Efficient Asymptotically-optimal Path Planning on Manifolds

L. Jaillet, J. M. Porta*

Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain

Abstract

This paper presents an efficient approach for asymptotically-optimal path planning on implicitly-defined configuration spaces. Recently, several asymptotically-optimal path planners have been introduced, but they typically exhibit slow convergence rates. Moreover, these planners can not operate on the configuration spaces that appear in the presence of kinematic or contact constraints, such as when manipulating an object with two arms or with a multifingered hand. In these cases, the configuration space usually becomes an implicit manifold embedded in a higher-dimensional joint ambient space. Existing sampling-based path planners on manifolds focus on finding a feasible solution, but they do not optimize the quality of the path in any sense and, thus, the returned solution is usually not adequate for direct execution. In this paper, we adapt several techniques to accelerate the convergence of the asymptotically-optimal planners and we use higher-dimensional continuation tools to deal with the case of implicitly-defined configuration spaces. The performance of the proposed approach is evaluated through various experiments.

Keywords: Asymptotically-optimal Path Planning, Kinematic Constraints, Bi-directional Search, RRT*, LPA*, Higher-Dimensional Continuation

1. Introduction

The determination of the feasible motions between configurations that optimize a given cost function is a fundamental task in Robotics. This problem is extremely hard but, due to its relevance, it has been a subject of active research for decades [1, 2]. In order to reduce the complexity of the problem, most of the existing approaches focus on finding feasible paths within parametrizable spaces, which is still a challenging case. In this line of work, sampling-based path planners [3, 4] can efficiently find feasible paths even in high-dimensional spaces and nowadays they are the standard for industry-level solutions [5]. Unfortunately, standard sampling-based approaches do not take into account the cost of the path and, therefore, the returned paths tend to be unsuitable for direct execution, as shown in Fig. 1. To address this issue, the so-called asymptotically-optimal sampling-based path planners received a particular amount of attention over the past few years. The seminal work of Karaman and Frazzoli [6] focused more on the theoretical properties of the proposed algorithms than on their efficiency and, thus, they are slow in finding a first feasible solution, and exhibit low convergence rates in practice. Moreover, these planners, as most of the existing sampling-based approaches, rely on an explicit, global parametrization of the configuration space. While this parametrization can be trivially obtained for open chain robots, it is not

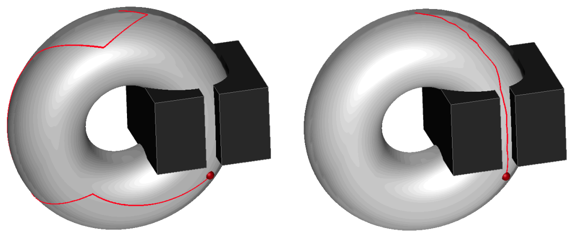
Figure 1: Two paths connecting the same query configurations for a small ball moving on a torus while avoiding obstacles. **Left** A jerky path obtained with a standard sampling-based path planner. **Right** A close-to-optimal solution when optimizing the path length.

*Correspondence to: Llorens i Artigas 4-6, 08028, Barcelona, Spain. Tel: +(34)934015751. Fax: +(34)934015750.

Email addresses: `ljaillet@iri.upc.edu` (L. Jaillet), `porta@iri.upc.edu` (J. M. Porta)

available in more complex situations such as when manipulating an object with two arms [7] or with a multi-fingered hand [8]. In these cases, the kinematic or contact constraints convert the configuration space into a non-parametric, implicitly-defined manifold embedded in the ambient space of the variables representing the degrees of freedom of the system.

This paper addresses the principal shortcomings of asymptotically-optimal sampling-based path planners introducing the AtlasBiRRT*, an efficient planner that can operate on implicitly-defined manifolds. The efficiency is achieved resorting to a bidirectional search strategy that rapidly determines feasible paths, and using tools taken from the Lifelong Planning A* (LPA*) algorithm [9] that fully exploit the cost improvements, whenever found. Moreover, to operate on manifolds, the proposed planner relies on the higher-dimensional continuation techniques recently introduced in the context of path planning [10–12].

After Section 2, which describes the related work, Section 3 introduces the AtlasBiRRT* planner at an abstract level, independent of the characteristics of the underlying configuration space. Next, Section 4 particularizes the basic operations of this algorithm to operate on implicitly-defined manifolds. Section 5 experimentally evaluates the new planner and, compares it with existing approaches. Finally, Section 6 summarizes the contributions of this paper and suggests points that deserve further attention.

