Retraction-Based RRT Planner for Articulated Models

Jia Pan [1] and Liangjun Zhang [2] and Dinesh Manocha [3]
[1]*panj@cs.unc.edu,* [3] *dm@cs.unc.edu, Dept. of Computer Science, University of North Carolina at Chapel Hill*
[2]*zhanglj@stanford.edu, Dept. of Computer Science, Stanford University*
Videos available at http://gamma.cs.unc.edu/ARRRT/

***Abstract*— We present a new retraction algorithm for high DOF articulated models and use our algorithm to improve the performance of RRT planners in narrow passages. The retraction step is formulated as a constrained optimization problem and performs iterative refinement on the boundary of C-Obstacle space. We also combine the retraction algorithm with decomposition planners to handle very high DOF articulated models. The performance of our approach is analyzed using Voronoi diagrams and we show that our retraction algorithm provides a good approximation to the ideal RRT-extension in constrained environments. We have implemented our algorithm and tested its performance on robots with more than 40 DOFs in complex environments. In practice, we observe significant performance (2-80X) improvement over prior RRT planners on challenging scenarios with narrow passages.**

I. Introduction

Sample-based planning has been widely used to compute collision-free path for robots in complex environments. These methods generate samples with randomized techniques, and connect them using local planning methods. The main goal is to capture the connectivity of the free space of a robot's configuration space by using tree or graph structures such as probabilistic roadmaps (PRMs) or rapidly-exploring random trees (RRTs) [15]. However, the performance of sample-based planning algorithms may degrade if the free space has narrow passages.

In this paper, we address the problem of computing collision-free motion for articulated models in constrained environments with multiple obstacles and narrow passages. Most of the prior work on handling narrow passages has been mainly limited to rigid models, e.g. [1], [8], [28], [31]. In practice, articulated models result in some additional challenges with respect to sample-based motion planning. First of all, the articulated models have many more degrees of freedom (DOFs), which increase the complexity of the underlying planning problem. Secondly, the planner needs to ensure that there are no self-collisions in the robot, in addition to preventing collisions between the robot and the obstacles. These self-collisions can generate many small and isolated components in the $\mathcal{C}$-Obstacle space and thereby result in additional challenges in terms of handling narrow passages.

Main results: We present a new retraction-based algorithm for articulated models and combine it with RRT planners. We formulate the retraction step as a constrained optimization problem that performs iterative refinement on the boundary of $\mathcal{C}$-Obstacle space to compute a free-space configuration that is nearest to the random sample. We further generalize our retraction algorithm to planners that use decomposition techniques to deal with very high DOF models. We also analyze our retraction algorithm based on Voronoi diagrams and show that our retraction is a good approximation for ideal RRT-extension in constrained environments. We have implemented our algorithm and tested its performance on models with 40 DOFs in complex environments with narrow passages. In practice, our algorithm can significantly improve the performance by 2-80 times as compared to RRT planners.

The rest of paper is organized as follows. In Sec. II, we briefly survey the related work. We present our optimization-based retraction algorithm and combine it with RRT planners in Sec. III. Sec. IV describes a modified algorithm to combine our retraction step with decomposition planners to handle very high DOF models. We analyze the performance of our planners using Voronoi diagrams in Sec. V. We discuss its implementation and highlight the performance on different benchmarks in Sec. VI.

II. Related Work

In this section, we give a brief overview of related work in motion planning for articulated models and handling narrow passages.

A. Motion Planning for Articulated Models

There is considerable work on motion planning for articulated models using sample-based planners. Some of the most popular algorithms are based on decomposition techniques, which assume that the articulated model can be decomposed into components with low-correlation and then use suitable strategies to coordinate between different components [5], [24], [32]. However, these methods may not work well in dealing with narrow passages. Other approaches use reduced kinematics [10], [18] and multi-level methods [29].

For articulated models with very high dimension (> 100), e.g. protein chains, potential energy based approaches are usually used [13], [20], [26], [27]. The potential energy formulation is often defined to guide the search toward a goal configuration.

B. Handling Narrow Passages for Articulated Models

Many techniques have been proposed to address the narrow passage problem for articulated models. These include adaptive sampling according to workspace information [12] or sampling history [4], dilation-based approaches [3], [25] and retraction-based approaches. Dilation-based approaches

[3], [25] shrink the obstacles and thereby improve the visibility of the free space. However, these methods are mainly limited to volumetric objects and may not be able to handle self-collisions well. Voronoi regions of workspace can be used to generate samples in narrow passages [7], [19].

C. Retraction-Based Planning

The retraction-based approaches have been widely used to improve the performance of sample-based planners in narrow passages [1], [21], [23], [28], [31]. The main idea is to retract a randomly generated configuration that lies in $\mathcal{C}$-Obstacle space towards a more desirable region, e.g. towards the closest point on the boundary of $\mathcal{C}$-Obstacle or the medial axis of the free space.

However, the retraction step can be non-trivial. For example, computing the closest boundary point for an in-colliding configuration boils down to penetration depth computation, which has high complexity [30]. Other algorithms use heuristics to compute samples near the boundary of $\mathcal{C}$-Obstacle space or near approximate medial axis [1], [23], [28]. These methods are mainly limited to closed models and are prone to robustness problems. Other methods perform contact space planning, i.e. generate more samples that touch the boundaries of $\mathcal{C}$-Obstacles [21], [22]. Based on efficient penetration depth computation, Zhang et al. [31] present a retraction-based planner for rigid models. [22], [31] search within the contact space randomly for a new retraction sample, but mainly work well for rigid robots.

III. Retraction-Based RRT Planner

In this section, we present our modified RRT planner. We first introduce the optimization-based retraction algorithm and then use it to improve the performance of RRT planners in narrow passages. Given a randomly generated sample in $\mathcal{C}$-Obstacle space, our algorithm retracts to the closest point on the boundary of $\mathcal{C}$-Obstacle space. The basic idea is similar to [1], [31]. The main difference lies in how we utilize the workspace information to perform iterative optimization for high DOF articulated models. Moreover, our approach can handle narrow passages that are caused by obstacles as well as self-collisions of the robot within the single retraction framework.

A. Notation and Definition

We use following notations and symbols in the paper:

$\mathcal{C}$	configuration space or $\mathcal{C}$-Space, includes several $\mathcal{C}$-Obstacles and the free space $\mathcal{C}_{free}$.
$\mathcal{C}_{cont}$	the *contact space*, the boundaries of $\mathcal{C}$-Obstacles.
$\mathbf{q}$	configuration for a robot; $\mathbf{q} \in \mathcal{C}$.
$\mathrm{d}\,\mathbf{q}$	variation of $\mathbf{q}$ in the $\mathcal{C}$-Space; also a control term that describes how to change $\mathbf{q}$ each extension step.

