Asymptotically-optimal Path Planning on Manifolds

Léonard Jaillet Josep M. Porta
Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain
Email: {ljaillet,porta}@iri.upc.edu

***Abstract*—This paper presents an approach for optimal path planning on implicitly-defined configuration spaces such as those arising, for instance, when manipulating an object with two arms or with a multifingered hand. In this kind of situations, the kinematic and contact constraints induce configuration spaces that are manifolds embedded in higher dimensional ambient spaces. Existing sampling-based approaches for path planning on manifolds focus on finding a feasible solution, but they do not optimize the quality of the path in any sense. Thus, the returned paths are usually not suitable for direct execution. Recently, RRT* and other similar asymptotically-optimal path planners have been proposed to generate high-quality paths in the case of globally parametrizable configuration spaces. In this paper, we propose to use higher dimensional continuation tools to extend RRT* to the case of implicitly-defined configuration spaces. Experiments in different problems validate the proposed approach.**

I. Introduction

The last years have witnessed a significant growth in the complexity of the robotic platforms. Nowadays, two arms service robots [21], anthropomorphic hands [28], or even humanoid robots [23] are relatively common in research labs. The availability of these platforms allows addressing increasingly relevant tasks, but they also pose significant challenges. From the point of view of path planning, one of the most critical problems is how to deal with the kinematic and contact constraints arising when manipulating an object. These constraints reduce the dimensionality of the set of valid configurations, which is in principle beneficial for path planning. However, this comes at the expense of converting the configuration space into an implicitly-defined manifold embedded in the ambient space of the variables representing the degrees of freedom of the system.

Due to the relevance of the applications in and beyond Robotics [30], several approaches have addressed the problem of path planning on manifolds. Complete planners [4] are able to find a solution path whenever one exists, but they are only adequate for low dimensional problems. A more practical approach is to adapt the successful sampling-based path planning algorithms developed in globally parametrizable spaces [16, 19]. The difficulty is that these approaches assume a parametrization of the configuration space to sample it uniformly. Since, except for some families of mechanisms [8], such parametrization does not exist, most of the existing algorithms generate samples in the configuration space from samples in the parametrizable ambient space using inverse kinematic functions [7], or iterative techniques [3, 29]. Although being probabilistically complete [3], these methods cannot guarantee a uniform distribution of samples on the configuration space, which hinders its effective exploration. To solve this issue, approaches based on higher dimensional continuation have been recently proposed [11]. In these approaches, local parametrizations of the manifold are defined and coordinated, allowing an effective exploration of the configuration space. Despite their efficiency, all these planners do not focus on the quality of the paths and their outputs are not suitable for direct execution, as shown in Fig. 1-left.

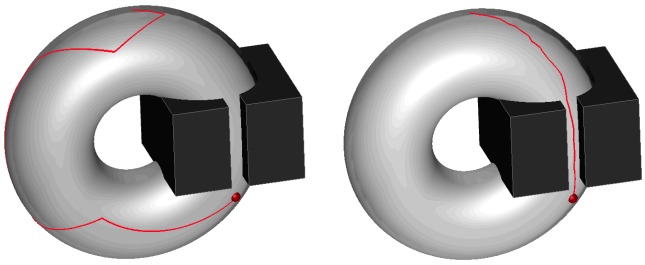


Fig. 1. Two paths connecting the same points for a small ball moving on a torus while avoiding obstacles. **Left** A low quality path obtained with AtlasRRT. **Right** A close to optimal path obtained with the AtlasRRT* algorithm proposed in this paper when optimizing the path length.

When the cost is the length of the path, the determination of optimal paths on manifolds is related to the computation of geodesic distances. This is an active field of research in Computer Graphics where the problem is addressed for triangulated meshes using variants of the fast marching method [17]. Even though some exceptions exist [22], these approaches are mainly limited to 3D surfaces and they cannot be directly applied to the configuration spaces arising in path planning. In Robotics, a recent method addresses the problem of optimal path planning on manifolds [10], but it requires an exhaustive representation of the manifold using a dense set of samples, which negatively effects the scalability of the method.

In globally parametrizable spaces, some approaches have been proposed that are less affected by the curse of dimensionality and that can deal with generic cost functions. On the one hand, locally optimal paths can be obtained using smoothing techniques [6, 26]. On the other hand, globally optimal paths can be approximated combining the construction of a RRT with stochastic optimization [12]. More recently, different sampling-based path planners have been introduced that asymptotically converge to the globally optimal path. The seminal work proposed the RRT* algorithm that achieves optimality with a moderate computational cost [14]. Latter on, many variants have been presented to speed up the convergence to the optimal path [1, 2, 15, 20].

Algorithm 1: RRT* asymptotically-optimal path planner

RRT*(COST, $\mathbf{x}_s, \mathbf{x}_g$)

input : A cost function for paths, COST, and a pair of samples to connect, $\mathbf{x}_s$ and $\mathbf{x}_g$.
output: A minimum-cost path connecting the two samples.

```
1  T ← {(∅, x_s)}
2  C(x_s) ← 0
3  for i ← 1 to N do
4      x_r ← SAMPLECONF
5      x_c ← NEARESTNODE(T, x_r)
6      x_n ← STEER(x_c, x_r)
7      if x_n ≠ ∅ then
8          γ ← γ_RRT*(log(|T|)/|T|)^(1/k)
9          X ← NEAR(T, x_n, γ)
10         x_m ← x_c
11         c_m ← C(x_c) + COST(PATH(x_c, x_n))
12         for x ∈ X do
13             c ← C(x) + COST(PATH(x, x_n))
14             if c < c_m then
15                 c_m ← c
16                 x_m ← x
17         T ← T ∪ {(x_m, x_n)}
18         C(x_n) ← c_m
19         for x ∈ X do
20             c ← C(x_n) + COST(PATH(x_n, x))
21             if c < C(x) then
22                 C(x) ← c
23                 PARENT(T, x) ← x_n
24 RETURN(PATHINTREE(T, x_g))
```

The purpose of this paper is to extend the sampling-based asymptotically-optimal path planners to the case of implicitly-defined configuration spaces. Section II describes the main challenges of adapting RRT* to path planning on manifolds. Those challenges are addressed in Section III using higher dimensional continuation tools [9]. This leads to the definition of the AtlasRRT* algorithm that is described and analyzed in Section IV and that can approximate optimal paths as shown in Fig. 1-right. Section V experimentally evaluates this algorithm and, finally, Section VI summarizes the contributions of this paper and suggests points that deserve further attention.