2. Related Work

One of the first popular approaches to deal with the path planning problem relied on a decomposition of the configuration space in cells. Assuming that the decomposition is available, the optimal path can be obtained using standard graph-based search techniques [13] on a graph where the nodes represent the cells and the edges correspond to the possible collision-free transitions between them [14]. These approaches are resolution complete, i.e., they guarantee to find the optimal solution, if it exists up to the resolution of the cells. However, improving the path by increasing the resolution leads to an exponential increment in the computational cost, even if adaptive approaches are used [15]. Moreover, the computational cost also increases exponentially when the dimensionality of the configuration space grows.

Sampling-based path planners were proposed with the aim of dealing with high-dimensional spaces. The Probabilistic Roadmap (PRM) method [3] defines a graph that approximates the collision-free regions of the configuration space. In this graph, the nodes are randomly-sampled, collision-free configurations and the edges are simple local collision-free paths between these configurations. The graphs generated by the PRM method can be reused to solve many path planning queries. In contrast, Rapidly-exploring Random Trees (RRTs) [4] efficiently solve a particular query, using the initial configuration as a root node to grow a tree exploring the collision-free regions, until the goal configuration is reached. Both PRMs and RRTs are probabilistically complete meaning that when the number of samples approaches infinity, they guarantee to find a solution, if it exists. While the RRT-based approaches provide only one path connecting the query configurations, the PRM-based techniques typically lead to several paths and the one with the lowest cost can be retrieved using graph-based search techniques [16]. However, this process does not necessarily converge to an optimal path as the sample density is increased [6].

Solution paths generated by sampling-based methods can be post-processed to improve their quality. The random shortcut method [17] is simple, but the result is only locally optimal and only particular optimization criteria such as the path length can be taken into account. More general criteria can be used in smoothing techniques [18–20], although the optimization is still local. Globally-optimal paths can be approximated in particular problems using variational methods [21] or in general problems combining the construction of an RRT with stochastic optimization [22]. However, in the latter case, the difference between the obtained solution and the optimal path can be large in some cases, and it is not reduced over time. More recently, different sampling-based path planners that progressively converge to the globally-optimal solution have been introduced [6]. Despite their theoretical guaranties, these planners tend to be slow and, consequently, several heuristics have been proposed to improve their convergence rate. These heuristics rely, for instance, on bidirectional trees [23], on rejecting samples that can not be part of the optimal path [23], on delaying the optimization until a feasible path is found [24], on propagating the cost improvements over the graph [25], or on reducing the size of the tree/graph [26], even at the cost of somehow sacrificing the optimality [27].

All the mentioned sampling-based approaches rely on a global parametrization of the configuration space, which, except for particular families of mechanisms [28], is not available when the problem includes kinematic constraints. In this case, the configuration space is a non-parametrizable manifold implicitly-defined by the constraints. Due to the relevance of the applications in and beyond Robotics [29–35], several approaches have addressed the problem of path planning on manifolds. When minimizing the length of the path, this problem 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 [36]. Even though some exceptions exist [37], these approaches are limited to surfaces in 3D and they cannot be directly applied to the configuration spaces arising in path planning. In Robotics, methods based on simplicial decompositions [38] or on dense sets of samples [39] have been proposed. However, these methods do not gently scale to high-dimensional problems. A more promising approach is to adapt the successful sampling-based path planning algorithms. In this line of work, most of the existing algorithms generate samples on the manifold configuration space from samples in the parametrizable joint ambient space using inverse kinematic functions [40], or iterative techniques [41–43]. Although being probabilistically complete [42], these methods cannot guarantee a regular distribution of samples on the manifold, which may hinder its efficient exploration. This issue is addressed by recent approaches [10–12] based on higher-dimensional continuation [44]. However, none of these sampling-based planners for manifolds considers a cost function for paths and, thus, the returned paths are not optimal in any particular sense.

To obtain an asymptotically-optimal planner on manifolds, in a previous work we proposed the AtlasRRT* algorithm [45] that combines the use of higher-dimensional continuation tools to operate on manifolds [44] with the RRT* planner by Karaman and Frazzoli [6]. However the AtlasRRT* planner shares some of the limitations of RRT*: it may be slow in finding a first feasible path and in converging to the optimal one. Herein, we present the AtlasBiRRT* planner that addresses these issues by adapting several ingredients from the Literature. First, the new planner uses a bidirectional search to rapidly determine a first feasible solution. Moreover, to improve the convergence rate, it incorporates elements from the LPA* algorithm [9], which efficiently propagates the cost improvements to the portion of the search space that may include the optimal path. To this end, the new planner has to keep all the neighboring relations for each sample. This generates a storage overhead with respect to the AtlasRRT* planner which only maintains a tree, but this overhead is balanced by a strong gain in efficiency.

