



HAL
open science

Linear Dimensionality Reduction in Random Motion Planning

Sébastien Dalibard, Jean-Paul Laumond

► **To cite this version:**

Sébastien Dalibard, Jean-Paul Laumond. Linear Dimensionality Reduction in Random Motion Planning. The International Journal of Robotics Research, 2011, 30 (12), pp. 1461-1476. hal-00486793v2

HAL Id: hal-00486793

<https://hal.science/hal-00486793v2>

Submitted on 17 Feb 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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 random sampling in motion planning algorithms. The principle of the method is to use on line the results of a probabilistic planner 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 progression along these favored directions. That way, if the free space appears as a small volume around a submanifold of a high-dimensional configuration space, the method overcomes the usual limitations of probabilistic motion planning algorithms and finds a solution quickly. The presented method is theoretically analyzed and experimentally compared to known motion planners.

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, probabilistic algorithms exploring the configuration space (\mathcal{CS}) have been developed 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. PRMs aim at capturing the topology of the collision free space ($\mathcal{CS}_{\text{free}}$) in a learning phase in order to handle multiple planning 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.

One class of these algorithms has received special attention in the literature for its ability to rapidly solve single-query problems: tree expansion strategies, introduced in both (Hsu *et al.*, 1999) and (Kuffner & LaValle, 2000), which include RRT planners, consist in growing a tree rooted at the start configuration towards the goal configuration, by repetitively expanding it in random directions.

These methods have been proved 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 PRM (Kavraki *et al.*, 1998), as well as for RRT (Kuffner & LaValle, 2000).

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 probabilistic 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, where 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 motion planners. Actually, the velocity of an RRT algorithm, for example strongly depends on the thickness of the narrow passage - i.e. on the radius of the largest sphere included in $\mathcal{CS}_{\text{free}}$. We will analyze that effect theoretically in Section 3 then in the experimental section. The purpose of our work is to speed up the progression of probabilistic algorithms in constrained regions of $\mathcal{CS}_{\text{free}}$, it deals solely with roadmap expansions inside narrow passages, 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 an RRT algorithm to overcome some bias of classic RRT algorithms. 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, workspace hints -

such as real world obstacle directions - are used to accelerate an RRT algorithm that progresses through a narrow passage.

1.3 Motivations

Motivations for the work presented here come from motion planning for digital actors. In these high-dimensional problems - we consider whole-body motion planning, known probabilistic 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 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.

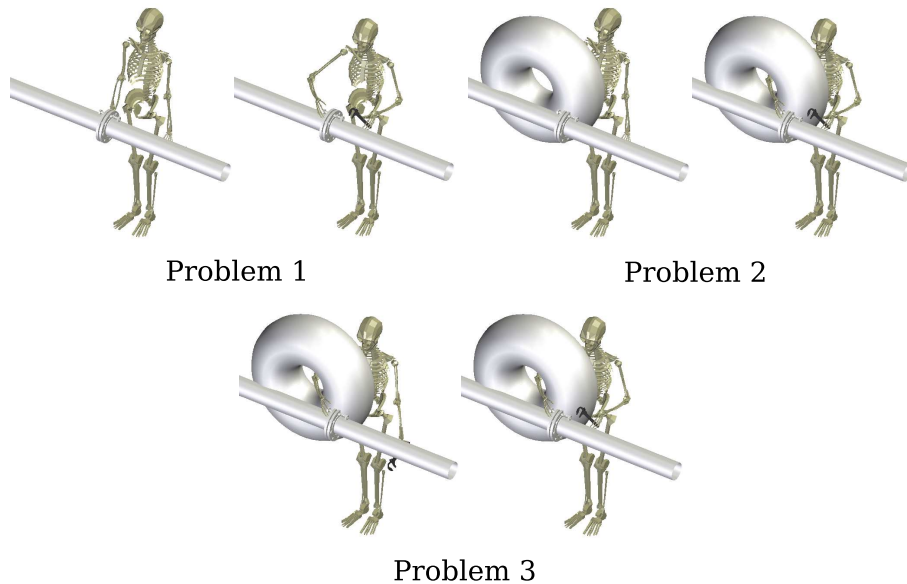


Fig. 1. Three animation motion planning problems. The mannequin has to get into position to fix the pipe. The first problem is unconstrained, we then add an obstacle for problems 2 and 3. In problem 3 the right arm is already in position and the mannequin has to find a way to move its left arm.