II. CHALLENGES OF ASYMPTOTICALLY-OPTIMAL PATH PLANNING ON MANIFOLDS

Algorithm 1 gives the pseudo-code of the RRT* path planner [13, 14]. This algorithm takes as input a cost function and a pair of configurations $\mathbf{x}_s, \mathbf{x}_g \in \mathbb{R}^k$ and attempts to connect them with a minimum-cost, collision-free path in N iterations. The algorithm initializes a tree, $\mathcal{T}$ (Line 1) and extends it by adding nodes towards points selected at random (Line 4). Each node $\mathbf{x}$ has an associated value $C(\mathbf{x})$ estimating the cost of the best path from $\mathbf{x}_s$ to $\mathbf{x}$, at a given iteration. New nodes are connected to the tree, minimizing this cost (Lines 10 to 18) and they are used to rewire the tree, eventually reducing the cost for neighboring nodes (Lines 19 to 23). The algorithm is fairly general and it can optimize paths using any cost function that is non-negative, additive, and somehow bounded by the path length. To devise a version of the RRT* planner for manifolds, the basic functions of RRT* (SAMPLECONF, NEARESTNODE, NEAR, STEER, and PATH) must be generalized.

First, SAMPLECONF (Line 4) generates a sequence of random samples in the configuration space. The distribution of the samples should be uniform since this property is assumed when fixing γ_{RRT*}, a parameter of the algorithm that determines the span of the connections at each iteration. The asymptotic optimality of the algorithm basically depends on the value of this parameter. The generation of uniformly distributed samples in an Euclidean space is straightforward, but this is not the case when the configuration space is an implicit manifold since a global parametrization of this space is generally not available.

The NEARESTNODE function (Line 5) identifies the node in $\mathcal{T}$ closer to $\mathbf{x}_r$. This should be done using the intrinsic metric of the configuration space. In a parametrizable space this metric is simple, but on a manifold, it corresponds to the more complex geodesic distance. The implementation of an efficient nearest-neighbor procedure for implicitly-defined manifolds is difficult and it has been only addressed recently in an approximated way [5]. A usual solution, that will be adopted in this paper, is to resort to the ambient space nearest-neighbor as an approximation of the manifold one, despite this may sometimes lead to inadequate tree extensions.

A related problem appears in function NEAR (Line 9) that identifies the set of nodes in $\mathcal{T}$ whose distance to $\mathbf{x}_n$ is less than γ. In this case, however, the NEAR function based on the ambient space metric returns a conservative set of neighbors. Then, each one of them can be checked for connection to $\mathbf{x}_n$ along geodesic curves to discard nodes that are actually too far away. This solution, though, relies on the ability to find geodesic paths between arbitrary points on the manifold.

Function STEER (Line 6) generates a new node moving on a collision-free path from $\mathbf{x}_c$ towards $\mathbf{x}_r$. If the motion of the robot is subject to differential constraints, $\mathbf{x}_r$ cannot be exactly reached and $\mathbf{x}_n$ is generated with the control inputs that better approximate $\mathbf{x}_r$ [15]. Without differential constraints, that is the case addressed in this paper, STEER follows the shortest path from $\mathbf{x}_c$ to $\mathbf{x}_r$. In parametrizable spaces, this shortest path corresponds to the straight line between those points. When operating on a manifold, though, a procedure to approach $\mathbf{x}_r$ from $\mathbf{x}_c$ along a geodesic path is necessary.

Finally, function PATH (Lines 11, 13 and 20) should identify the lowest-cost path between nearby configurations. However, the determination of the lowest-cost path between apparently near configurations can be arbitrarily difficult, specially when operating on manifolds. Thus, at this local path planning step, the RRT* algorithm considers the shortest path between the two configurations to connect. Since the cost is bounded by the scaled path length, any optimal path can be piecewise linearly approximated with a bounded error, that in the limit vanishes. In parametrizable spaces, the shortest path corresponds to the straight line but, on manifolds, this naturally translates to a geodesic path, which is more difficult to determine.

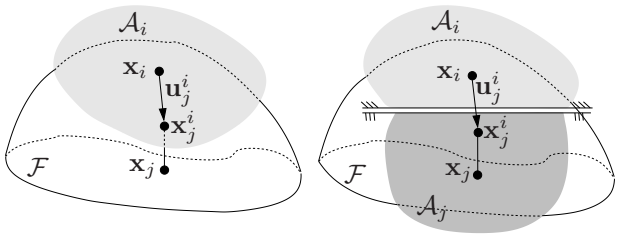


Fig. 2. **Left** A generic approximation of the exponential map is obtained by orthogonally projecting on $\mathcal{F}$ a point $\mathbf{x}_j^i$ on the tangent space at $\mathbf{x}_i$. **Right** When a new chart is defined at $\mathbf{x}_j$, the applicability areas of the two charts, $\mathcal{A}_i$ and $\mathcal{A}_j$, are coordinated to avoid overlaps.

III. Exploiting the Local Parametrization of Manifolds

As seen before, the main challenges to develop a version of RRT* on manifolds are the need to uniformly sample on the manifold and the necessity to connect configurations along geodesic curves. Both problems would be trivially solved if a global isometric parametrization of the manifold was available, but these parametrizations do not exist even for simple manifolds such as a sphere in 3D. However, from Differential Geometry, it is well known that a manifold can be described by a collection of local parametrizations called charts, which can be coordinated within an atlas [25]. Higher-dimensional continuation techniques provide principled numerical tools to compute the atlas of an implicitly defined manifold starting from a given point [9]. In this paper, we rely on such tools to define an asymptotically-optimal path planner on manifolds.