B. Retraction-Step

As shown in Fig. 1(a), given an in-colliding sample $\mathbf{q}_r$, the retraction step attempts to compute the closest boundary point $\mathbf{q}_m$, which can be formally defined as

$$\mathbf{q}_m \equiv \mathbf{q}^* = \underset{\mathbf{q} \in \mathcal{C}_{cont}}{\operatorname{argmin}} \;\delta(\mathbf{q}, \mathbf{q}_r), \qquad (1)$$

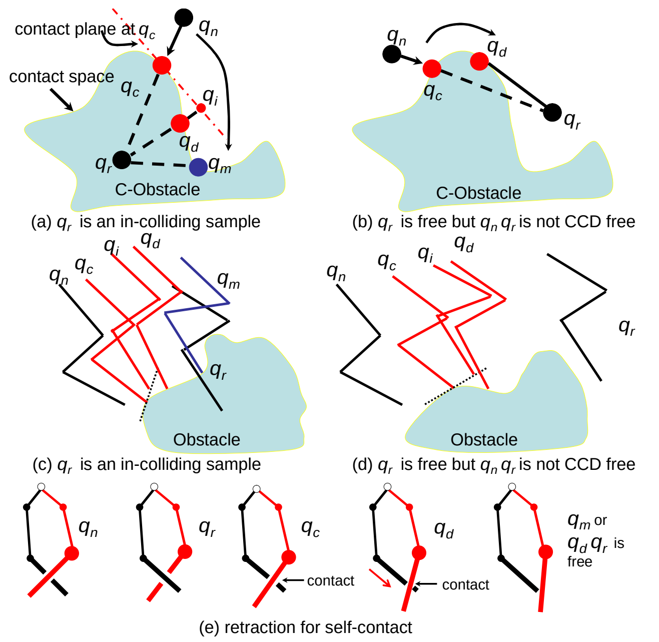

contact plane at q_c
contact space
q_n
q_c
q_i
q_d
q_r
q_m
C-Obstacle
(a) q_r is an in-colliding sample
C-Obstacle
(b) q_r is free but $q_n q_r$ is not CCD free
Obstacle
(c) q_r is an in-colliding sample
Obstacle
(d) q_r is free but $q_n q_r$ is not CCD free
contact
q_m or $q_d q_r$ is free
(e) retraction for self-contact

Fig. 1. (a) Given an in-colliding sample $\mathbf{q}_r$, retraction tries to move it to the closest point $\mathbf{q}_m \in \mathcal{C}_{cont}$ by iterative retraction-step. (b) If $\mathbf{q}_r$ is collision-free but can not connect to $\mathbf{q}_n$ by direct extension, we also perform retraction for $\mathbf{q}_r$. (c) and (d) show the retraction computation in the workspace. (e) shows the retraction process for a self-collision. $\mathbf{q}_n$ has the red (right) chain in front of the black (left) one while $\mathbf{q}_r$'s case is just the opposite. $\mathbf{q}_r$ and $\mathbf{q}_n$ can not connect to each other due to self-collisions. Retraction makes the red chain slide over black one and creates retraction samples like $\mathbf{q}_c$, $\mathbf{q}_d$. Retraction stops once $\mathbf{q}_d$ can connect to $\mathbf{q}_r$ or a local minimum $\mathbf{q}_m$ is reached.

where δ is a distance metric defined in $\mathcal{C}$-Space. The choice of a suitable metric is important but difficult. For rigid models, *DISP* metric, which does not involve any weighting factor, is used in [31]. For articulated models there is no such an equivalent metric: *DISP* metric does not have a closed form to compute for articulated models and therefore can not be used for our optimization algorithm. Moreover, a weighted Euclidean metric is prone to the values of different weights. Therefore we simply use the norm-2 metric $\delta(\mathbf{q}, \mathbf{q}_r) = \|\mathbf{q} - \mathbf{q}_r\|_2$, which is simple and works well on our benchmarks.

As shown in Fig. 1(a), in order to retract an in-colliding sample $\mathbf{q}_r$, we start with a non-colliding sample $\mathbf{q}_n$ (usually the nearest node in the RRT-tree): its projection on the contact space is the initial guess $\mathbf{q}_c$, which is pushed into a set S. Next we perform the following steps iteratively:

1) Perform contact query for $\mathbf{q}_c$, i.e. compute the closest feature pairs that cause the contact, including robot's self-collisions and collisions with the obstacles.
2) Use constrained optimization to compute a configuration $\mathbf{q}_i$ nearest to $\mathbf{q}_r$ within $\mathbf{q}_c$'s neighborhood.
3) Project $\mathbf{q}_i$ onto the contact space and compute $\mathbf{q}_d$, which is the new retraction sample. Push $\mathbf{q}_d$ into S.
4) Assign $\mathbf{q}_c = \mathbf{q}_d$ and go to step 1.

These steps are repeated until the distance to $\mathbf{q}_r$ can not be further reduced, which means a local minima has been found or the maximum number of iterations have been performed. All the retraction samples are collected in a set S, which is used by our planner in Sec. III-C.

In step 1, we perform the contact query. Unlike the rigid models, articulated models can have self-collisions, which makes the topology of $\mathcal{C}$-Obstacle space more complex. As Fig 1(e) shows, there is no obstacle between $\mathbf{q}_r$ and $\mathbf{q}_n$, but these two samples can not be connected by linear interpolation in $\mathcal{C}$-Space due to self-collisions. The retraction step locates the two components of the articulated model that are in-contact, slides them over each other and finally finds a local collision-free path between $\mathbf{q}_r$ and $\mathbf{q}_n$. More details about self-collisions are given in Sec. VI-A.

We next describe the constrained optimization in step 2. We first calculate a linear approximation of the contact space by the tangent space at $\mathbf{q}_c$ and then try to compute a $\mathbf{q}$ within the tangent space that minimizes $\delta(\mathbf{q}, \mathbf{q}_r)$.

We first formalize the tangent space constraint. Suppose the contact query in step 1 reports N contacts, each with a contact position $\mathbf{c}_i$ and a contact normal $\mathbf{n}_i$, where $1 \leq i \leq N$. $\mathbf{n}_i$ and $\mathbf{c}_i$ define a workspace tangent space at the i-th contact point. When robot's configuration $\mathbf{q}$ changes, the contact positions will also modify, but we constrain them to stay within the original tangent space:

$$\mathbf{J}_i \; \mathrm{d}\,\mathbf{q} = \mathrm{d}\,\mathbf{c}_i \tag{2}$$

$$\mathbf{n}_i^T \, \mathrm{d}\,\mathbf{c}_i = 0 \tag{3}$$

where $\mathbf{J}_i$ is the $3 \times |\mathcal{C}|$ Jacobian matrix for contact position $\mathbf{c}_i$ when robot is in current configuration $\mathbf{q}$. Therefore, the tangent space constraint for $\mathbf{c}_i$ can be represented as

$$\mathbf{K}\,\mathrm{d}\,\mathbf{q} = \mathbf{0}, \tag{4}$$

where $\mathbf{K} = [\mathbf{K}_1^T, ..., \mathbf{K}_N^T]^T$ is the *contact matrix*, and $\mathbf{K}_i = \mathbf{n}_i^T \mathbf{J}_i$. Then Eq. 1 can be approximated by

$$\mathbf{q}^* = \min_{\mathbf{K}\,\mathrm{d}\,\mathbf{q}=\mathbf{0}} \delta(\mathbf{q}, \mathbf{q}_r) \tag{5}$$