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 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 motion planner to sample configurations in a small volume around a sub-manifold of \mathcal{CS} - the sub-manifold corresponding to a still right hand. This slows down random expansions (we will quantify it in Section 3) and motivates our work on how to identify on line when the search 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 slows down random expansions, and a new planner based on this study. Our algorithm analyzes on line the valid configurations sampled by a probabilistic motion planner to estimate if they lie in a small volume around a submanifold of \mathcal{CS} . If so, the sampling distribution is locally adapted to fit into that volume. That way, if the path to find follows a narrow passage, the algorithm does not lose time in trying to expend the roadmap in the directions orthogonal to the passage. Thanks to the method used to describe the free space from the sampled configurations, 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.

Note that our method is a way to estimate on line if previously sampled configurations lie inside a narrow passage. It does not help in sampling a first configuration inside a narrow passage. Therefore, the adaptation of random sampling we propose is only effective if the probabilistic roadmap contains configurations inside a narrow passage. This might be the case from the start in some problems, for example industrial disassembly ones. Otherwise, some heuristics, independent from our method, can be used to find the entrance of narrow passages more efficiently: see for example Iterative Path Planning (Ferre & Laumond, 2004).

Our work uses a long known and classic 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: a 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 motion planning 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 random expansion strategy. By expansion strategies, we mean adaptive sampling strategies that repetitively try to expand the boundary of the fraction of $\mathcal{CS}_{\text{free}}$ visible from the roadmap by sampling new configurations around nodes believed to be close to this boundary (Hsu *et al.*, 2006). The second phase of the PRM algorithm as presented in (Kavraki *et al.*, 1996), for instance, is an expansion strategy.

We evaluated our method by using it on a popular algorithm that is based solely on random expansions: the RRT algorithm (LaValle, 1998), (Kuffner & LaValle, 2000). The experimental section will show examples of use of our method on different expansion based algorithms. Let us remind briefly the RRT algorithm structure.

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 $\mathcal{CS}_{\text{free}}$. At each step, 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} . One extension from a configuration of the tree to a random configuration corresponds to one step of roadmap random expansion. In the rest of the paper, we will use both terms.

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

3 Analyzing random expansions 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

Algorithm 1 RRT(q_0)

```

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

```

passages. These difficulties occur when connected areas of $\mathcal{CS}_{\text{free}}$ 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 are the local properties of $\mathcal{CS}_{\text{free}}$ that slow down random expansions. 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 $\mathcal{CS}_{\text{free}}$. To describe how a roadmap will expand from q , one should look at its visibility region $V(q)$. If $V(q)$ is equally constrained in all directions, then an RRT algorithm, for instance, will behave nicely. It 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 inside a narrow passage. The volume of $V(q)$ is small because of constraints in some directions. A random tree will grow slowly, while it could expand easily along free 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 roadmap could be expanded the furthest, while directions accounting for low volume are highly constrained and slow down motion planners.

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 expansion inside 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 this slows down random expansions, let us look at a 2D example, where one dimension is more constrained than the other.

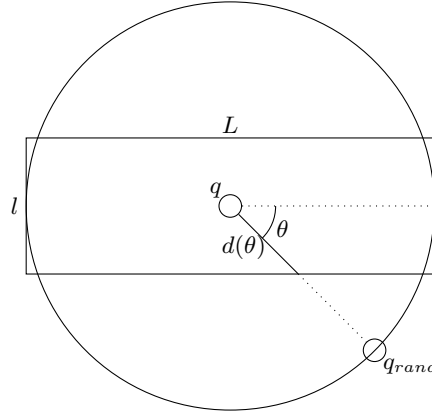


Fig. 2. One step of random expansion inside a 2D narrow passage. L is fixed and l approaches 0. We are interested in the expected length \bar{d} of the new vertex of the roadmap. For $|\theta[\pi]| \leq \arcsin(l/L)$, $d(\theta) = L/2$, otherwise $d(\theta) = l/2 \sin(\theta)$

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 q is extended is sampled at random on a circle of center q and radius half the length of the rectangle. Thus, the random extension from q is isotropic, and the longest possible progress is $L/2$.

We can compute the expected length \bar{d} of the new edge returned by one step of expansion. The exact value of $d(\theta)$ is indicated in Fig. 2.