Algorithm 1: AtlasBiRRT* path planner

```
input : The initial percolation threshold, $\gamma^*$ and a pair of
        samples to connect, $\mathbf{x}_s$ and $\mathbf{x}_g$.
output: A minimum-cost path $\mathcal{P}$, connecting the two
        samples, if any.
1  $A \leftarrow$ InitAtlas($\mathbf{x}_s, \mathbf{x}_g$)
2  $\mathcal{V} \leftarrow \{\mathbf{x}_s, \mathbf{x}_g\}$
3  $\mathcal{E} \leftarrow \emptyset$
4  $P(\mathbf{x}_s) \leftarrow$ None
5  $P(\mathbf{x}_g) \leftarrow$ None
6  $C(\mathbf{x}_s) \leftarrow 0$
7  $C(\mathbf{x}_g) \leftarrow 0$
8  $\mathcal{P} \leftarrow \emptyset$
9  $l \leftarrow \infty$
10 for $i \leftarrow 1$ to $N$ do
11     $\mathbf{x}_r \leftarrow$ SampleConf($A$)
12     $\mathbf{x}_c \leftarrow$ NearestNode($\mathcal{V}, \mathbf{x}_r$)
13     $\mathbf{x}_n \leftarrow$ Steer($A, \mathbf{x}_c, \mathbf{x}_r$)
14     if $\mathbf{x}_n \neq \mathbf{x}_c$ then
15         $\gamma \leftarrow \gamma^*(log|\mathcal{V}|/|\mathcal{V}|)^{1/k}$
16         $\mathcal{X} \leftarrow$ Near($\mathcal{V}, \mathbf{x}_n, \gamma$)
17         $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathbf{x}_n\}$
18         $\mathbf{x}_m \leftarrow \mathbf{x}_c$
19         $c_m \leftarrow C(\mathbf{x}_c) +$ Cost(Path($A, \mathbf{x}_c, \mathbf{x}_n$))
20         for $\mathbf{x} \in \mathcal{X}$ do
21             $c_e \leftarrow$ Cost(Path($A, \mathbf{x}, \mathbf{x}_n$))
22             if $c_e \neq \infty$ then
23                 $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\mathbf{x}, \mathbf{x}_n, c_e), (\mathbf{x}_n, \mathbf{x}, c_e)\}$
24                 if $C(\mathbf{x}) + c_e < c_m$ then
25                     $\mathbf{x}_m \leftarrow \mathbf{x}$
26                     $c_m \leftarrow C(\mathbf{x}) + c_e$
27         $P(\mathbf{x}_n) \leftarrow \mathbf{x}_m$
28         $C(\mathbf{x}_n) \leftarrow c_m$
29         $(P, C, \mathcal{P}, l) \leftarrow$ Rewire($\mathcal{V}, \mathcal{E}, P, C, \mathcal{P}, l, \mathbf{x}_s, \mathbf{x}_g, \mathbf{x}_n$)
30 Return $\mathcal{P}$
```

3. The AtlasBiRRT* Algorithm

Algorithm 1 gives the pseudo-code of the AtlasBiRRT* planner. The algorithm takes as input the initial percolation threshold, γ^*, used to determine the local neighborhood of the new samples and a pair of configurations, $\mathbf{x}_s, \mathbf{x}_g \in \mathbb{R}^n$, and attempts to connect these configurations with a minimum-cost, collision-free path in N iterations building two trees, respectively rooted at $\mathbf{x}_s$ and $\mathbf{x}_g$. The algorithm operates under the same assumptions as previous asymptotically-optimal path planning approaches [6]. In particular, the cost function evaluating the paths will be assumed to be monotonic, non-negative, additive, and somehow bounded by the path length. To simplify the presentation, we will further assume that it is symmetric, which is typically the case when considering only kinematic constraints. This last assumption, though, can be easily dropped, if necessary.