A. Local Parametrization of a Manifold

Let us consider a n-dimensional joint ambient space and a k-dimensional configuration space, $\mathcal{F}$, implicitly defined by a set of constraints

$$\mathbf{F}(\mathbf{x}) = 0,$$

with $\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}^{n-k}$, $n > k > 0$ and where we assume that the configuration space is a smooth manifold everywhere, without considering the presence of singularities.

A chart, $\mathcal{C}_i$, locally parametrizes the k-dimensional manifold around a given point, $\mathbf{x}_i$, with a bijective map, $\mathbf{x}_j = \psi_i(\mathbf{u}_j^i)$, between parameters $\mathbf{u}_j^i$ in $\mathbb{R}^k$ and n-dimensional points $\mathbf{x}_j$ on the manifold, with $\psi_i(\mathbf{0}) = \mathbf{x}_i$. The map from the parameter space to the manifold is the exponential map and the inverse is the logarithmic map. A generic approximation to the exponential map valid for any manifold can be implemented using $T_{\mathbf{x}_i}\mathcal{F}$, the k-dimensional space tangent at $\mathbf{x}_i$ (see Fig. 2-left). An orthonormal basis for this tangent space is given by the $m \times k$ matrix, $\mathbf{\Phi}_i$, satisfying

$$\begin{bmatrix} \mathbf{J}(\mathbf{x}_i) \\ \mathbf{\Phi}_i^\top \end{bmatrix} \mathbf{\Phi}_i = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix},$$

with $\mathbf{J}(\mathbf{x}_i)$ the Jacobian of $\mathbf{F}$ evaluated at $\mathbf{x}_i$, and $\mathbf{I}$, the identity matrix. Using this basis, the mapping ψ_i is computed by first computing the mapping ϕ_i from parameters in the tangent space to coordinates in the joint ambient space,

$$\mathbf{x}_j^i = \phi_i(\mathbf{u}_j^i) = \mathbf{x}_i + \mathbf{\Phi}_i\, \mathbf{u}_j^i\ , \tag{1}$$

and then, orthogonally projecting this point on the manifold to obtain $\mathbf{x}_j$. This projection can be computed by solving

$$\begin{cases} \mathbf{F}(\mathbf{x}_j) = \mathbf{0}, \\ \mathbf{\Phi}_i^\top\, (\mathbf{x}_j - \mathbf{x}_j^i) = \mathbf{0}, \end{cases}$$

using a Newton procedure [27].

The logarithmic mapping, ψ_i^{-1}, can be approximated as the projection of a point on the tangent subspace

$$\mathbf{u}_j^i = \psi_i^{-1}(\mathbf{x}_j) = \mathbf{\Phi}_i^\top(\mathbf{x}_j - \mathbf{x}_i).$$

B. Defining a RRT on a Manifold

Using the exponential and logarithmic maps, a full atlas of the manifold can be defined [9]. Thus, in principle, one could use such a full atlas to determine an optimal path between any two given configurations using a fast marching like algorithm. However, the construction of a full atlas is computationally demanding, specially in high dimensions. Therefore, we depart from our previous work [11], where the atlas construction is intertwined with the definition of a RRT.

In this approach, the RRT is initiated on a manifold by sampling on $T_{\mathbf{x}_s}\mathcal{F}$ and projecting on the manifold when necessary using the exponential map. Formally, if $\mathbf{x}_c$ is a point on $\mathcal{F}$ already included in the RRT (initially $\mathbf{x}_s$) and $\mathbf{u}_c$ are the parameters of this point in chart $\mathcal{C}_c$, then a new vector or parameters $\mathbf{u}_n$ is generated with a small displacement from $\mathbf{u}_c$ towards $\mathbf{u}_r$, a random vector of parameters on $\mathcal{C}_c$. Then, the next point to add to the RRT is obtained as $\mathbf{x}_n = \psi_c(\mathbf{u}_n)$. However, the area of the manifold properly parametrized by a given chart is limited. As the norm of $\mathbf{u}_n$ increases, the distance to the manifold and the curvature typically increase too, and the Newton process implementing ψ_c could even diverge. Thus, a new chart is added to the atlas whenever there is a large error with respect to the manifold, i.e., when

$$\|\phi_c(\mathbf{u}_n) - \mathbf{x}_n\| > \epsilon, \tag{2}$$

or when the curvature of the manifold with respect to $\mathcal{C}_c$ is large, i.e., when

$$\frac{\|\mathbf{u}_c - \mathbf{u}_n\|}{\|\mathbf{x}_c - \mathbf{x}_n\|} < \cos(\alpha), \tag{3}$$

for user-defined parameters ϵ and α. Finally, a new chart is also added when the tree expands too far away from the chart center, i.e., when

$$\|\mathbf{u}_n\| > R, \tag{4}$$

for a given R. This maximum span for a chart helps to obtain a regular covering of the manifold.

The area of the manifold parametrized by a new chart overlaps with those for charts already in the atlas. To reduce the overlaps, the area of applicability $\mathcal{A}_i$ of a given chart $\mathcal{C}_i$ is bounded by a set of lineal inequalities, as illustrated in Fig. 2-right. These inequalities are defined in the tangent

space associated with each chart and points not fulfilling them correspond to points on the manifold that are parametrized by a neighboring chart. The set of inequalities bounding $\mathcal{A}_i$ is initially empty and enlarged as new charts are created around chart $\mathcal{C}_i$. If a new chart $\mathcal{C}_j$ is created on a point $\mathbf{x}_j$ then

$$2\,\mathbf{u}^\top \mathbf{u}_j^i \leq \|\mathbf{u}_j^i\|^2\ ,$$

is added to the set of inequalities bounding $\mathcal{A}_i$, with $\mathbf{u}_j^i = \psi_i^{-1}(\mathbf{x}_j)$. This inequality bisects the vector $\mathbf{u}_j^i$, keeping the half-space including the origin. When a chart $\mathcal{C}_i$ is fully surrounded by other charts, $\mathcal{A}_i$ becomes a convex polytope.