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

As the width of the rectangle approaches 0, the expected length of the extension converges to 0, even though q sees configurations far away in one of the two dimensions. This example shows how the search is slowed down in a narrow passage: if some directions are more constrained than others, an isotropic expansion strategy will not go as fast as possible in the free directions. If \mathcal{CS} is of higher dimension n , if only one dimension is constraint, $\bar{d} \simeq_{l \rightarrow 0} k(n).l \log(l)$, where $k(n)$ is a constant depending on the dimensionality of \mathcal{CS} , and if $p > 1$ dimensions are constraint, $\bar{d} \simeq_{l \rightarrow 0} k'(n, p).l$, which means the extensions get shorter.

The important point is that what slows down random expansions 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 explicitly $\mathcal{CS}_{\text{free}}$. The only information that we have and will use are the free configurations already obtained.

Given a node we are about to extend from, we will assume its nearest neighbors in the roadmap 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 $\mathcal{CS}_{\text{free}}$. Given that description, we change the random direction of the expansion to follow the directions of the passage inside which the node lies.

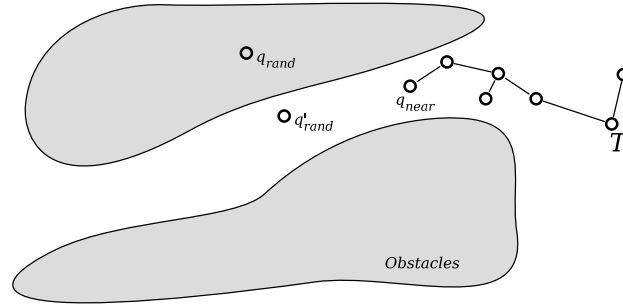


Fig. 3. Changing the random direction of extension according to $\mathcal{CS}_{\text{free}}$ local shape. Instead of extending from q_{near} to q_{rand} , we use \mathcal{CS} information and extend to q'_{rand} .

Note that while ideally we would like to describe the visibility region of q_{near} , we compute the variances and covariances of a neighborhood of q_{near} in $\mathcal{CS}_{\text{free}}$. There is no reason for those two regions of \mathcal{CS} to be the same, but we assume that the latter one is a good approximation of the first one. Using only nodes that q_{near} can actually “see” would take more iterations of classic sampling 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  $\Lambda$ , matrix  $U$ 
 $\Lambda, U \leftarrow$  EigenSolve( $C$ )
return  $\Lambda, U$ 

```

4.2 Applying PCA to Probabilistic Algorithms

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

More precisely, let us note $\lambda_1 > \dots > \lambda_n > 0$ the eigenvalues of the covariance matrix, and $\mathbf{u}_1, \dots, \mathbf{u}_n$ 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 :

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

This transformation projects q_{rand} coordinates corresponding to low variance directions towards q_{near} , while keeping the directions with high variances unchanged. Note that the probability of 0 being an eigenvalue of the covariance matrix is null, so all the directions are kept.

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