Next we formalize the objective function. Instead of finding the optimal $\mathbf{q}^*$ for Eq. 5, we try to find the optimal control $\mathrm{d}\,\mathbf{q}^*$ instead (i.e. its integration output $\int \mathrm{d}\,\mathbf{q}^* = \mathbf{q}^*$). This approach does not solve the original optimization equation (Eq. 5) but it is more convenient for our tangent constraint. The new objective function is $(\mathrm{d}\,\mathbf{q} - \mathrm{d}\,\mathbf{q}_{desire})^T(\mathrm{d}\,\mathbf{q} - \mathrm{d}\,\mathbf{q}_{desire})$, where $\mathrm{d}\,\mathbf{q}_{desire} = \alpha(\mathbf{q}_r - \mathbf{q}_c)$. This can be viewed as a simple feed-back control: where the initial guess $\mathbf{q}_c$ is far from the target, we hope that the robot moves faster, otherwise it should move slower to avoid overshooting. The actual control $\mathrm{d}\,\mathbf{q}$ should approximate the desired control $\mathrm{d}\,\mathbf{q}_{desire}$ as much as possible. As a result the local constrained optimization in step 2 can be formalized as:

$$\mathrm{d}\,\mathbf{q}^* = \operatorname*{argmin}_{\mathbf{K}\,\mathrm{d}\,\mathbf{q}=\mathbf{0}} (\mathrm{d}\,\mathbf{q} - \mathrm{d}\,\mathbf{q}_{desire})^T(\mathrm{d}\,\mathbf{q} - \mathrm{d}\,\mathbf{q}_{desire}). \tag{6}$$

According to first-order necessary condition for equality constraints [2], $\mathrm{d}\,\mathbf{q}^*$ satisfies $\boldsymbol{\lambda}^T \nabla(\mathbf{K}\,\mathrm{d}\,\mathbf{q}) = \nabla[(\mathrm{d}\,\mathbf{q} - \mathrm{d}\,\mathbf{q}_{desire})^T(\mathrm{d}\,\mathbf{q} - \mathrm{d}\,\mathbf{q}_{desire})]$. Overall, the optimal control could be expressed as:

$$\mathrm{d}\,\mathbf{q}^* = (\mathbf{I} - \mathbf{K}^\dagger\mathbf{K})\,\mathrm{d}\,\mathbf{q}_{desire} = \alpha(\mathbf{I} - \mathbf{K}^\dagger\mathbf{K})(\mathbf{q}_r - \mathbf{q}_c), \tag{7}$$

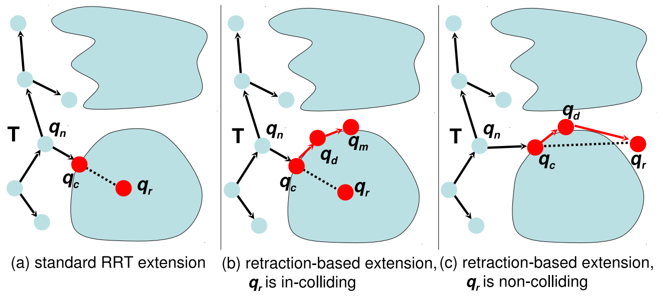

T
q_n
q_c
q_r
q_d
q_m
(a) standard RRT extension
(b) retraction-based extension, q_r is in-colliding
(c) retraction-based extension, q_r is non-colliding

Fig. 2. (a) Standard RRT-extension grows the RRT-tree T from $\mathbf{q}_n$ to $\mathbf{q}_r$, the extension is truncated when meeting the obstacles and stops at $\mathbf{q}_c \in \mathcal{C}_{cont}$. (b) If $\mathbf{q}_r$ is in-colliding, retraction-based extension creates a sequence of contact samples $S = \{\mathbf{q}_c, \mathbf{q}_d, ..., \mathbf{q}_m\}$. RRT tree can grow towards $\mathbf{q}_r$ closer. (c) If $\mathbf{q}_r$ is non-colliding, but segment $\overline{\mathbf{q}_r\mathbf{q}_n}$ is not collision-free, retraction is executed similarly.

where $\mathbf{I}$ is unit matrix and $\mathbf{K}^\dagger$ is the pseudo-inverse of $\mathbf{K}$. Then the configuration $\mathbf{q}_i$ in step 2 can be computed as

$$\mathbf{q}_i = \mathbf{q}_c + \mathrm{d}\,\mathbf{q}^* \tag{8}$$

and α can be used to control the stepsize of each retraction.

If $\mathbf{q}_r$ is in free space, but the segment $\overline{\mathbf{q}_r\mathbf{q}_n}$ is not collision-free, we can also apply the retraction strategy. As Fig. 1(b) shows, we perform retraction similarly, except that for each retraction sample $\mathbf{q}_d$, we check whether it can connect to $\mathbf{q}_r$ without collisions. Once such a configuration is found, we stop the iterative step and add $\mathbf{q}_r$ into set S.

For articulated models, we must handle joint limits carefully. Suppose the upper and lower joint limits for the articulated model are $\mathbf{q}_{upper}$ and $\mathbf{q}_{lower}$, respectively. If some joints of $\mathbf{q}_d$ are out of these limits, we update the desired control to $\mathrm{d}\,\mathbf{q}_{desire} = \alpha(\mathbf{q}_r - \mathbf{q}_c) + \beta[(\mathbf{q}_{upper} - \mathbf{q}_d)^- + (\mathbf{q}_{lower} - \mathbf{q}_d)^+]$, where $(\cdot)^+ = \max(\cdot, 0)$ and $(\cdot)^- = \min(\cdot, 0)$. Next we compute $\mathbf{q}_d$ again, which is very fast as all the matrixes can be reused and only one extra matrix-vector multiplication is needed. If the new $\mathbf{q}_d$ still violates the limits, we just truncate to remain within the joint limits.

C. Retraction-Based RRT Planner

In this section, we use the retraction algorithm to improve the performance of RRT. Our new RRT-planner is designed for articulated models and retracts many of the generated samples including ones that belong to free space.

The RRT algorithm [16] explores the free $\mathcal{C}$-Space by randomly sampling and building a RRT-tree. Multiple trees are also used in some variations of RRT, e.g. Bi-RRT [11]. The RRT algorithm starts with a tree T with a root node, then it adds more nodes into the tree iteratively by a tree extension step. As Fig. 2(a) shows, standard RRT extension selects a node $\mathbf{q}_n$ in the tree that is nearest to a random sample $\mathbf{q}_r$ and attempts to extend the tree from $\mathbf{q}_n$ towards $\mathbf{q}_r$ by connecting them with a straight line in the $\mathcal{C}$-Space. However, the obstacles result in a truncated RRT-extension: i.e. the extension will stop at $\mathbf{q}_c$, the first in-contact configuration between $\mathbf{q}_n$ and $\mathbf{q}_r$. If $\mathbf{q}_n$ is itself a configuration in $\mathcal{C}_{cont}$, then the RRT tree stays the same.