C. Defining a RRT* on a Manifold

The local parametrization provided by the charts can be exploited to address the issues raised in Section II. First, the parametrization allows obtaining a close to uniform distribution of samples in the part of the manifold covered by the charts at a given moment. To this end, we select a chart at random, we sample a vector of parameters $\mathbf{u}$, rejecting the samples that are not in the corresponding applicability area. In this way, the probability of generating a valid sample in a given chart is proportional to the size of its applicability area. Therefore, the distribution of samples will be uniform in the union of the applicability areas for all charts. Moreover, using Eq. (3), we have that for any two points $\mathbf{x}_i$ and $\mathbf{x}_j$ parametrized by $\mathbf{u}_i$ and $\mathbf{u}_j$ in the same chart

$$\|\mathbf{x}_i - \mathbf{x}_j\| \leq \sec(\alpha)\,\|\mathbf{u}_i - \mathbf{u}_j\|. \tag{5}$$

Thus, there is a bounded distortion between points in the tangent space and the associated points on the manifold. Therefore, the volume of the patch of the manifold covered by a given chart is a scaled factor of the volume of the corresponding applicability area. With this, when sampling on the atlas, the critical value for γ_{RRT*} (Theorem 38 in [13]) is

$$\gamma_{RRT*} > \left[2\left(1+\frac{1}{k}\right)\,\frac{\mu(A_{\text{free}})}{\zeta_k}\,\sec(\alpha)\right]^{1/k}, \tag{6}$$

where k is the dimension of the manifold, $\mu(A_{\text{free}})$ is the Lebesgue measure of the applicability areas of the charts that correspond to collision free configurations, and ζ_k is the volume of the unitary k-dimensional ball.

The local parametrization provided by the charts can also be exploited to approximate geodesic paths. In particular, consider a linear interpolation, $(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m)$, between two points, $\mathbf{u}_1$ and $\mathbf{u}_m$, in the tangent space of a given chart $\mathcal{C}_c$ and the corresponding path on the manifold $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ with $\mathbf{x}_i = \psi_c(\mathbf{u}_i)$, $i \in \{1, \ldots, m\}$. Then, the length of the path can be approximated by

$$p = \sum_{i=2}^{n} \|\mathbf{x}_{i-1} - \mathbf{x}_i\|,$$

and its length in parameter space is

$$l = \sum_{i=2}^{n} \|\mathbf{u}_{i-1} - \mathbf{u}_i\| = \|\mathbf{u}_n - \mathbf{u}_1\|.$$

Algorithm 2: Sampling on an atlas.

SampleConf (A)

input : An atlas, A.
output: A random point.

```
repeat
    r ← RandomChartIndex(A)
    u_r ← RandomInBall(R_s)
until u_r ∈ A_r
Return(φ_r(u_r))
```

Note that $l \leq p$ and that, using Eq. (5), $p \leq \sec(\alpha)\ l$. Moreover, assume that p^* is the length of the geodesic path connecting the two points and l^* is its length in parameter space. Since paths are defined as straight lines in parameter space we have that $l < l^*$ and, thus

$$l \leq l^* \leq p^* \leq p \leq \sec(\alpha)\, l \leq \sec(\alpha)\, l^*.$$

With this, the relative error of a path generated from a straight line in parameter space with respect to the geodesic path is

$$\frac{p - p^*}{p^*} \leq \frac{\sec(\alpha)\, l - l}{p^*} \leq \sec(\alpha) - 1.$$

In practice, this upper bound is overly confident since as samples get denser the relative error tends to vanish. In any case, α should be always below $\pm\pi/2$ to get a bounded error. Geodesic paths can be approximated in large areas by generating new charts as the curvature grows.

As in the Euclidean case [13], the asymptotic optimality of RRT* on manifolds is given by the ability to generate samples close to the optimal path and to connect to other samples also close to this path, at a given sample density. The first property is guaranteed by the atlas-based sampling and the second is given by the bound for γ_{RRT*} in Eq. (6). Note that the optimality is achieved irrespectively of whether some connections between samples fail or are too long at the beginning of the process, or if in some areas the density of samples is higher than the average (e.g., in the overlap between charts). These aspects only have influence on the convergence rate of the approach, but not on its long term optimality.

IV. AtlasRRT* Algorithm

Using the tools described above, we define the AtlasRRT* algorithm, an adaptation of Algorithm 1 to operate on manifolds. AtlasRRT* has the same structure as RRT*, but it additionally maintains an atlas. The atlas is initialized with one chart at $\mathbf{x}_s$ and new charts are added taking into account Eqs. (2) to (4). Moreover, AtlasRRT* keeps track of the chart parametrizing each node of the tree. Besides this, the only difference between AtlasRRT* and RRT* is the implementation of functions SampleConf, Steer, and Path.

Using the atlas, SampleConf is implemented as described in Algorithm 2. A chart is selected at random with uniform distribution and then, a point, $\mathbf{u}_r$ is sampled within a ball of radius $R_s > R$. The process is repeated until $\mathbf{u}_r$ is inside

Algorithm 3: The STEER/PATH functions.

Steer/Path$(A, \mathbf{x}_n, \mathbf{x}_r)$

input : The atlas A and two points, $\mathbf{x}_n$ and $\mathbf{x}_r$.
output: In the case of PATH, a collision-free path connecting the two samples, if it exists. In the case of STEER, a point on the manifold as close as possible to $\mathbf{x}_r$.