```

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

```

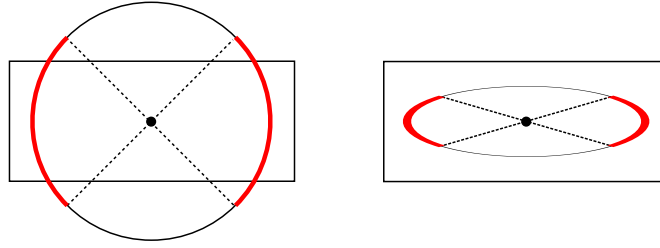


Fig. 4. One step of extension inside a narrow passage. On the left, isotropic sampling and on the right PCA adapted sampling, concentrated along the direction of the passage. The bold part shows where half of the measure is concentrated.

Recalling the example of Fig. 2, we can show the difference between classic expansion and PCA-controlled expansion. Fig.4 shows how an isotropic distribution around q is transformed into an ellipsoidal distribution. The bold part indicates where half of the probability stands, before and after changing 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 extension does not converge to 0, but to a positive limit. The respective expected length of extension are plotted in Fig. 5.

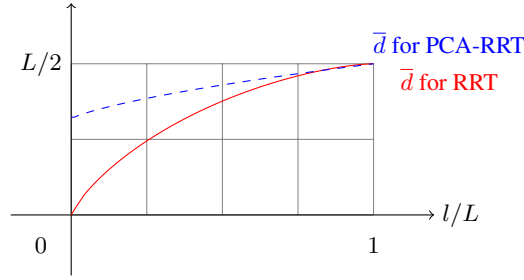


Fig. 5. Expected length of extension inside a narrow passage. The curves have to be looked at from right to left. As the space gets narrower, isotropic sampling (plain curve) converges to a null extension, while PCA controlled sampling (dashed curve) converges to a positive limit.

4.3 Accuracy of the PCA description

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

The following result partially answers these questions. It is a simplified version of mathematical results found in (Zwald & Blanchard, 2005). The accuracy of PCA description is expressed as the distance δ between the high variance subspace found by PCA and the real high variance subspace $\mathcal{CS}_{\text{free}}$ 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 $\mathcal{CS}_{\text{free}}$ lies around, and $\lambda_1 > \dots > \lambda_n > 0$ the eigen values of the local covariance matrix of $\mathcal{CS}_{\text{free}}$.

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 $1 - e^{-\xi}$:

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

This theorem is a simpler rewriting of Theorem 4 in (Zwald & Blanchard, 2005). It illustrates two points: the convergence speed 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, known probabilistic algorithms behave properly. We will discuss in the next section how not to suffer from PCA computation in a locally unconstrained space.

Remark

The theorem presented here is true under the hypothesis that the samples are drawn independently from the same probability distribution. It is not the case with configurations obtained by an RRT algorithm, for instance. Nevertheless, we found relevant to show this result to give an intuition about the convergence speed of PCA. We will also use the bound given by the theorem as a heuristic convergence criterion in the next section. Even if the theorem is not strictly applicable on samples drawn by a probabilistic planner, we found experimentally that its heuristic use was highly effective.

5 Implementation details

5.1 Normalization of the degrees of freedom

The PCA method is not scale independent: when computing the covariance matrix of a set of configurations, the values of the different degrees of freedom (DoF) are compared to each other. This can be an issue, for example when comparing translation with rotations, or bounded DoFs with unbounded ones. One solution is to compute separate PCAs for the rotations and the translations, but then we fail to express the potential correlations between them. The other solution is to normalize all the values of the DoFs, so that they all lie between 0 and 1: if a certain DoF (translation or rotation) is bounded between v_{min} and v_{max} , we divide its value by $(v_{max} - v_{min})$, and if a rotation DoF is unbounded, we divide its values by 2π . If we have unbounded translation DoFs, we have to compute a PCA on these DoFs separately, since it is irrelevant to compare their values with the other DoFs. In the following experimental section, we show results obtained with the latter solution, where all the translation DoFs were bounded. Note that we use Euler angles to describe rotations, we perform the PCA on these angles.