Our retraction-based extension algorithm can improve the performance of RRT algorithm by enabling RRT tree to explore free space more efficiently. As shown in Fig. 2(b)(c), when extension is interrupted by obstacles, retraction strategy helps to create samples along the obstacles and to grow the RRT tree toward $\mathbf{q}_r$. There are several benefits of this enhanced scheme: First, more samples are created in and near the narrow passages; Secondly, in free environment, RRT implicitly biases for unexplored area and retraction helps RRT to keep such property when obstacles exist. We further analyze these behaviors in Sec. V.

IV. Retraction-Based Decomposition RRT Planner

In this section we present a retraction algorithm for a decomposition RRT planner for high-DOF models. We first briefly introduce a decomposition planner. Next we combine our optimization-based retraction algorithm into this decomposition planner.

A. Decomposition Planner

Our basic decomposition planner is based on prior approaches [24] and [32]. For a robot system R with D DOFs, we decompose it into M parts: $R = \{R_1, ..., R_M\}$, with $\mathcal{C}_j$ as the configuration space for subsystem R_j with D_j DOFs and $\sum_{j=1}^{M} D_j = D$. The system configuration space $\mathcal{C}$ is the joint configuration of all M robots: $\mathcal{C} = \mathcal{C}_1 \times ... \times \mathcal{C}_M$. Each subsystem can be regarded as a single robot in a multi-robot system [24] or one part of an articulated robot [32]. The configuration of the system is $\mathbf{q} = \{\mathbf{q}^1, ..., \mathbf{q}^M\}$, where $\mathbf{q}^j \in \mathcal{C}_j$ is the configuration for subsystem R_j. $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ are the initial and goal configurations respectively.

As Algo. 1 shows, the planner computes a collision-free path using incremental steps: in the j-th step it computes a collision-free path τ_j for a merged subsystem $\tilde{R}_j = \{R_1, ..., R_j\}$. When $j = 1$, τ_1 is computed for $\tilde{R}_1 = R_1$ with standard RRT algorithm. In the following steps, τ_j is computed by a special RRT algorithm shown in Fig 3. Notice that $\tilde{R}_j = \{\tilde{R}_{j-1}, R_j\}$ and τ_{j-1} is already a collision-free path for $\tilde{R}_{j-1}$, we use a greedy bias similar to [24], [32]: $\tilde{R}_{j-1}$ will be constrained on τ_{j-1} and only R_j's configuration allows random sampling. As a result, the planning dimension is reduced from $\sum_{k=1}^{j} D_k$ to $1 + D_j$. Such special RRT can be viewed as planning R_j regarding $\tilde{R}_{j-1}$ as moving obstacles. τ_{j-1} is called the *guiding path* for $\tilde{R}_j$ [32], which is parameterized over $[0, 1]$, i.e. $\tau_{j-1} = \tau_{j-1}(s), s \in [0, 1]$.

These methods can work well in many cases. However, the guiding path bias is a greedy strategy and the overall algorithm may fail to compute a collision-free path. For example, moving obstacle $\tilde{R}_{j-1}$'s trajectory τ_{j-1} may make a very narrow passage for R_j or block it. In order to handle this problem, we can use *perturbation* [32] or *decomposition adjustment* presented in Sec. VI-A.

B. Retraction-Step

As Fig. 3 shows, the retraction for a decomposition planner is similar to that for RRT planner, except that we must consider the guiding path constraint. Suppose we are

Algorithm 1: Basic Decomposition Planner

```
Input  : decomposition profile: $R = \{R_1, R_2, ..., R_M\}$
         init and goal configuration $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$
Output: a collision-free path $\tau$ in $\mathcal{C}$
begin
    $\tau_1 \leftarrow$ RRT($\mathcal{C}_1$, $\mathbf{q}^1_{init}$, $\mathbf{q}^1_{goal}$)
    if $\tau_1 =$ NIL then
        return FAILURE
    for $j = 2$ to $M$ do
        $\tau_j \leftarrow$ RrtWithGuidingPath($[0,1] \times \mathcal{C}_j$,
        $\tau_{j-1}$, (0, $\mathbf{q}^j_{init}$), (1, $\mathbf{q}^j_{goal}$))
        if $\tau_j =$ NIL then
            return FAILURE
    return $\tau = \tau_M$
end
```

computing collision-free path τ_j for merged system $\tilde{R}_j = \{\tilde{R}_{j-1}, R_j\}$. The configuration for $\tilde{R}_j$ can be represented as $\tilde{\mathbf{q}} = (\mathbf{q}^{prev}, \mathbf{q}^{cur})$, where $\mathbf{q}^{prev}$ and $\mathbf{q}^{cur}$ are configurations of $\tilde{R}_{j-1}$ and R_j respectively. $\mathbf{q}^{prev}$ is one point on the guiding path $\tau_{j-1}(s)$ and $\mathbf{q}^{cur}$ is a random sample in $\mathcal{C}_j$. Therefore we have $\tilde{\mathbf{q}} = \begin{pmatrix}\tau_{j-1}(s)\\ \mathbf{q}^{cur}\end{pmatrix}$. We also define a (D_j+1)-dim vector $\check{\mathbf{q}} = \begin{pmatrix}s\\ \mathbf{q}^{cur}\end{pmatrix}$ which is one-to-one related with $\tilde{\mathbf{q}}$. The resulting algorithm is similar to the 4 steps in Sec. III-B, but there are some differences:

The first difference is with respect to contact query for $\tilde{\mathbf{q}}_c$. Only the contacts between R_j and obstacles or $\tilde{R}_{j-1}$ are needed to be checked, because τ_{j-1} is already collision-free and guiding path constraint requires $\tilde{R}_{j-1}$'s configuration lying on τ_{j-1} during the retraction step.

The tangent constraint is also different. As $\tau_{j-1}(s)$ is a polygonal line in $\mathcal{C}$-Space, its derivative $\frac{d}{dt}\tau_{j-1}(s)$ is a staircase function: $\mathbf{q}^{prev}$ changes with a constant ratio within each path segment. Suppose such rate for segment containing $\tau_{j-1}(s_c)$ is $\mathbf{v}$ (refer to Fig 3(b)), Eq. 2 can be updated as:

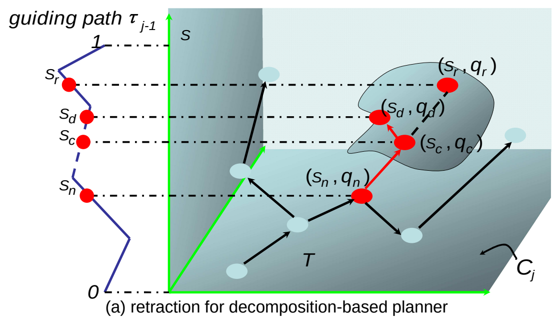
$$\begin{aligned}\mathbf{J}_i \,\mathrm{d}\,\tilde{\mathbf{q}} &= (\mathbf{J}_i^{prev}, \mathbf{J}_i^{cur}) \begin{pmatrix}\mathrm{d}\,\mathbf{q}^{prev}\\ \mathrm{d}\,\mathbf{q}^{cur}\end{pmatrix}\\ &= (\mathbf{J}_i^{prev}, \mathbf{J}_i^{cur}) \begin{pmatrix}\mathrm{d}\,s\mathbf{v}\\ \mathrm{d}\,\mathbf{q}^{cur}\end{pmatrix}\\ &= \mathrm{d}\,\mathbf{c}_i\end{aligned} \tag{9}$$