1 $c \leftarrow$ CHARTINDEX$(\mathbf{x}_n)$
2 $\mathbf{u}_n \leftarrow \psi_c^{-1}(\mathbf{x}_n)$
3 $\mathbf{u}_r \leftarrow \psi_c^{-1}(\mathbf{x}_r)$
4 $d \leftarrow \|\mathbf{x}_n - \mathbf{x}_r\|$
5 **if** STEER **then**
6 $\quad \mathbf{u}_r \leftarrow \mathbf{u}_n + (\mathbf{u}_r - \mathbf{u}_n)(d/\|\mathbf{u}_r - \mathbf{u}_n\|)$
7 $\quad \mathbf{x}_r \leftarrow \phi_c(\mathbf{u}_r)$
8 BLOCKED $\leftarrow$ FALSE
9 $\mathcal{P} \leftarrow \emptyset$
10 **while not** BLOCKED **and** $\|\mathbf{u}_n - \mathbf{u}_r\| > \delta$ **and** $d > 0$ **do**
11 $\quad \mathbf{u}_j \leftarrow (\mathbf{u}_r - \mathbf{u}_n)\,\delta/\|\mathbf{u}_r - \mathbf{u}_n\|$
12 $\quad \mathbf{x}_j \leftarrow \psi_c(\mathbf{u}_j)$
13 $\quad$ **if** COLLISION$(\mathbf{x}_j)$ **then**
14 $\quad\quad$ BLOCKED $\leftarrow$ TRUE
15 $\quad$ **else**
16 $\quad\quad$ NEW $\leftarrow$ FALSE
17 $\quad\quad$ **if** $\|\phi_c(\mathbf{u}_j) - \mathbf{x}_j\| > \epsilon$ **or** $\|\mathbf{u}_n - \mathbf{u}_j\|/\|\mathbf{x}_n - \mathbf{x}_j\| < \cos(\alpha)$ **or** $\|\mathbf{u}_j\| > R$ **then**
18 $\quad\quad\quad c \leftarrow$ NEWCHART$(A, \mathbf{x}_n)$
19 $\quad\quad\quad$ NEW $\leftarrow$ TRUE
20 $\quad\quad$ **else**
21 $\quad\quad\quad$ **if** $\mathbf{u}_j \notin \mathcal{A}_c$ **then**
22 $\quad\quad\quad\quad c \leftarrow$ NEIGHBORCHART$(c, \mathbf{u}_j)$
23 $\quad\quad\quad\quad$ NEW $\leftarrow$ TRUE
24 $\quad\quad$ **if** NEW **then**
25 $\quad\quad\quad \mathbf{u}_j \leftarrow \psi_c^{-1}(\mathbf{x}_j)$
26 $\quad\quad\quad \mathbf{u}_r \leftarrow \psi_c^{-1}(\mathbf{x}_r)$
27 $\quad\quad\quad$ **if** STEER **then**
28 $\quad\quad\quad\quad \mathbf{u}_r \leftarrow \mathbf{u}_j + (\mathbf{u}_r - \mathbf{u}_j)(d/\|\mathbf{u}_r - \mathbf{u}_j\|)$
29 $\quad\quad\quad\quad \mathbf{x}_r \leftarrow \phi_c(\mathbf{u}_r)$
30 $\quad\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{\mathbf{x}_j\}$
31 $\quad\quad$ **if** STEER **then**
32 $\quad\quad\quad d \leftarrow d - \|\mathbf{x}_n - \mathbf{x}_j\|$
33 $\quad\quad \mathbf{x}_n \leftarrow \mathbf{x}_j$
34 $\quad\quad \mathbf{u}_n \leftarrow \mathbf{u}_j$
35 **if** PATH **then**
36 $\quad$ **if** $\|\mathbf{u}_n - \mathbf{u}_r\| > \delta$ **then**
37 $\quad\quad$ RETURN$(\emptyset)$
38 $\quad$ **else**
39 $\quad\quad$ RETURN$(\mathcal{P})$
40 **else**
41 $\quad$ RETURN(LAST$(\mathcal{P})$)

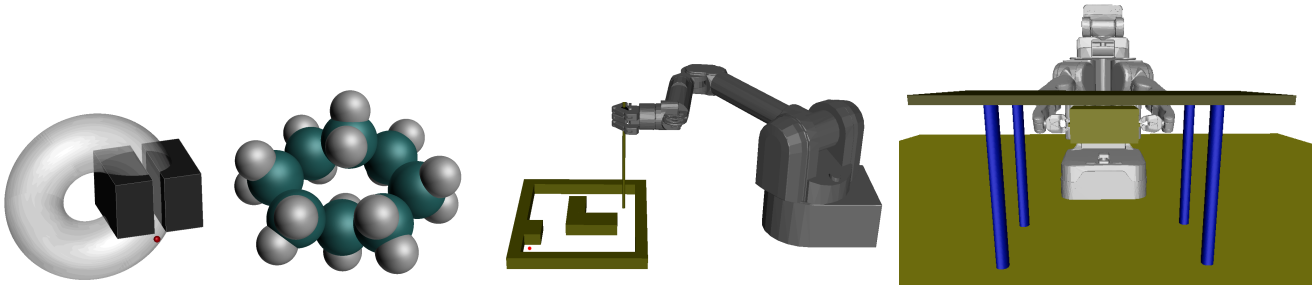
the applicability area $\mathcal{A}_r$, i.e., until it fulfills the inequalities created by neighboring charts, if any. Finally, the process returns the ambient space coordinates for $\mathbf{u}_r$ computed using ϕ_r defined in Eq. (1). To accelerate the convergence to a solution, a bias towards $\mathbf{x}_g$ is typically used. In our implementation, $\mathbf{x}_g$ has a 1% chance of being selected as random sample, as long as a path to the goal has not been found.