5.2 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 favor expansion in directions that have previously been sampled at random. This is a main drawback since the new 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 classic RRT Extend with probability 0.5. That way, the tree still covers densely the free space, while the PCA-Extend function speeds up the search in narrow passages.

5.3 Recursive PCA

The previous section presented ideas about how many points are necessary to correctly describe the space. It depends on each region of the space, narrow passages needing less points than free regions. On a first implementation of the algorithm, we chose to fix arbitrarily the number of points on which we compute a PCA. For an n -dimensional space, this number has to be at least $(n + 1)$, so that all the eigenvalues of the covariance matrix are positive. We chose to use $k.n$ points for a k between 2 and 10. This solution works, given that the user is ready to tune k for each problem.

As we do not want any tuning to be done by a user, but *automatically* by the algorithm, we have implemented a version of the algorithm where the number p of nearest neighbors used to compute the PCA is no longer a parameter. It is done by evaluating the PCA result each time we add a new neighbor: we start by computing the PCA on $(n + 1)$ points then we update that computation by adding the points one by one.

We chose to use the bound given by Theorem 1 as a heuristic 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 $\lambda = (\lambda_1, \dots, \lambda_n)$, 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 \sqrt{D} , so the relative error made on evaluating that subspace is δ/\sqrt{D} .

For a given ϵ , if $f_D(\lambda, p, r)/\sqrt{D} \leq \epsilon/1.707$ (choosing $\xi = 1$ in Eq. 1), we get, with probability at least $1 - e^{-1}$, $\delta/\sqrt{D} \leq \epsilon$. In our implementation, we chose $\epsilon = 0.1$ as a convergence criterion: if for some $D \in [1, n - 1]$, $f_D(\lambda, p, r)/\sqrt{D} \leq 0.059$, 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 $\mathcal{CS}_{\text{free}}$, while we only have access to an estimate of it. Therefore, we can not guarantee that the bound is valid. We also have to keep in mind that the hypothesis of the theorem are not strictly met, as we remarked it in the previous section. We still found that this convergence criterion was both theoretically founded and experimentally effective. 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 $\mathcal{CS}_{\text{free}}$. This data was 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 computes a rank one update of the PCA results each time we add a point.

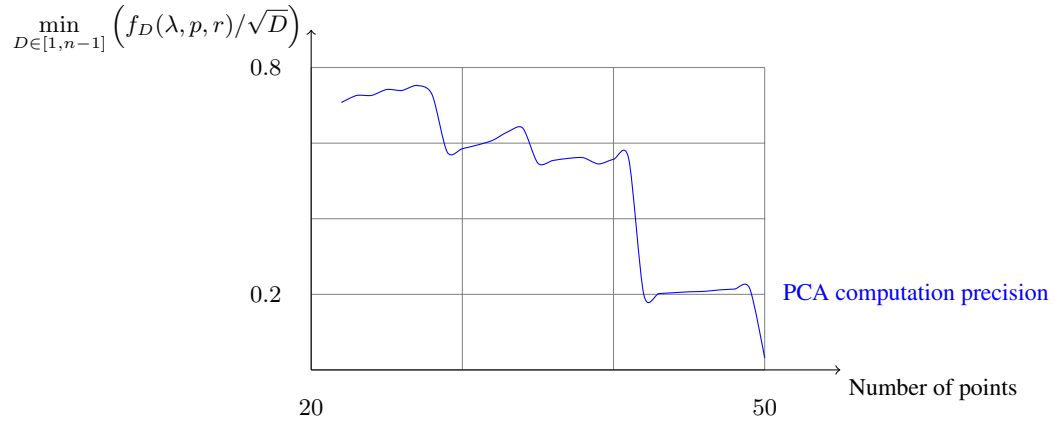


Fig. 6. Convergence of the estimate of the distance between the real high variance subspace of $\mathcal{CS}_{\text{free}}$ and the subspace identified by the PCA. The computation stops when that estimate reaches a lower threshold, which happens for 50 points. The data is taken from a trial of the PCA-RRT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

5.4 Complexity

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

The computation of the original PCA, with $(n + 1)$ points is done in $O(n^3)$ operations. Afterward, 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 adapted sampling to several classic probabilistic motion planning algorithms. First and mostly, we evaluated it by using it on an RRT algorithm.

All our algorithms have been implemented in KineoWorksTM (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 planning problem, as long as the configuration space contains narrow passages that slow down motion planners. 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 classic sampling and PCA controlled sampling.

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 sampling 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 KineoWorksTM. 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.

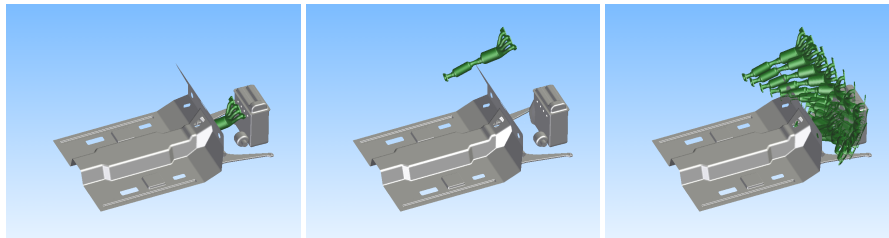


Fig. 7. Exhaust disassembly problem. Start and goal configurations and solution path.

Tab. 1 shows the average results over one hundred trials. The PCA used in average 14.5 neighbors to describe locally $\mathcal{CS}_{\text{free}}$. 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 not 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	Step Length