Combined with Eq. 3, the tangent constraint Eq. 4 becomes

$$\mathbf{K}\begin{pmatrix}\mathrm{d}\,s\\ \mathrm{d}\,\mathbf{q}^{cur}\end{pmatrix} = \mathbf{K}\,\mathrm{d}\,\check{\mathbf{q}} = \mathbf{0}, \tag{10}$$

where $\mathbf{K} = [\mathbf{K}_1^T, ..., \mathbf{K}_N^T]$ is the *contact matrix* in decomposed case, and $\mathbf{K}_i = (\mathbf{n}_i^T \mathbf{J}_i^{prev}\mathbf{v}, \mathbf{n}_i^T \mathbf{J}_i^{cur})$.

Other retraction steps are similar to Sec. III-B: we also define a desired control $\mathrm{d}\,\check{\mathbf{q}}_{desire} = \begin{pmatrix}\alpha_1(s_r - s_c)\\ \alpha_2(\mathbf{q}_r - \mathbf{q}_c)\end{pmatrix}$ and the optimal control will be $\mathrm{d}\,\check{\mathbf{q}}^* = (\mathbf{I} - \mathbf{K}^\dagger\mathbf{K})\,\mathrm{d}\,\check{\mathbf{q}}_{desire}$. Finally, the retraction result on the tangent space is $\check{\mathbf{q}}_i = \check{\mathbf{q}}_c + \mathrm{d}\,\check{\mathbf{q}}^*$, and we pullback it onto $\tilde{\mathcal{C}}_j$ and get $\tilde{\mathbf{q}}_i$. $\tilde{\mathbf{q}}_i$'s projection on



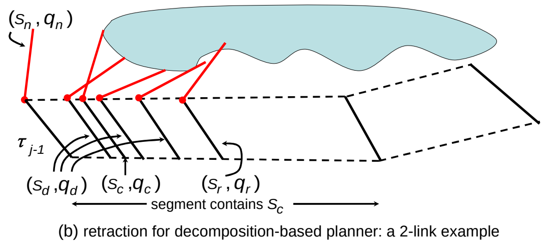


Fig. 3. (a) shows retraction-based extension for decomposition planner: The horizontal plane represents R_j's configuration space $\mathcal{C}_j$. s-axis represents the guiding path τ_{j-1}, s is the path's parameterization. The whole configuration space is $[0, 1] \times \mathcal{C}_j$, and one of its configuration is $\check{\mathbf{q}} = (s, \mathbf{q}) \equiv (\tau_{j-1}(s), \mathbf{q})$, where $\mathbf{q} \in \mathcal{C}_j$ is a projection of $(s, \mathbf{q})$ onto $\mathcal{C}_j$ and s represents a point $\tau_{j-1}(s)$ on guiding path. RRT-extension tries to connect a node on decomposed RRT tree $(s_n, \mathbf{q}_n)$ towards a random sample $(s_r, \mathbf{q}_r)$. Due to self-collision or collision with obstacles, the extension stops at $(s_c, \mathbf{q}_c)$, a contact state. Then a retraction in $[0, 1] \times \mathcal{C}_j$ creates a series of contact states $(s_d, \mathbf{q}_d)$ closer to $(s_r, \mathbf{q}_r)$. (b) shows the retraction algorithm for a 2-link example in workspace. Here the red (upper) link is sampled randomly and the black (lower) link is constrained on guiding path.

$\mathcal{C}_{cont}$ is the new extraction sample $\tilde{\mathbf{q}}_d$. We still have to check whether $\tilde{\mathbf{q}}_d$ is valid configuration: $\mathbf{q}_c$ should be within R_j's joint limits; s_d and s_c must lie in the same segment of guiding path (this can be viewed as extra joint limits requirement for an extra 'joint' s). Any violation can be resolved by the method introduced in Sec. III-B.

C. *Retraction-Based Decomposition Planner*

The retraction step improves Algo. 1 with a new RRT-extension step for RrtWithGuidingPath. The new RRT-extension is the same as the one in Sec. III-C except we replace the RRT-extension and retraction algorithms by the corresponding versions for the decomposition planner in Sec. IV-B. We also discuss other improvements for decomposition planner besides retraction in Sec. VI-A.2.

V. Analysis

In this section we analyze the behavior of the retraction-based RRT planner based on Voronoi diagram distortion.

It is well known that the behavior of RRT algorithms can be analyzed using Voronoi diagrams [15]. As shown in Fig. 4(a), given a tree built by the RRT algorithms, we consider the Voronoi diagram of all the tree nodes in the $\mathcal{C}$-Space. In the RRT-extension step, in order to find the nearest node for a random sample $\mathbf{q}_r$ is equivalent to find a tree node $\mathbf{q}$ whose associated Voronoi cell $V(\mathbf{q})$ contains $\mathbf{q}_r$. Therefore the probability of $\mathbf{q}$ to be selected for extension is proportional to the volume of $V(\mathbf{q})$. As a result, RRT planners have an implicit bias towards unexplored regions.

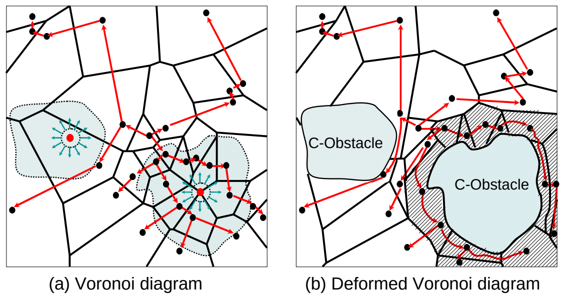


Fig. 4. (a) shows the Voronoi diagram. The black nodes are RRT-tree nodes and the red lines are edges between nodes. Imagine the voronoi diagram is deformable and mark holes in places where obstacles are to be placed (the red points). Then enlarge the hole to the shape of obstacle and deform the diagram, we obtain (b).

Such exploring property assumes no obstacles exist so that exploring $\mathcal{C}_{free}$ is equivalent to exploring $\mathcal{C}$-Space.

When the environment contains obstacles, ideally we hope that RRT will still bias towards unexplored regions in $\mathcal{C}_{free}$. However, during sampling generation and finding a tree node nearest to the sample, the RRT algorithm simply ignores the obstacles in the scene. Rather it takes the obstacles into account during the extension step. As a result, if a tree node $\mathbf{q}$ is selected for extension, there is high probability that the Voronoi cell $V(\mathbf{q})$ has a large volume, but $V(\mathbf{q}) \bigcap \mathcal{C}_{free}$ may still be small, especially in narrow passages. In other words, when there are obstacles in the scene, RRT's Voronoi bias still prefers unexplored $\mathcal{C}$-Space but not unexplored $\mathcal{C}_{free}$, though $\mathcal{C}$-Space is now larger than $\mathcal{C}_{free}$.