The main difference between the STEER and PATH functions is that STEER aims towards a point that is not on the manifold, but on the tangent space of a chart, giving a direction to expand the tree instead of a particular point to reach. Thus, Algorithm 3 presents the two functions in a compact way. Both functions get as input the atlas maintained by AtlasRRT* and two points, $\mathbf{x}_n$ and $\mathbf{x}_r$. First, they determine the chart parametrizing $\mathbf{x}_n$ (Line 1) and compute the parameters of $\mathbf{x}_n$ and $\mathbf{x}_r$ in this chart (Lines 2 and 3). In the case of STEER, we ensure that $\mathbf{u}_r$ is at least at distance d from $\mathbf{u}_n$ (Lines 6 and 7), with d the original distance between $\mathbf{x}_n$ and $\mathbf{x}_r$ (Line 4). Then, the functions proceed to move from $\mathbf{u}_n$ towards $\mathbf{u}_r$ in small steps of size δ (Lines 11 and 12). If the new configuration, $\mathbf{x}_j$, is in collision, the expansion is stopped (Line 14). Otherwise, the algorithm checks if the new point triggers the creation of a new chart (Line 17) or if it is in the applicability area of a neighboring chart (Line 21). In any of these cases, $\mathbf{u}_j$ and $\mathbf{u}_r$ are recomputed projecting $\mathbf{x}_j$ and $\mathbf{x}_r$ (Lines 25 and 26) on the new or neighbor chart determined at Lines 18 and 22, respectively. In the case of STEER, the random sample is also projected on the new chart, checking again that it is far enough from the previous point (Lines 28 and 29). Finally, the new configuration is added to the computed path, $\mathcal{P}$ (Line 30) and it is set as the point from where to continue the path (Lines 33 and 34). In the case of the STEER function, the distance already travelled is discounted (Line 32) to avoid growing an infinite branch. At the end of the PATH procedure, the computed path is returned, unless the goal configuration is not actually reached. The STEER function always returns the last element included in the path, if any.

The additional computational complexity of AtlasRRT* with respect to RRT* concentrates in the computation of the mapping ψ_c (Line 12) and in the addition of new charts to the atlas (Line 18). The first operation scales with $O(n^3)$ since it is implemented as a Newton process with a bounded number of iterations, where at each iteration a QR decomposition is used. The second operation, which is executed less often, requires to generate the new chart, that is $O(n^3)$, and to identify the neighboring charts in the atlas to avoid the overlaps. This last operation can be implemented using hierarchical structures reducing their cost to logarithmic in the number of charts in the atlas.

V. EXPERIMENTS AND RESULTS

Figure 3 shows the four benchmarks used to evaluate the AtlasRRT* algorithm. The first one is a small ball (shown in red) moving on a implicitly-defined torus with two obstacles forming a narrow corridor. This example is used since, due to its simplicity, the results are easy to visualize. The second test case is the cyclooctane, a molecule that can be modelled with eight revolute joints forming a kinematic loop. In this problem, there is a collision whenever two hydrogen atoms (shown in white in the figure) are closer than the sum of their Van der Waals radii. This example is used to illustrate the ability of AtlasRRT* to determine the optimal path among many feasible ones. The third example is the Barret arm solving a simple



(a) (b) (c) (d)

Fig. 3. The four benchmarks used in this paper. (a) A ball moving on a torus. (b) The cyclooctane molecule. (c) The Barret arm and hand solving a maze. (d) The PR2 service robot moving a box with the two arms.

TABLE I
DIMENSION OF THE CONFIGURATION AND AMBIENT SPACES, AND RELATIVE ERRORS RESPECT TO THE OPTIMAL PATH OBTAINED WITH ATLASRRT$^*_0$ AND WITH ATLASRRT* USING THE γ_{RRT*} VALUES GIVEN IN PARENTHESIS. IN BOTH CASES, COLLISIONS ARE NOT ACTIVE.

Benchmark	k	n	AtlasRRT$^*_0$	AtlasRRT*
Torus	2	3	82.54 %	0.25 % (10)
Cyclooctane	2	8	40.32 %	8.75 % (12)
Barret	3	9	98.23 %	0 % (2.5)
PR2	4	16	127.08 %	0 % (4)

maze. The figure shows the initial configuration and the goal is marked with a red circle. The task is constrained because the peg must remain in contact with and orthogonal to the maze plane without rotating about its axis. Finally, the last test case is the PR2 robot executing a coordinated manipulation task consisting in placing a box on a table moving it from underneath the table. The two last problems are used to test the scalability of the method. In all cases, the cost to optimize is the path length and the experiments are carried out with $\delta = 0.05$, $\epsilon = 0.1$, $R = 0.4$, $R_s = 2$, and α =0.45 rad. With such parameters, the error factor with respect to geodesic is below $\sec(\alpha) = 1.1$, which is reasonably small. The value of γ_{RRT*} depends on the volume of the free space that is different for each problem. In a simple problem such as the torus, it can determined by building the full atlas and evaluating Eq. (6). In the rest of the problems the value for this parameter was determined experimentally. All the experiments were executed on an Intel Core i7 at 2.93 Ghz running Mac OS X and averaged over 25 repetitions. The source code together with the described benchmarks can be downloaded from [18].

To the best of our knowledge, there are no other sampling-based asymptotically-optimal path planners for manifolds. Thus, for the purpose of comparison, we implemented what we will call C-RRT*, an alternative version of RRT* for manifolds that samples in the ambient space and uses a Jacobian pseudo-inverse strategy to connect different samples [3]. In this algorithm, the sampling is not uniform in the configuration space and the connection between samples is not guaranteed to be close to the geodesic path. Comparison are also done with respect to AtlasRRT$^*_0$, i.e. AtlasRRT* with $\gamma_{RRT*} = 0$.