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, high- dimensional problems are hard to solve. The problems presented in Fig. 1 deal with a rather high-dimensional 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 follow in $\mathcal{CS}_{\text{free}}$ is non-linear.

As in the previous disassembly problem, we want to evaluate the interest of PCA for speeding up expansions inside a narrow passage, and not for finding the entrance of the passage. For all these problems, we chose to root the growing search tree at the final configuration, i.e. inside the passage. Thus, the RRT only has to follow the passage.

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 left hand in position. To do so, the motion planner has to find a path around a linear submanifold of \mathcal{CS} .

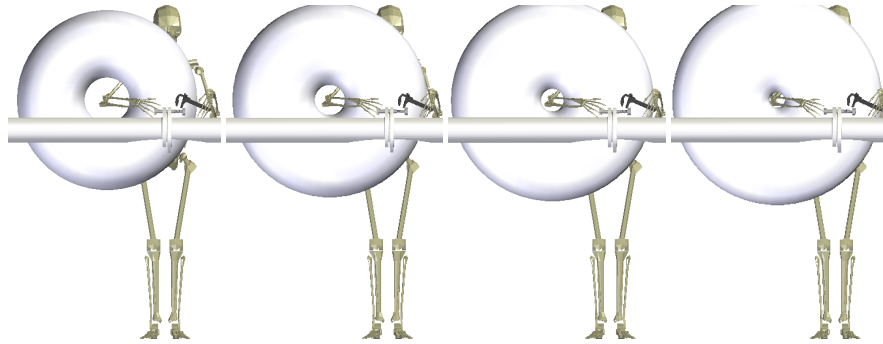


Fig. 8. Different scenarios of increasing difficulty because of the narrowness of the hole in which the right hand is.

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 the torus shaped obstacle, which means the right hand degrees of freedom hardly slow down the path finding process for the left hand. On the next experiments, 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.

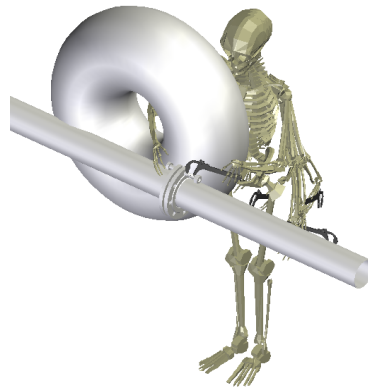


Fig. 9. Solution path for the first problem. The character only has to move its left hand.

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 neighbors used by the PCA is 31.2. It varies slightly between 28.3 for the easiest problem and 32.6 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, classic 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.

Size of the hole	Iterations	Nodes	Average Computation Time	Standard Deviation	Step Length
No torus	1,104	111.8	4.377 s	3.097 s	50.7
24 cm	2,970	166.6	14.881 s	11.531 s	9.65
16 cm	5,111	200.0	27.795 s	28.418 s	7.72
12 cm	6,588	198.6	34.679 s	42.311 s	5.11
8 cm	11,533	177.9	50.992 s	53.595 s	2.70

Table 2. Results of PCA-RRT for the first set of animation problems.

Size of the hole	Iterations	Nodes	Average Computation Time	Standard Deviation	Step Length	Failure Rate
No torus	56,446	182.2	123.302 s	455.952 s	1.23	4%
24 cm	240,691	188.7	514.678 s	850.086 s	0.335	20%
16 cm	254,300	166.0	528.041 s	861.391 s	0.255	22%
12 cm	303,732	177.7	686.534 s	1,019.66 s	0.165	23%
8 cm	305,794	164.4	709.301 s	1,010.62 s	0.0836	26%

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

One can see that PCA-RRT outperforms classic RRT for every problem, as expected on those benchmarks. An interesting point is that as the hole gets narrower, the average gain of using the PCA increases. On the easiest problem, with no torus shaped obstacle, the RRT algorithm has a surprisingly high average computation time. This is mainly due to a few costly runs where the RRT failed to find a solution. The fact that randomized motion planning algorithms have high variances in their running times has often been experimentally observed in the literature, and can be seen in our figures. Controlling the sampling by the PCA apparently “de-randomizes” the algorithm and decreases the variance of the computation time.

Fig. 10 shows the evolution of the ratio between the average length of extensions for PCA-RRT and classic 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} .

Ratio between the average extension of PCA-RRT and RRT

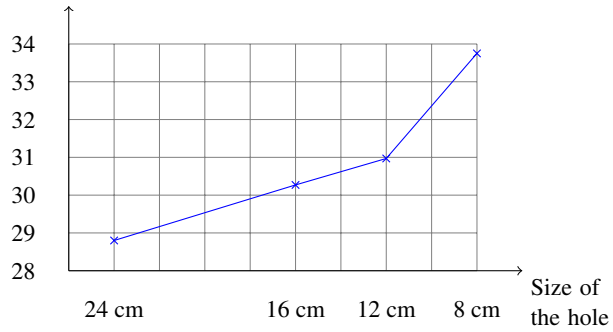


Fig. 10. Evolution of the ratio between the average distance between two nodes with PCA-RRT and classic RRT. As the hole gets smaller, the passage in \mathcal{CS} gets narrower and the gain of using PCA is higher.

Two hand motion

The environment is the same as before, but this time, the character has to get both hands into position. The path to find in \mathcal{CS} is non linear. Fig. 11 shows a solution path for this problem.

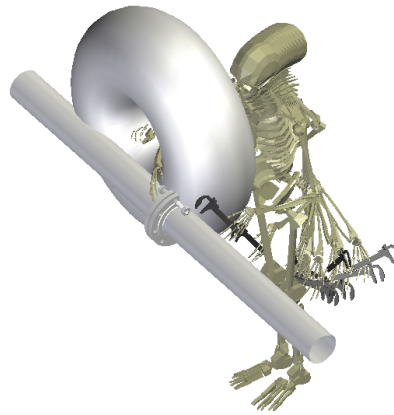


Fig. 11. Solution path for the second problem. The character has to move both hands.

We ran the classic 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