The standard bidirectional RRT strategy [46] is designed to rapidly establish a connection between the two maintained RRTs, i.e., a feasible path between the two query configurations. As soon as this connection is established, the search is stopped. In asymptotically-optimal path planning, though, the trees need to be continuously extended, with the aim of progressively improving the solution path. Thus, the direct use of the standard bidirectional strategy would generate two overlapping trees loosing the benefit of such approach. To avoid this issue, herein we propose an alternative bidirectional search strategy. The two query nodes, $\mathbf{x}_s$ and $\mathbf{x}_g$, implicitly induce a Voronoi decomposition of the configuration space where this space is virtually divided into two regions, depending on which of the two given configurations is closer. Therefore, an RRT* can be defined in each one of these regions, which avoid any branch crossing. For instance, Fig. 2 shows the two trees generated applying this strategy on a torus-like configuration space with obstacles in $N = 10000$ iterations. In this set up, feasible paths are identified when the two trees meet and the optimal path is the one that crosses the two trees with the lowest accumulated cost. When a new sample is generated, the usual RRT* strategy is used, checking the connection with a set of neighboring nodes within a given radius. In this way, we can determine the closest sample and, thus, the tree to initially contain the new sample. When the search radius is large, the set of neighbors typically includes nodes of both trees, which increases the probability of finding a feasible solution. If the set of neighbors only includes nodes in one of the trees, an extra step can be added attempting to connect the new node to the closest one in the other tree. This can accelerate the convergence in some cases, without compromising the asymptotic optimality. This procedure is not detailed in Algorithm 1 for the sake of brevity.

The algorithm starts by initializing an atlas (Line 1), which will provide a parametrization of the space to explore and whose role will be detailed in Section 4. Next, the set of nodes maintained by the AtlasBiRRT*, $\mathcal{V}$, is initialized with the two query configurations (Line 2) and an initially empty set of edges, $\mathcal{E}$, is defined (line 3). The two maintained trees are represented using the parent relations, P, initialized in Lines 4 and 5. Moreover, each node in the tree has an associated value, C, indicating the cost of the best path to the root of the tree including the node (Lines 6 and 7). Finally, the optimal path, $\mathcal{P}$ is initially empty (Line 8) and its cost is set to infinity (Line 9), which by convention is the cost of unfeasible paths. At each iteration (Lines 10 to 29), AtlasBiRRT* generates a random sample (Line 11), extends the graph towards this sample creating a new node $\mathbf{x}_n$ (Line 13) and connects $\mathbf{x}_n$ to the graph with the collision-free edges to neighboring nodes (Lines 15 to 23), identifying the lowest-cost one (Lines 24 to 26) to define the parent relation (Lines 27 and 28).

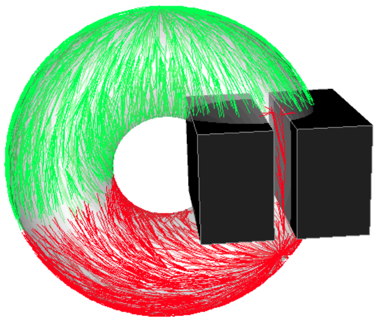
Figure 2: A bidirectional RRT* constructed on a torus-like manifold. The two RRT trees are shown in red and green, respectively.

The new nodes and their associated edges might be used to improve the cost of many nodes already in the graph. A similar situation occurs in heuristic search. Consider, for instance, a 2D grid with blocked and free cells where the objective is to find the best path from a start to a given goal cell. If one of the blocked cells becomes free, the cost-to-goal for many cells might need to be updated, possibly affecting cells that are not in contact to the changed cell. The updates, though, can be limited to the cells that are likely to be part of the optimal path, i.e., changes in cells far from the optimal path might have no effect at all. The grid can be associated to a graph where the valid transitions between cells define the edges and, then, heuristic search algorithms such as LPA* [9] can be used to efficiently update the costs when there are changes in some of the edges. Actually, in asymptotic-optimally path planning, adding a new node has the same affect as unblocking a cell in the grid example. Based on this idea, the AtlasBiRRT* planner uses the iterative rewire procedure detailed in Algorithm 2. The core of this procedure (Lines 6 to 10) is the same as the rewiring procedure originally proposed in the RRT* [6] and also used in [45]. However, here this core is applied not just for the immediate neighbors of the last node added to the graph, but for all the connected nodes for which the estimated total cost is lower than the cost of the best path found so far. The estimated total cost for a node is a lower bound of the cost

Algorithm 2: The Rewire procedure

input : The graph given by the nodes $\mathcal{V}$, and the edges $\mathcal{E}$, the parent relations P and the cost C for each node, the best path up to now $\mathcal{P}$, and its cost l, the query configurations $\mathbf{x}_s$ and $\mathbf{x}_g$, and the last added node, $\mathbf{x}_n$.