Table I shows the dimension of the configuration space, k, the dimension of the ambient space, n, and the average relative error of the path obtained with AtlasRRT$^*_0$ and AtlasRRT* after 1000 iterations with respect to the optimum. For AtlasRRT*, the values of γ_{RRT*} are indicated in parenthesis. The optimal path is obtained by smoothing the best trajectory returned by AtlasRRT*. Exceptionally, in these experiments collisions are not considered and, thus, the only constraint arises from the manifold structure of the configuration spaces. The results show that the proposed algorithm reasonably converges to the optimal path in all cases, while AtlasRRT$^*_0$ generates paths that have significantly higher cost.

Figure 4 compares the cost of the paths obtained with AtlasRRT*, C-RRT*, and AtlasRRT$^*_0$ for the four benchmarks used in Fig. 3, considering collisions. In all cases, the path obtained with AtlasRRT$^*_0$ has a high cost and it is not improved once discovered. The path obtained with C-RRT* progressively improves, but slowly due to the non-uniform sampling, even in simple problems like the torus one. In the cyclooctane problem the path obtained by C-RRT* is sometimes worse than the one obtained with AtlasRRT$^*_0$. The reason is that the optimal path goes by an inner part of the manifold where C-RRT* has difficulties sampling [3]. Whereas AtlasRRT* converges to the optimal path in all cases, C-RRT* does not find a solution in all of the repetitions in the Barret and the PR2 problems. Note that in the figure, costs are plotted when at least half of the repetitions are successful and the eventual increments in the curves are caused by the different data averaged at each iteration. In the PR2 experiment, though, the original AtlasRRT* is too slow and we enhance it with two heuristics proposed in [24] and [1, 15]. The first one sets $\gamma_{RRT*} = 0$ until a first path to the goal is found. The second heuristic prevents expanding nodes that cannot be part of the optimal path, i.e., if the cost of the path to that node plus a conservative estimate of the cost to the goal is larger than the cost of the best path to the goal found so far. The same heuristics are used for C-RRT* in this benchmark.

Figure 5 shows the trees generated in a typical run of AtlasRRT* on the torus at two different stages. While in the left tree the narrow corridor is not yet discovered and, thus,

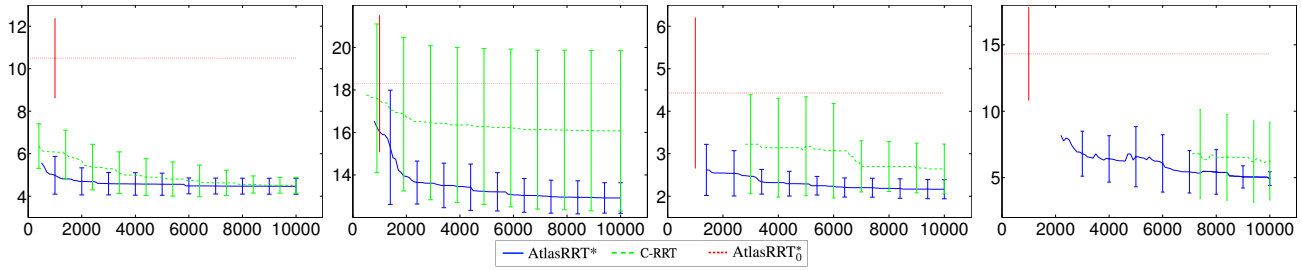


Fig. 4. Path cost versus iterations for AtlasRRT*, C-RRT*, and AtlasRRT$^*_0$ in (a) the torus, (b) the cyclooctane, (c) the Barret, and (d) the PR2 problems. Costs are given from the moment a solution is found in at least in half of the repetitions. The bars give the standard deviation.

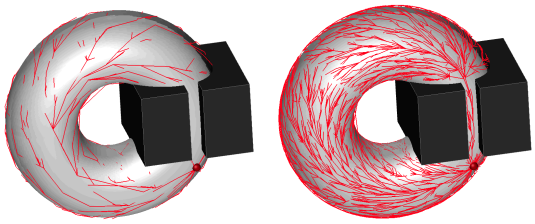


Fig. 5. Two stages of the AtlasRRT* tree construction for the torus example.

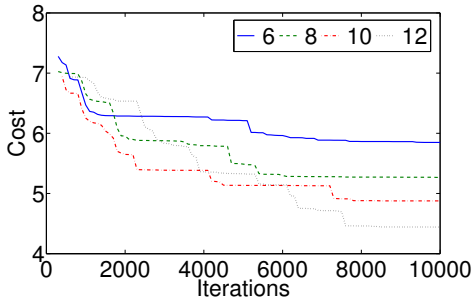


Fig. 6. Path cost versus number of iterations for AtlasRRT* on the torus and for different values of γ_{RRT*}.

the path is sub-optimal, in the right tree the optimal path is finally found. This latter tree actually provides optimal paths to all regions of the configuration space.

The selection of the right value for γ_{RRT*} is critical for the performance of the algorithm. For the case of the torus, the value for γ_{RRT*} given by Eq. (6) can be numerically computed and is about 8.1. Figure 6 shows the performance of AtlasRRT* with different values for γ_{RRT*}. Clearly, the larger the value, the lower the cost of the final path after 10000 iterations. However, with γ_{RRT*} below 8.1, the convergence seems stalled before reaching the optimum.

From the previous plot, one could think that using a large value for γ_{RRT*} is a good strategy since in this way AtlasRRT* would always converge to the optimal path. However, Fig. 7 shows that the larger γ_{RRT*}, the more it takes to complete the 10000 iterations due to the increment of nearby samples checked for connection. Thus, with a limited time, it is advantageous to use a value of γ_{RRT*} just above the threshold that guarantees convergence to the optimum.

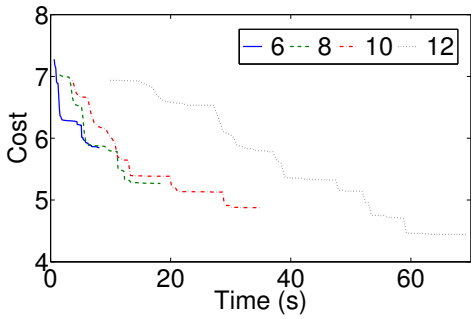


Fig. 7. Path cost versus execution time for AtlasRRT* on the torus for 10000 iterations and for different values of γ_{RRT*}.