of points used by the PCA is 33.2. Its computation cost is 70% of the total computation time. The standard deviation of the computation time for the classic 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 classic RRT and PCA-RRT jumps to 300.

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 classic 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 search a lot, because of the presence of narrow passages. The PCA used in average 32.4 points to describe $C_{S_{\text{free}}}$. The computation of PCA represented 88% of the total computation time.

6.3 PCA controlled PRM

We implemented a PRM algorithm as presented in (Kavraki *et al.*, 1996). It consists in two phases, first uniform random sampling, then adaptive sampling around the nodes believed to be in narrow passages. The measure for choosing these latter nodes was implemented according to its description in (Kavraki *et al.*, 1996). We followed

Size of the hole	Iterations	Nodes	Average Computation Time	Standard Deviation	Step Length	Failure Rate
No torus	1,901	165,4	8.026 s	5.373 s	27	0%
24 cm	137,982	2,638.9	1,055.062 s	926.660 s	18	18%

Table 4. Results of PCA-RRT for the second set of animation problems.

Size of the hole	Iterations	Nodes	Average Computation Time	Standard Deviation	Step Length	Failure Rate
No torus	257,173	306.9	622.947 s	993.628 s	0.43	22%
24 cm	1,000,000	333.7	2,168.174 s	250.409 s	0.058	100%

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

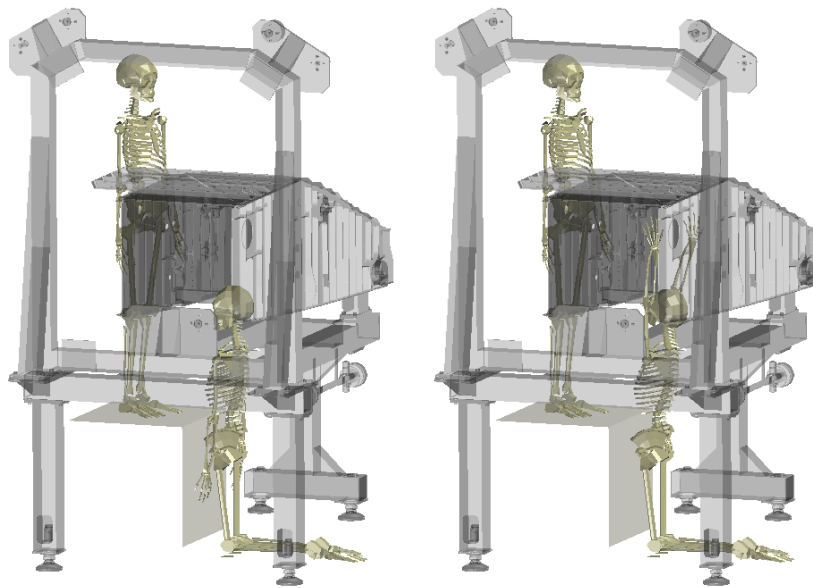


Fig. 12. Motion planning problem for a virtual actor. The main difficulty is to place both hands of the character beneath the plain hold. Picture and 3D model of the plain hold are under a Copyright of Dassault Aviation. They were given to us in the scope of the French ANR-RNTL project PerfRV2.



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

Iterations	Nodes	Average Computation Time	Standard Deviation	Step Length	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.

Iterations	Nodes	Average Computation Time	Standard Deviation	Step Length	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.

the global scheme presented in (Hsu *et al.*, 2006) for single query settings: run the first phase for 100 iterations, then the second for 200, and then repeat.

According to this description, it is straightforward to use PCA controlled sampling in the expansion phase: when a node c is chosen to be expanded, we run the PCA on its neighbors in the roadmap. If the PCA converges, we use its result to transform the direction towards which c is expanded, as presented in Section 4.2.

We show the results of PRM and PCA-PRM on the exhaust problem in Tab. 8 and 9. Both PRM and PCA-PRM always succeeded in finding a path in less than 1,000,000 iterations.

For this problem, the first uniform sampling phase of the PRM algorithm is less efficient than starting with random expansions. It is often the case in mechanical disassembly problems, as the initial configuration of the robot is already inside a

Iterations	Nodes	Average Computation Time	Standard Deviation
7,688	6,973	490.557 s	798.127 s

Table 8. Results of PCA-PRM on the exhaust problem.

Iterations	Nodes	Average Computation Time	Standard Deviation
11,957	9,423	869.541 s	1,542.159 s

Table 9. Results of PRM on the exhaust problem.

narrow passage. PCA can still improve the quality of the adaptive sampling phase, even for a small number of degrees of freedom.

6.4 PCA controlled Iterative Path Planning

Iterative Path Planning (IPP) is another tree expansion strategy presented in (Ferre & Laumond, 2004). It consists in computing a draft path by running an RRT while allowing some penetration within the obstacles. The path is then iteratively re-shaped by decreasing the allowed penetration and rerunning RRTs. It has been proved to be very efficient for mechanical disassembly problems. IPP is part of KineoWorksTM, so we used its implementation of it. We adapted the sampling for this algorithm the same way we did it for RRTs.

We show the results of IPP and PCA-IPP on the exhaust problem in Tab. 10 and 11.

Iterations	Nodes	Average Computation Time	Standard Deviation
185	40.2	3.324 s	1.381 s

Table 10. Results of PCA-IPP on the exhaust problem.