Our retraction strategy can improve the performance of RRT planner in an environment with obstacles. We first analyze what is the ideal RRT-extension when obstacles exist and then show that our retraction approach is a good approximation of the ideal case. We start from the RRT-tree and its associated Voronoi diagram(Fig. 4(a)). Next we try to embed $\mathcal{C}$-Obstacles into the Voronoi diagram. Imagining the diagram is deformable (i.e. edges and cells) and we mark holes in places where $\mathcal{C}$-Obstacles are going to be placed (red points in Fig. 4(a)). Then we extend the holes to the shapes of $\mathcal{C}$-Obstacles. This will distort the diagram as shown in Fig. 4(b). Since we use an elastic deformation, the distortion function f is a *topological transformation* (or *homeomorphism*) [17]. After the distortion function has been applied, the new cells of the partition and tree nodes will move away from $\mathcal{C}$-Obstacles and tree edges will become curved. However, homeomorphism ensures that the curved tree edges with stay within the free space (Fig. 4(b)) and enter/leave the same cells by the same facets. These curved edges are the ideal RRT-extension in the presence of obstacles, and the difference between basic extension is that tree nodes must be connected by curved interpolation, especially for those edges in high-distortion regions, like narrow passages. These curved cells are no longer strict Voronoi cells, in fact they are closely related to the *Landmark Voronoi Complex* [6], which is the natural extension of the geometric Voronoi diagram to the case of a graph with the shortest-path metric.

Next we show that the retraction-step approximates the ideal extension. First, the retraction step is executed when

RRT-extension is truncated and we add more samples near the contact space. As regions near $\mathcal{C}$-Obstacles always have high distortion (Fig. 4(b)), the retraction step uses truncation as a heuristic to detect a high-distortion region. Secondly, retraction tries to find a node in free space nearest to random sample by searching along contact space. Ideal extension does similar thing: for cells near $\mathcal{C}$-Obstacles (the shaded cells in Fig. 4(b)), tree nodes will move to some place near $\mathcal{C}$-Obstacles that is nearest to their original position. Therefore, retraction-based extension implicitly uses the changed Voronoi bias and can help RRT planner to perform better in narrow passages.

VI. Implementation And Results

In this section, we present results of our retraction-based planner on articulated robots. We first discuss some implementation issues. Then we highlight the performance of our new planner on a set of benchmarks with narrow passages. We show that for the basic RRT planner and the decomposition planner, the retraction algorithm can improve their efficiency obviously. All the timings reported here were taken on a laptop with 2.8GHz CPU and 2GB memory.

A. Implementation

1) Retraction Algorithm: We use PQP [14] [31] for collision detection and contact query. Based on the three types of contacts (vertex-vertex, vertex-face and edge-edge) PQP reported, we compute the contact position and contact normal for each contact, which is used in the retraction step in Sec. III-B and Sec. IV-B. Besides reporting contact with the obstacles, we also check for any self-collision in the articulated model. However, we pre-filter these contacts to remove some of the self-contacts. For example, for a human shape robot we will not report the self-contact between left hand and left arm, which are links adjacent to each other. Thus we manually decompose robot into several parts, and filter the self-contacts within the same parts. This strategy can improve algorithm's robustness.

Our approach works on triangle soup models, which may not be a smooth manifold. Thus $\mathbf{q}_i$ in Eq. 8 may be in-colliding due to surface noise. This can also be caused by concave geometry of $\mathcal{C}$-Obstacle at $\mathbf{q}_c$. This will make the projection operation (step 3, to get $\mathbf{q}_d$) difficult. Moreover, our motion planner tries to connect each retraction sample $\mathbf{q}_d$ to the RRT-tree, but the manifold noise or local convex geometry of $\mathcal{C}$-Obstacle at $\mathbf{q}_d$ will make connecting two nearby contact samples by linear interpolation, a part of RRT-extension, rather difficult. Our solution is to relax the tangent constraints in Eq. 3 a little, i.e. change it to $\mathrm{d}\,\mathbf{c}_i \cdot \mathbf{n}_i = \epsilon_i$, where ϵ_i is a small positive real number. As a result, the optimal control in Eq. 7 will be $\mathrm{d}\,\mathbf{q}^* = \mathbf{K}^\dagger \epsilon + \alpha(\mathbf{I} - \mathbf{K}^\dagger \mathbf{K})(\mathbf{q}_r - \mathbf{q}_c)$. If this relaxation still does not work, we use the *vertex enhancement* [9] method to generate additional samples around $\mathbf{q}_c$ or $\mathbf{q}_d$.

2) Decomposition Planner: Here we introduce some methods other than retraction that are used by our implementation to improve the efficiency of decomposition planner.

First we replace the Euclidean metric by a semi-geodesic metric. Euclidean distance between two samples $(s_1, \mathbf{q}_1)$ and $(s_2, \mathbf{q}_2)$ is $\|\tau(s_1) - \tau(s_2)\|_2^2 + \|\mathbf{q}_1 - \mathbf{q}_2\|_2^2$. However this metric does not bias enough for guiding path, e.g. when a guiding path τ looks like a loop in $\mathcal{C}$-Space, the two endpoints have small Euclidean distance, but the guiding path implies they should be very far away. Instead we use the semi-geodesic distance $(\int_{s_1}^{s_2} \tau'(s)ds)^2 + \|\mathbf{q}_1 - \mathbf{q}_2\|_2^2$. This metric considers the guiding path constraint and can measure distance better.

Decomposition planner is greedy, therefore it is possible that the guiding path constraint will make it hard to compute a collision-free path after adding new subsystem of robot. For example, if τ_{j-1} is not suitable, $[0,1] \times \mathcal{C}_j$ may not contain a collision-free path. Our solution is automatically re-organize the overall decomposition. We record the maximum and minimum s values of current RRT-tree's nodes, whose configuration is of the form $(s, \mathbf{q})$. If the two values do not come close toward each other for a long time, we guess guiding path τ_{j-1} is not good. Then we abandon τ_{j-1} and merge R_j and R_{j-1} into one system and execute RRT again in configuration space $[0,1] \times \mathcal{C}_{j-1} \times \mathcal{C}_j$ with τ_{j-2} as guiding path. In the worst case, the decomposition planner will degrade into a basic RRT planner.

B. Results

We test our retraction-based algorithm on a set of benchmarks. In our experiment, we run every benchmark 10 times with different random seed and compute the average running time. We use two criteria to compare the performance between planners with and without retraction: 1) planning time 2) ND-ratio, which is the ratio of number of non-degraded RRT-extensions (i.e. not the no-progress extension) to the number of all RRT-extensions. We prefer a planner with lower planning time and higher ND-ratio, which means more RRT-extensions will contribute to the free-space exploring and fewer computational capability is wasted on unsuccessful attemps. ND-ratio is a rough measurement for performance, though faster algorithms may have lower ND-ratio because they can find a path with only a few RRT-extensions.