output: The possibly updated parent P and cost C for each node, and the new optimal path $\mathcal{P}$ and its length l.

```
1  $c \leftarrow C(\mathbf{x}_n) + \textsc{Heuristic}(\mathbf{x}_n, \textsc{Goal}(\mathbf{x}_n))$
2  $Q \leftarrow \{(\mathbf{x}_n, c)\}$
3  while $Q \neq \emptyset$ and $\textsc{MinCost}(Q) < l$ do
4      $\mathbf{x}_n \leftarrow \textsc{MinNode}(Q)$
5      $\textsc{RemoveFromQueue}(Q, \mathbf{x}_n)$
6      for $\mathbf{x} \in \textsc{Neighbours}(\mathbf{x}_n)$ do
7          $c_e \leftarrow \textsc{CostOfEdge}(\mathcal{E}, \mathbf{x}_n, \mathbf{x})$
8          if $C(\mathbf{x}_n) + c_e < C(\mathbf{x})$ then
9              $C(\mathbf{x}) \leftarrow C(\mathbf{x}_n) + c_e$
10             $P(\mathbf{x}) \leftarrow \mathbf{x}_n$
11             $c \leftarrow C(\mathbf{x}) + \textsc{Heuristic}(\mathbf{x}, \textsc{Goal}(\mathbf{x}))$
12             if $\mathbf{x} \in Q$ then
13                 $\textsc{UpdateInQueue}(Q, \mathbf{x}, c)$
14             else
15                 $Q \leftarrow Q \cup \{(\mathbf{x}, c)\}$
16         else
17             if $\textsc{Goal}(\mathbf{x}) \neq \textsc{Goal}(\mathbf{x}_n)$ then
18                 $c \leftarrow C(\mathbf{x}) + c_e + C(\mathbf{x}_n)$
19                 if $c < l$ then
20                     $\mathcal{P} \leftarrow \textsc{PathInGraph}(\mathcal{V}, P, \mathbf{x}, \mathbf{x}_n)$
21                     $l \leftarrow c$
```

of the path connecting $\mathbf{x}_s$ and $\mathbf{x}_g$ via that node. For this estimation, we take into account that the cost associated with each node only provides accurate information of the cost either to $\mathbf{x}_s$ or to $\mathbf{x}_g$. The cost to the other configuration is estimated using a given heuristic function that provides a lower bound of the cost between any pair of configurations (Lines 1 and 11). The function Goal appearing in Lines 1, 11, and 17 identifies the goal configuration for a given node, i.e., the query configuration that is not the root of the tree containing the node.

Adapting the LPA* strategy, the nodes potentially affected by the rewiring procedure are stored in a priority queue, Q. Each element in the queue is a pair including a node and its associated priority, which is given by the total estimated cost for the node (Line 1). This priority allows processing the most promising nodes first and discarding nodes that can not be part of the optimal path. Here, we take into account that the space to explore is continuous and, thus, with probability one, two randomly-sampled nodes will never have the same estimated total cost. This allows the use of a single-valued priority instead of the twofold used in LPA*. The adaptation is further simplified taking into account that the cost of the nodes can only decrease and that only a single node is added at each iteration. For each node in the queue with priority lower than l — the cost of the best path up to the moment— the algorithm checks if the neighboring nodes can improve their cost using it (Lines 6 to 11). If this is the case, the improved node is added to the queue to propagate the rewire from it in subsequent iterations (Lines 12 to 15). Note that the tree containing a node might change if the node is rewired. In this way the trees progressively adapt to their respective Voronoi regions, avoiding any overlap. Each edge connecting a pair of nodes, $\mathbf{x}$ and $\mathbf{x}_n$, in different trees gives a new path connecting the query configurations. Lines 17 to 21 check if this new path has to replace the best one found up to this point. Here, the PathInGraph procedure determines the best path from $\mathbf{x}_s$ to $\mathbf{x}_g$ via $\mathbf{x}$ and $\mathbf{x}_n$, which can be readily determined using the trees maintained by the planner.

The asymptotic optimality of AtlasBiRRT* planner is given by the fact that each of the two generated trees is an RRT* and, thus, they are guaranteed to asymptotically generate optimal paths to any configuration in the corresponding Voronoi region. Moreover, by storing the neighborhood relations between nodes, the graph generated in the AtlasBiRRT* planner is the same as that generated by the RRG algorithm [6] and, thus, the worst-case time and space complexity of both algorithms is the same, $O(|\mathcal{V}| \log |\mathcal{V}|)$.

4. Dealing with Manifolds

The AtlasBiRRT* planner presented in the previous section can be used in parametric spaces and it would provide solution paths more efficiently than the standard RRT* planner. However, it would not be able to deal with the constrained configuration spaces arising in many relevant applications. To devise a version of the planner for such spaces, its basic functions (SampleConf, NearestNode, Near, Steer, and Path) must be properly particularized. Next, we detail the issues that have to be considered for each function.

Function SampleConf (Line 11) generates a sequence of random samples in the configuration space. The distribution of these samples is relevant to fix the percolation threshold γ^*, a fundamental parameter that determines the span of the connections at each iteration.