Iterations	Nodes	Average Computation Time	Standard Deviation
212	42.2	3.543 s	1.731 s

Table 11. Results of IPP on the exhaust problem.

On this problem, IPP outperforms PCA-RRT. The heuristic of widening the narrow passages is more efficient than the one of linearizing the passages. We can still gain some computation time by controlling the random sampling with PCA within the IPP algorithm.

IPP was designed to accelerate mechanical disassembly planning. In high-dimensional problems such as the animation benchmarks presented in the paper, its heuristic is less efficient. Tab 12 and 13 show the results of 100 trials of PCA-IPP and IPP on the animation problem where the robot has to its left arm with its right arm inside a 24 cm hole. PCA-IPP never failed to find a solution path, we indicated PCA-IPP failure rate in the table.

Iterations	Nodes	Average Computation Time	Standard Deviation
20,134	87.1	67.066 s	111.421 s

Table 12. Results of PCA-IPP on the animation benchmark.

Iterations	Nodes	Average Computation Time	Standard Deviation	Failure Rate
74,097	357.3	309.634 s	1,135.415 s	2%

Table 13. Results of IPP on the animation benchmark.

Here, IPP outperforms classic RRT, but both PCA-RRT and PCA-IPP are faster than IPP. PCA-IPP is slower than PCA-RRT. In fact, the benefits of using IPP depend a lot on the length of the narrow passages to follow. If a narrow passage is long, as it is the case here, many configurations sampled inside the passage for a given penetration will be rejected when the penetration threshold is lowered. This induces a lot of wasted computation time. If the narrow passage is punctual, as in the previous disassembly benchmark, the first trials of RRT will find the entrance of the passage, which means the problem will almost be solved.

In all cases, we found PCA sampling more efficient than classic sampling, but the combination of IPP heuristic and PCA sampling is not necessarily better than RRT with PCA controlled sampling.

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. This is also the case if dynamic or other non-holonomic constraints are applied to the system. For all these problems, PCA controlled sampling is irrelevant.

In order to use dimensionality reduction, 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 classic 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 than 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 classic 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 sampling 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 search to take place in an intersection of hyperplanes, 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 classic 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 search -except for the useless PCA computations- in terms of number of expansion steps.

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 another reason why we use classic RRT half of the iterations: the turn will be found with probability one and not that slower than the classic RRT would do.

8 Conclusion

An adaptation of probabilistic motion planning algorithms is proposed. The statistical method we used, PCA, is well-known and widely used in literature, but not yet to control random sampling in motion planning algorithms. Using dimensional reduction tool in roadmap expansion based algorithms, and analyzing the result of it is the contribution of this work. As expected, we observed that this technique could significantly improve probabilistic planners performances in constrained environments. Note that it does not solve the problem of finding the entrance of a narrow passage, but only accelerate the expansions inside 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. That automatic tuning relies on the use of a heuristic based on theoretical results about PCA convergence. The class of problems where we found it to be the most useful are high-dimensional 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.
- [Ferre & Laumond, 2004]Ferre, E., & Laumond, J.P. 2004. An iterative diffusion algorithm for part disassembly. *Pages 3149–3154 of: 2004 International Conference on Robotics and Automation (ICRA'2004)*.
- [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.*