In our experiment, there are 3 types of planners: Whole body planner is the basic RRT planner without decomposition technique, which is discussed in Sec. III. Decomposition planner is discussed in Sec. IV. Designed+Decomposition planner is a special decomposition planner: the trajectory of some of its DOFs (e.g. lower body for human robot) is predesigned and planner aims at computing the trajectory for other DOFs with the predetermined path as dynamic obstacles. This special planner is especially useful for computer animation, where animator can design the trajectory for some parts of a human-like articulated model and use planner to compute trajectory for remain parts. For each type of planner, we compare 4 types of variations: RRT, RRT with retraction (RRT-R), BiRRT, BiRRT with retraction (BiRRT-R).

In the first *bridge* benchmark (Fig. 5) a hyper-redundant robot (HRR) tries to go through a hole and brackets of bridge. We only test our retraction-based basic RRT planner here.

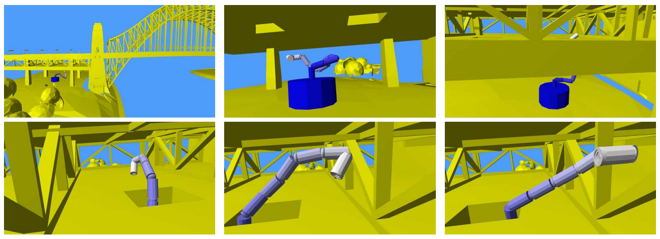


Fig. 5. Bridge: 40-DOF HRR robot tries to go through the hole and brackets of bridge. Retraction algorithm's maximum speedup is 4 times.

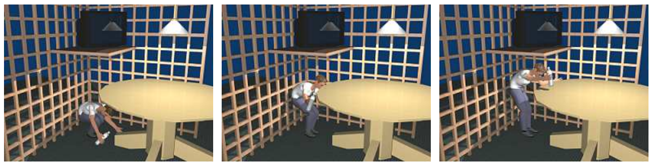


Fig. 6. Picking Object: 41-DOF human-shape robot stands up and places object on the table. Retraction algorithm's maximum speedup is 80 times.

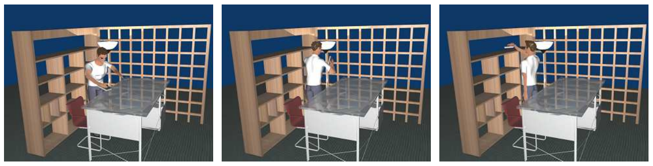


Fig. 7. Placing Object: 41-DOF human-shape robot picks the book and puts it on the shelf. Retraction algorithm's maximum speedup is 4 times and can succeed when non-retraction planner fails.

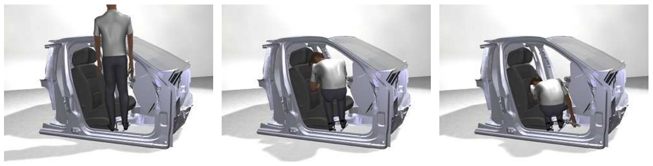


Fig. 8. Bending: 41-DOF human-shape robot bends and stretches to put the tool inside the car. Retraction algorithm's maximum speedup is 2 times and can succeed when non-retraction planner fails.

In other 3 benchmarks, a humanoid robot tries to execute some tasks in constrained environments. We compare all 3 types of planners for the first two benchmarks *Object Picking* (Fig. 6) and *Object Placing* (Fig. 7), because the constrained environments make it possible for planner to find a reasonable trajectory for lower body. For the *Bending* (Fig. 8) benchmark, complete planning can not find a good path for lower body, so we only test decomposition planner on it. The complexities of environments are shown in Table I. All the four benchmarks have narrow passages: in *bridge* hole and brackets of bridge make narrow passages; in *Object Picking* grate, table and ceiling are main narrow passages for arm and head; in *Object Placing*, grate, lamp and bookshelf cause narrow passages for arms; in *Bending*, car and the chair in car make environment challenging.

The planning results of all the benchmarks are shown in Figs. 5, 6, 7 and 8. The performance results are summarized in Tables II, III, IV and V. According to the tables, our retraction algorithm can significantly improve the performance of RRT planners in almost all cases with narrow passages. The ND-ratio of retraction-based planners are also higher, which means fewer degraded RRT-extension and makes planning more efficient.

From the result we can find that our retraction-based algorithm usually provide much larger speedup for whole body planner than for decomposition-based planner, for RRT planner than for Bi-RRT planner. It is reasonable: Both decomposition and RRT-connect algorithms are special techniques designed to improve RRT planner's efficiency. They already add some extra samples due to their own special biases to help robot go through narrow passages. Many of these extra samples may also be the retraction samples created by our retraction-based algorithms. In one word, RRT-connect or decomposition techniques have already provided part of retraction algorithm's benefit, so our method's benefit is not so obvious when combining with them. However, even for these cases, our new planner can still show an acceleration about 2-5 times. For RRT or whole body planner, the acceleration ratio is about 5-80 times.

	Bridge	Picking	Placement	Bending
#obstacles	1	4	5	7
#DOF	40	41 (27*)	41 (27*)	41
#polygons	31718	7967	52810	372609

TABLE I

GEOMETRIC COMPLEXITY OF OUR BENCHMARKS (* ARE FOR PREDESIGNED+DECOMPOSITION PLANNER).

Whole body	RRT	RRT-R	BiRRT	BiRRT-R
time (sec.)	30.515	8.031	25.109	10.140
ND ratio (%)	54.55	51.82	42.61	49.74

TABLE II

PERFORMANCE FOR BRIDGE BENCHMARK.

Whole body	RRT	RRT-R	BiRRT	BiRRT-R
time (sec.)	33.24	0.417	2.069	0.339
ND ratio (%)	38.65	56.71	30.69	67.53
Decomposition	RRT	RRT-R	BiRRT	BiRRT-R
time (sec.)	13.76	2.188	0.974	0.284
ND ratio (%)	66.17	91.55	66.48	86.27
Designed+Decomposition	RRT	RRT-R	BiRRT	BiRRT-R
time (sec.)	2.969	2.128	1.583	1.195
ND ratio (%)	80.95	93.02	87.20	91.03

TABLE III

PERFORMANCE FOR PICKING OBJECT BENCHMARK.

Whole body	RRT	RRT-R	BiRRT	BiRRT-R
time (sec.)	FAIL	62.31	43.20	24.97
ND ratio (%)	<10	31.23	22.15	44.50
Decomposition	RRT	RRT-R	BiRRT	BiRRT-R
time (sec.)	FAIL	35.53	2.514	3.126
ND ratio (%)	<10	68.58	46.12	55.55
Designed+Decomposition	RRT	RRT-R	BiRRT	BiRRT-R
time (sec.)	28.26	16.05	9.730	2.402
ND ratio (%)	71.00	81.16	68.01	78.57

TABLE IV

PERFORMANCE FOR PLACING OBJECT BENCHMARK.