VI. Conclusions

In this paper we have demonstrated the feasibility of generalizing asymptotically-optimal sampling-based path planners to operate on implicitly-defined configuration spaces. This is achieved by resorting to higher dimensional continuation tools. Thanks to these tools, the manifold can be properly sampled and we can determine close-to-geodesic paths for generic manifolds. Both properties are fundamental to obtain an efficient, asymptotically-optimal planner.

The introduced path planner can find optimal paths in configuration spaces with moderate dimension embedded in high dimensional ambient spaces. However, as the dimension of the configuration space increases, so does the computational complexity and we have to resort to heuristics to speed up the convergence. In the future, we would like to explore new heuristic strategies exploiting the atlas structure to improve the performance of the algorithm, possibly accepting sub-optimal solutions. The atlas structure could also be used to estimate the local density of samples. This might be used to automatically adjust γ_{RRT*} in order to obtain a good trade off between efficiency and optimality. Finally, the use of a bi-directional RRT instead of a one-directional tree could also result in a significant performance improvement.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness under project DPI2010-18449. Léonard Jaillet was supported by the CSIC under a JAE-Doc fellowship partially founded by the ESF.

References

[1] B. Akgun and M. Stilman. Sampling heuristics for optimal motion planning in high dimensions. In *EEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2640 –2645, 2011.

[2] R. Alterovitz, S. Patil, and A. Derbakova. Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning. In *IEEE International Conference on Robotics and Automation*, pages 3706–3712, 2011.

[3] D. Berenson, S. Srinivasa, and J. Kuffner. Task Space Regions: A Framework for Pose-Constrained Manipulation Planning. *International Journal of Robotics Research*, 30 (12):1435–1460, 2011.

[4] J. Canny. *The Complexity of Robot Motion Planing*. MIT Press, 1988.

[5] R. Chaudhry and Y. Ivanov. Fast approximate nearest neighbor methods for non-Euclidean manifolds with applications to human activity analysis in videos. In *European Conference on Computer Vision*, pages 735–748, 2010.

[6] P. C. Chen and Y. K. Hwang. SANDROS: a dynamic graph search algorithm for motion planning. *IEEE Transactions on Robotics and Automation*, 14(3):390–403, 1998.

[7] L. Han and N. M. Amato. A Kinematics-Based Probabilistic Roadmap Method for Closed Chain Systems. In *Algorithmic and Computational Robotics - New Directions*, pages 233–246, 2000.

[8] L. Han, L. Rudolph, J. Blumenthal, and I. Valodzin. Convexly Stratified Deformation Spaces and Efficient Path Planning for Planar Closed Chains with Revolute Joints. *International Journal of Robotics Research*, 27 (11-12):1189–1212, 2008.

[9] M. E. Henderson. Multiple Parameter Continuation: Computing Implicitly Defined k-Manifolds. *International Journal of Bifurcation and Chaos*, 12(3):451–476, 2002.

[10] T. Igarashi and M. Stilman. Homotopic Path Planning on Manifolds for Cabled Mobile Robots. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 1–18, 2010.

[11] L. Jaillet and J.M. Porta. Path Planning with Loop Closure Constraints using an Atlas-based RRT. *International Symposium on Robotics Research*, 2011.

[12] L. Jaillet, J. Cortés, and T. Siméon. Sampling-based Path Planning on Configuration-Space Costmaps. *IEEE Transactions on Robotics*, 26(4):635–646, 2010.

[13] S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[14] S. Karaman and E. Frazzoli. Incremental Sampling-based Algorithms for Optimal Motion Planning. In *Robotics: Science and Systems*, pages 267–274, 2011.

[15] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime Motion Planning using the RRT*. In *IEEE International Conference on Robotics and Automation*, pages 1478 –1483, 2011.

[16] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[17] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences*, 95(15):8431–8435, 1998.

[18] KRD Group. The CuikSuite software. http://www.iri.upc.edu/people/porta/Soft/CuikSuite2-Doc/html, 2012.

[19] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics - New Directions*, pages 293–308, 2000.

[20] J. D. Marble and K. E. Bekris. Asymptotically Near-Optimal is Good Enough for Motion Planning. In *International Symposium on Robotics Research*, 2011.

[21] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The Office Marathon: Robust Navigation in an Indoor Office Environment. In *IEEE International Conference on Robotics and Automation*, 2010.

[22] F. Mémoli and G. Sapiro. Fast computation of weighted distance functions and geodesics on implicit hypersurfaces. *Journal of Computational Physics*, 173:730–764, 2001.

[23] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schafer, B. Brunner, H. Hirschmuller, and G. Hirzinger. A Humanoid two-arm System for Dexterous Manipulation. In *IEEE-RAS International Conference on Humanoid Robots*, pages 276–283, 2006.

[24] A. Perez, S. Karaman, A. Shkolnik, E. Frazzoli, S. Teller, and M. R. Walter. Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4307–4313, 2011.

[25] A. Pressley. *Elementary Differential Geometry*. Springer Verlag, 2001.

[26] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation*, pages 489–494, 2009.

[27] W. C. Rheinboldt. MANPACK: A Set of Algorithms of Computations on Implicitly Defined Manifolds. *Computers and Mathematics with Applications*, 32(12):15–28, 1996.

[28] Schunk GmbH & Co. KG. Schunk Anthropomorphic Hand. http://www.schunk.com, 2006.

[29] M. Stilman. Global Manipulation Planning in Robot Joint Space With Task Constraints. *IEEE Transactions on Robotics*, 26(3):576–584, 2010.

[30] W. J. Wedemeyer and H. Scheraga. Exact Analytical Loop Closure in Proteins Using Polynomial Equations. *Journal of Computational Chemistry*, 20(8):819–844, 1999.