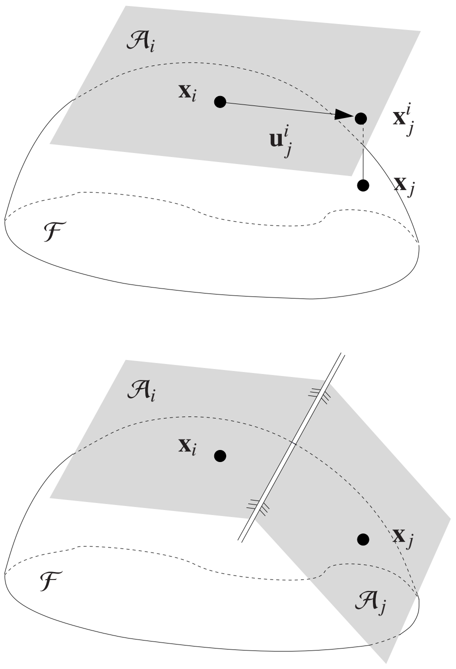

Figure 3: **Top** A generic approximation of the exponential map is obtained by orthogonally projecting to $\mathcal{F}$ a point $\mathbf{x}^i_j$ on the tangent space at $\mathbf{x}_i$. **Bottom** When a new chart is defined at $\mathbf{x}_j$, the applicability areas of the two charts, $\mathcal{A}_i$ and $\mathcal{A}_j$, are coordinated to avoid overlaps and to keep track of the part of the manifold already parametrized.

Functions NearestNode (Line 12) and Near (Line 16) rely on the ability to compute distances between configurations. When operating on a manifold, these distances should correspond to geodesic curves. However, a reasonable solution, that will be adopted herein, is to resort to the ambient space metric as an approximation of the geodesic one. Such metric allows the use of efficient search algorithm, like KD-trees [47]. When selecting the nearest node, this approximation may lead to few inadequate graph extensions, but when finding the set of nodes within a neighborhood, the ambient space metric is conservative, i.e., the returned set of nodes necessarily includes all the geodesic neighbors, although some of the returned elements might be actually farther than expected. Note that algorithms with sub-lineal computational complexity like KD-trees can not be used to determine the k-nearest nodes of a given configuration on the manifold. This makes impractical the use of planners such as the k-nearest RRT* [6], which have the advantage of not requiring a perlocation threshold.

Finally, functions Steer (Line 4) and Path (Lines 19) operate on collision-free, optimal paths. The determination of the lowest-cost path between configurations on the manifold can be arbitrarily difficult and, thus, the AtlasBiRRT* algorithm considers the geodesic path. Since the cost is bounded by the scaled path length, any optimal path can be piecewise approximated by geodesic segments with a bounded error, that in the limit vanishes.

Summarizing, the main challenges to operate on manifolds are the need to characterize the distribution of samples on such spaces, which is used when determining the percolation threshold, and the necessity to follow geodesic paths. These issues would be easily solved if a global isometric parametrization of the manifold was available, but these parametrizations do not exist for general manifolds. 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 forming an atlas [48]. Higher-dimensional continuation techniques provide principled numerical tools to compute the atlas for an implicitly-defined manifold reachable from a given point [44]. In this paper, we rely on such tools to define the basic operations of the AtlasBiRRT* planner.

4.1. 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

$$\mathcal{F} = \{\mathbf{x} \in \mathbb{R}^n \;:\; \mathbf{F}(\mathbf{x}) = \mathbf{0}\}, \tag{1}$$

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.

A chart, C_i, locally parametrizes the k-dimensional manifold around a given point, $\mathbf{x}_i$, with a bijective map between parameters $\mathbf{u}^i_j$ in $\mathbb{R}^k$ and n-dimensional points $\mathbf{x}_j$ on the manifold. The map from the parameter space to the manifold, $\mathbf{x}_j = \boldsymbol{\psi}_i(\mathbf{u}^i_j)$, is known as the exponential map and the inverse, $\mathbf{u}^i_j = \boldsymbol{\psi}_i^{-1}(\mathbf{x}_j)$, is the logarithmic map. A generic approximation of the exponential map valid for any manifold can be implemented using $T_{\mathbf{x}_i}\mathcal{F}$, the k-dimensional linear space tangent at $\mathbf{x}_i$ (see Fig. 3-top). An orthonormal basis for this tangent space is given by the $n \times k$ matrix, $\boldsymbol{\Phi}_i$, satisfying

$$\begin{bmatrix} \mathbf{J}(\mathbf{x}_i) \\ \boldsymbol{\Phi}_i^\top \end{bmatrix} \boldsymbol{\Phi}_i = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}, \tag{2}$$

with $\mathbf{J}(\mathbf{x}_i)$ the Jacobian of $\mathbf{F}$ evaluated at $\mathbf{x}_i$, and $\mathbf{I}$, the $k \times k$ identity matrix. Using $\boldsymbol{\Phi}_i$, the mapping $\boldsymbol{\psi}_i$ is defined by first computing the mapping $\boldsymbol{\phi}_i$ from parameters in the tangent space to coordinates in the joint ambient space,

$$\mathbf{x}^i_j = \boldsymbol{\phi}_i(\mathbf{u}^i_j) = \mathbf{x}_i + \boldsymbol{\Phi}_i\, \mathbf{u}^i_j \,, \tag{3}$$