Decomposition	RRT	RRT-R	BiRRT	BiRRT-R
time (sec.)	FAIL	71.19	17.03	9.765
ND ratio (%)	<10	36.47	16.41	28.77

TABLE V

PERFORMANCE FOR BENDING BENCHMARK.

VII. LIMITATIONS

Our method has a few limitations. First, our method behaves well when obstacles and links of articulated models are locally convex near the contact points. If local convex property holds, moving the links on contact points' tangent space will not create new contacts and collisions because there always exists a local separating plane between two non-colliding convex shapes. However, if locally convex property does not hold, $\mathbf{q}_i$ can have collision which is difficult to resolve.

Secondly, the retraction algorithm needs to compute the normals at the contacts, which is sensitive to the surface's quality. As a result, the performance of the retraction algorithm may be sensitive to the smoothness of the surface.

Another disadvantage is that we try to keep all the contacts on their workspace tangent planes simultaneously. However, this is not always possible, e.g. when contact matrix $\mathbf{K}$ in Eq. 4 or Eq. 10 is over-determined or the null-space of $\mathbf{K}$ is not within joint limits.

Finally, the optimization algorithm used for retraction computation may be trapped in local minima. Compared to the rigid body case [31], $\mathcal{C}$-Space of an articulated model can have much higher dimension, and so the local minima problem is more serious. Moreover, the Euclidean metric used by our optimization does not consider obstacles, which also adds to the difficulty of dealing with local minima.

VIII. Conclusions And Future Work

In this paper, we present an optimization-based retraction algorithm for articulated models to improve the performance of RRT and decomposition planners. Our algorithm retracts randomly generated $\mathcal{C}$-Obstacle samples so that they can be more likely to be connected to the RRT-tree. We apply our algorithm for articulated models with high DOFs for challenging planning scenarios. The experimental results show that our algorithm can efficiently explore narrow passages with significant speedups.

There are many avenues for future work. We are interested in analyzing our retraction-based planners using different distance metrics. We are also interested in using *Landmark Voronoi Complex* to design new PRM or RRT based planners and analyze their performances for narrow passages.

Acknowledgements. This research was supported in part by ARO Contract W911NF-04-1-0088, NSF awards 0636208, 0917040 and 0904990, DARPA/RDECOM Contract WR91CRB-08-C-0137, and Intel.

References

[1] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based prm for 3d workspaces," *Proceedings of WAFR*, pp. 197–204, 1998.

[2] A. Antoniou and W.-S. Lu, *Practical Optimization: Algorithms and Engineering Applications*. Springer, 2007.

[3] H.-L. Cheng, D. Hsu, J.-C. Latombe, and G. Sánchez-Ante, "Multi-level free-space dilation for sampling narrow passages in PRM planning," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 2006, pp. 1255–1260.

[4] S. Dalibard and J.-P. Laumond, "Control of probabilistic diffusion in motion planning," in *Proc. of Eighth Workshop on Algorithmic Foundations of Robotics*, 2008.

[5] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Proc. of IEEE Conf. on Robot. & Autom.*, pp. 1419–1424, 1986.

[6] Q. Fang, J. Gao, L. J. Guibas, V. Silva, and L. Zhang, "Glider: Gradient landmark-based distributed routing for sensor networks," in *Proc. of the 24th Conference of the IEEE Communication Society (INFOCOM)*, 2005, pp. 339–350.

[7] M. Foskey, M. Garber, M. Lin, and D. Manocha, "A voronoi-based hybrid planner," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2001.

[8] D. Hsu, J. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," in *Proc. Int. Symp. on Robotics Research*, 2005.

[9] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, pp. 12(4):566–580, 1996.

[10] J. Kuffner, "Goal-directed navigation for animated characters using real-time path planning and control," in *In Proceedings of Captech'98*. Springer-Verlag, 1998, pp. 171–186.

[11] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE International Conference on Robotics and Automation*, 2000.

[12] H. Kurniawati and D. Hsu, "Workspace-based connectivity oracle: An adaptive sampling strategy for prm planning," in *Proc. of 7th International Workshop on the Algorithmic Foundations of Robotics*, 2006.

[13] A. Ladd and L. Kavraki, "Using motion planning for knot untangling," *International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 797–808, 2004.

[14] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," Department of Computer Science, University of North Carolina, Tech. Rep. TR99-018, 1999.

[15] S. M. LaValle, *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/), 2006.

[16] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," *Robotics: The Algorithmic Perspective (Proc. of the 4th Int'l Workshop on the Algorithmic Foundations of Robotics*, 2000.

[17] J. R. Munkres, *Topology*. Pearson Education, 2000.

[18] J. Pettre, J.-P. Laumond, and T. Simeon, "3d collision avoidance for digital actors locomotion," 2003, pp. 400–405.

[19] C. Pisula, K. Hoff, M. C. Lin, and D. Manocha, "Randomized path planning for a rigid body based on hardware accelerated voronoi sampling," *Proc. of International Workshop on Algorithmic Foundation of Robotics*, 2000.

[20] B. Raveh, A. Enosh, O. Schueler-Furman, and D. Halperin, "Rapid sampling of molecular motions with prior information constraints," *PLoS Comput Biol*, vol. 5, 2009.

[21] S. Redon and M. Lin, "Practical local planning in the contact space," *Proc. of IEEE ICRA*, 2005.

[22] ——, "A fast method for local penetration depth computation," *Journal of Graphics Tools*, vol. 11, no. 2, pp. 37–50, 2006.

[23] S. Rodriguez, X. Tang, J. Lien, and N. Amato, "An obstacle-based rapidly-exploring random tree," in *Proceedings of International Conference on Robotics and Automation*, 2006, pp. 895–900.

[24] M. Saha and P. .Isto, "Multi-robot motion planning by incremental coordination," in *Proc. of IROS*, 2006, pp. 5960–5963.

[25] M. Saha, J. Latombe, Y. Chang, Lin, and F. Prinz, "Finding narrow passages with probabilistic roadmaps: the small step retraction method," *Intelligent Robots and Systems*, vol. 19, no. 3, pp. 301–319, Dec 2005.

[26] G. Song and N. M. Amato, "Using motion planning to study protein folding pathways," in *Int. Conf. Comput. Molecular Biology*, 2001, pp. 287–296.

[27] S. Thomas, G. Song, and N. M. Amato, "Protein folding by motion planning," *Physical Biology*, vol. 2, pp. 148–155, 2005.

[28] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "Motion planning for a rigid body using random networks on the medial axis of the free space," *Proc. of the 15th Annual ACM Symposium on Computational Geometry (SoCG'99)*, pp. 173–180, 1999.

[29] E. Yoshida, "Humanoid motion planning using multi-level dof exploitation based on randomized methods," 2005, pp. 3378–3383.

[30] L. Zhang, Y. Kim, and D. Manocha, "A fast and practical algorithm for generalized penetration depth computation," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[31] L. Zhang and D. Manocha, "An efficient retraction-based RRT planner," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 3743–3750.

[32] L. Zhang, J. Pan, and D. Manocha, "Motion planning of human-like robots using constrained coordination," in *IEEE-RAS International Conference on Humanoid Robots*, 2009.