lydia E. 2002. A dimensionality reduction approach to modeling protein flexibility. *Pages 299–308 of: RECOMB '02: Proceedings of the sixth annual international conference on Computational biology*. New York, NY, USA: ACM Press.

[Yershova *et al.*, 2005]Yershova, A., Jaillet, L., Simeon, T., & LaValle, SM. 2005. Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain. *Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on*, 3856–3861.

[Zwald & Blanchard, 2005]Zwald, Laurent, & Blanchard, Gilles. 2005. On the Convergence of Eigenspaces in Kernel Principal Component Analysis. *In: Neural Information Processing Systems*.

Appendix 1: about PCA convergence (for review purpose)

The results presented here have been found in (Zwald & Blanchard, 2005). We present here simplified versions.

The interest variable X takes its value in a measurable space $\mathcal{X}$, following the distribution P. We have access to n realizations of X, on which we compute a PCA. Let C be the covariance matrix of P, and C_n the empirical covariance (measured after n realizations). We are also interested in the subspace of dimension D spanned by the eigenvectors associated with the D largest eigenvalues of C. Let S_D be that space and P_{S_D} the orthogonal projector on that space. Its empirical equivalent (obtained with C_n) is $\widehat{S_D}$ (respectively $P_{\widehat{S_D}}$).

The following results deal with the convergence speed of C_n to C and $P_{\widehat{S_D}}$ to P_{S_D}.

Lemma 1. *Supposing that there exists M such that for all $x \in \mathcal{X}$, $||x||^2 \leq M$, then for all $\xi > 0$, with probability greater than $1 - e^{-\xi}$,*

$$||C_n - C|| \leq \frac{2M}{\sqrt{n}} \left(1 + \sqrt{\frac{\xi}{2}}\right)$$

Theorem 2. *Assume that there exists M such that for all $x \in \mathcal{X}$, $||x||^2 \leq M$. Denoting $\lambda_1 > \lambda_2 > \dots$ the eigenvalues of C, if $D > 0$ is such that $\lambda_D > 0$, put $\delta_D = (\lambda_D - \lambda_{D+1})/2$ and*

$$B_D = \frac{2M}{\delta_D} \left(1 + \sqrt{\frac{\xi}{2}}\right)$$

Then provide that $n \geq B_D^2$, for all $\xi > 0$, the following bound holds with probability at least $1 - e^{-\xi}$:

$$\left|\left|P_{\widehat{S_D}} - P_{S_D}\right|\right| \leq \frac{B_D}{\sqrt{n}}$$

Appendix 2: about recursive PCA (for review purpose)

We found several ways to compute PCA recursively in literature. The one presented here was found in (Erdogmus *et al.*, 2004). The following calculus shows how to

diagonalize the covariance matrix of $(k+1)$ points when we have done it already for k points.

Assume we have already computed a PCA on k points $(\mathbf{x_i})_{i=1..k}$. We have measured $\mathbf{x_{k+1}}$ and we want to update the result of the PCA. The new covariance C_{k+1} is equal to:

$$C_{k+1} = \frac{1}{k+1}\sum_{i=1}^{k+1} \mathbf{x_i}\mathbf{x_i}^T = \frac{k}{k+1}C_k + \frac{1}{k+1}\mathbf{x_{k+1}}\mathbf{x_{k+1}}^T \tag{2}$$

We have already diagonalized C_k and we want to diagonalize C_{k+1}. Let $C_k = Q_k \Lambda_k Q_k^T$ and $C_{k+1} = Q_{k+1}\Lambda_{k+1}Q_{k+1}^T$, where Q and Λ denote the orthonormal eigenvector and diagonal eigenvalue matrices, respectively. Also define $\mathbf{y_{k+1}} = Q_k^T \mathbf{x_{k+1}}$. Substituting these notations in 2, we get:

$$Q_{k+1}\Lambda_{k+1}Q_{k+1}^T = Q_k\left[\frac{k}{k+1}\Lambda_k + \frac{1}{k+1}\mathbf{y_{k+1}}\mathbf{y_{k+1}}^T\right]Q_k^T \tag{3}$$

Thus, we just have to diagonalize $\frac{k}{k+1}\Lambda_k + \frac{1}{k+1}\mathbf{y_{k+1}}\mathbf{y_{k+1}}^T$ to get Q_{k+1} and Λ_{k+1}. Denoting

$$\frac{k}{k+1}\Lambda_k + \frac{1}{k+1}\mathbf{y_{k+1}}\mathbf{y_{k+1}}^T = VDV^T$$

where D is diagonal and V orthogonal, we get:

$$\begin{cases} Q_{k+1}{=}Q_kV \\ \Lambda_{k+1}{=}\ D \end{cases}$$

The matrix to be diagonalized has a very specific structure, it is the sum of a diagonal matrix and a very small matrix compared to the diagonal one. The following calculus shows how to approximate the result of this diagonalization with rank one approximations on $\mathbf{y_{k+1}}\mathbf{y_{k+1}}^T$.

We need to diagonalize $(\Lambda + \alpha\alpha^T)$ where $\alpha\alpha^T << ||\Lambda||$. Let $(\Lambda + \alpha\alpha^T) = VDV^T$. Because the matrix to be diagonalized is close to Λ, denote $D = \Lambda + P_\Lambda$ and $V = I + P_V$, where P_Λ and P_V are small perturbation matrices. With these definitions, expanding VDV^T gives:

$$\begin{aligned} VDV^T{=}& \quad (I+P_V)(\Lambda+P_\Lambda)(I+P_V)^T \\ {=}&\Lambda + P_\Lambda + DP_V^T + P_V D + P_V\Lambda P_V^T + P_V P_\Lambda P_V^T \end{aligned} \tag{4}$$

Assuming $P_V\Lambda P_\Lambda^T$ and $P_V P_\Lambda P_V^T$ are negligible, and equating 4 to $\Lambda + \alpha\alpha^T$, we get:

$$\alpha\alpha^T = P_\Lambda + DP_V^T + P_V D \tag{5}$$

We also know that V is orthogonal, which gives $(I+P_V)(I+P_V)^T = I$. $P_V P_V^T$ is negligible, so $P_V = -P_V^T$. Combining this with the fact that P_Λ and D are diagonal, the solutions for the perturbation matrices are found from 5 as follows:

$$\begin{cases} (P_\Lambda)_{ij} = & 0 & (\text{if } i \neq j) \\ (P_\Lambda)_{ii} = & \alpha_i^2 & \\ (P_V)_{ij} = & \frac{\alpha_i \alpha_j}{\lambda_j + \alpha_j^2 - \lambda_i - \alpha_i^2} & (\text{if } i \neq j) \\ (P_V)_{ii} = & 0 & \end{cases}$$

Given these matrices, we are able to update the eigenvectors and eigenvalues of the covariance matrix.