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

$$\left.\begin{aligned} \mathbf{F}(\mathbf{x}_j) &= \mathbf{0}, \\ \boldsymbol{\Phi}_i^\top\,(\mathbf{x}_j - \mathbf{x}^i_j) &= \mathbf{0}, \end{aligned}\right\} \tag{4}$$

using a Newton procedure [49]. The logarithmic map can be approximated as the projection of a point on the tangent subspace,

$$\mathbf{u}^i_j = \boldsymbol{\Phi}_i^\top(\mathbf{x}_j - \mathbf{x}_i). \tag{5}$$

Using the exponential and logarithmic maps, a full atlas of the manifold can be defined starting from a given point [44]. In principle, one could rely on such atlas to determine an optimal path between any two given configurations using a fast marching like algorithm [50]. However, the construction of a full atlas is computationally demanding, specially in high dimensions. Therefore, we depart from our previous work [12], where the atlas construction is intertwined with the definition of an RRT.

Formally, if $\mathbf{x}_c$ is a configuration on $\mathcal{F}$ already included in any of the two trees maintained by AtlasBiRRT* (initially $\mathbf{x}_c$ is $\mathbf{x}_s$ or $\mathbf{x}_g$), C_c is the chart parametrizing this configuration, and $\mathbf{u}_c$ are the parameters of this configuration in C_c, then a new vector of parameters $\mathbf{u}_n$ is generated with a small displacement from $\mathbf{u}_c$ towards $\mathbf{u}_r$, a random vector of parameters on C_c, and the next point to add to the tree is obtained as $\mathbf{x}_n = \boldsymbol{\psi}_c(\mathbf{u}_n)$. However, the area of the manifold properly parametrized by a given chart is limited. As the RRT branch grows, i.e., as the norm of $\mathbf{u}_n$ increases, the distance and the curvature of the manifold with respect to the tangent space typically increase too, and the Newton process implementing $\boldsymbol{\psi}_c$ could even diverge. Thus, a new chart is defined on the last valid sample in the extended RRT branch whenever the Newton process of the new sample fails, or when this sample has a large error with respect to the manifold, i.e., when

$$\|\boldsymbol{\phi}_c(\mathbf{u}_n) - \mathbf{x}_n\| > \epsilon, \tag{6}$$

or when the curvature of the manifold with respect to C_c is large, i.e., when

$$\frac{\|\mathbf{u}_c - \mathbf{u}_n\|}{\|\mathbf{x}_c - \mathbf{x}_n\|} < \cos(\alpha), \tag{7}$$

Algorithm 3: Sampling on an atlas.

```
input  : An atlas, $A$.
output: A random point in the tangent space of a given
        chart.
1 repeat
2 |   $r \leftarrow$ RandomChartIndex($A$)
3 |   $\mathbf{u}_r \leftarrow$ RandomInBall($R_s$)
4 until $\mathbf{u}_r \in \mathcal{A}_r$
5 Return $\boldsymbol{\phi}_r(\mathbf{u}_r)$
```

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

$$\|\mathbf{u}_n\| > R, \tag{8}$$

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

To avoid the overlap between the portions of the manifold parametrized by neighboring charts, the area of applicability $\mathcal{A}_i$ of a given chart C_i is bounded by a set of linear inequalities, as illustrated in Fig. 3-bottom. These inequalities are defined in the tangent space associated with each chart and points not fulfilling them correspond to points on the manifold 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 C_i. If a new chart C_j is created on a point $\mathbf{x}_j$ then

$$2\, \mathbf{u}^\top \mathbf{u}^i_j \leq \|\mathbf{u}^i_j\|^2 \,, \tag{9}$$

is added to the set of inequalities bounding $\mathcal{A}_i$, with $\mathbf{u}^i_j = \boldsymbol{\psi}_i^{-1}(\mathbf{x}_j)$. This inequality bisects the vector $\mathbf{u}^i_j$, keeping the half-space including the origin.

4.2. Sampling and Node Connection on Manifolds

Using the atlas, the SampleConf procedure is implemented as described in Algorithm 3. A chart is selected at random with uniform distribution (Line 2) and then, a point, $\mathbf{u}_r$ is sampled within a ball of radius $R_s > R$ (Line 3). The process is repeated until $\mathbf{u}_r$ is inside the applicability area $\mathcal{A}_r$, i.e., until it fulfills the inequalities created by neighboring charts, if any (Line 4). Finally (Line 5), the process returns the ambient space coordinates for $\mathbf{u}_r$ computed using the $\boldsymbol{\phi}_r$ map defined in Eq. (3).

Algorithm 4 presents the Steer function using the atlas. This function gets as input the atlas maintained by AtlasBiRRT* and two points, $\mathbf{x}_n$ and $\mathbf{x}_r$, where $\mathbf{x}_n$ is a node in the graph and $\mathbf{x}_r$ is a random point obtained using SampleConf. First, the function determines the chart

Algorithm 4: The Steer procedure.

```
input : The atlas $A$, one node in the graph, $\mathbf{x}_n$, and one
        random node in the tangent space of a given
        chart $\mathbf{x}_r$.
output: 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  $\mathbf{u}_r \leftarrow \mathbf{u}_n + (\mathbf{u}_r - \mathbf{u}_n)\, d/\|\mathbf{u}_r - \mathbf{u}_n\|$
6  $\mathbf{x}_r \leftarrow \phi_c(\mathbf{u}_r)$
7  Blocked $\leftarrow$ False
8  while not Blocked and $d > 0$ do
9      $(\mathbf{x}_j, \mathbf{u}_j, c,$ Blocked, New$) \leftarrow$ NewConf$(A, c, \mathbf{x}_n, \mathbf{u}_r)$
10     if not Blocked then
11         if new then
12             $\mathbf{u}_r \leftarrow \psi_c^{-1}(\mathbf{x}_r)$
13             $\mathbf{u}_r \leftarrow \mathbf{u}_j + (\mathbf{u}_r - \mathbf{u}_j)\, d/\|\mathbf{u}_r - \mathbf{u}_j\|$
14             $\mathbf{x}_r \leftarrow \phi_c(\mathbf{u}_r)$
15         $d \leftarrow d - \|\mathbf{x}_n - \mathbf{x}_j\|$
16         $\mathbf{x}_n \leftarrow \mathbf{x}_j$
17 Return $\mathbf{x}_n$
```

parametrizing $\mathbf{x}_n$ (Line 1) and computes the parameters of $\mathbf{x}_n$ and $\mathbf{x}_r$ on this chart (Lines 2 and 3) ensuring that $\mathbf{u}_r$ is at least at distance d from $\mathbf{u}_n$ (Lines 5 and 6), with d the original distance between $\mathbf{x}_n$ and $\mathbf{x}_r$ (Line 4). Then, the NewConf procedure is used to generate a new configuration on the manifold, $\mathbf{x}_j$ in the direction given by $\mathbf{u}_r$ (Line 9). If during this step, a collision is detected, the Steer procedure is stopped. Otherwise, if NewConf generated a new chart, the random parameters, $\mathbf{u}_r$, are recomputed ensuring that the target is far enough from the previous point (Lines 12 and 13) and the associated random configuration is set accordingly (Line 14). Next, to avoid growing an infinite branch, the distance already travelled is discounted (Line 15) and the new configuration is set as the point from where to continue the path (Line 16). Finally, the Steer function returns the last configuration generated along the extension (Line 17).

The procedure NewConf that generates a new configuration in a given direction is detailed in Algorithm 5. The function linearly interpolates in parameter space with small steps of size δ (Line 2) between $\mathbf{u}_n$ (the parameters of $\mathbf{x}_n$ in chart c computed in Line 1) and the target point $\mathbf{u}_r$, and projects the resulting parameters to the manifold (line 3). In the algorithm, function Valid checks if this projection is actually successful. The

Algorithm 5: The NewConf procedure.

```
input : The atlas $A$, a chart, $c$, and a configuration $\mathbf{x}_n$ and
        a parameter vector $\mathbf{u}_r$ indicating the advance
        direction, both in chart $c$.
output: A new configuration $\mathbf{x}_j$, its parameters $\mathbf{u}_j$, the
        chart $c$ including $\mathbf{u}_j$, and a couple of flags,
        Blocked and new, indicating respectively if the
        path is blocked by an obstacle and if a new chart
        has been created.
1  $\mathbf{u}_n \leftarrow \psi_c^{-1}(\mathbf{x}_n)$
2  $\mathbf{u}_j \leftarrow (\mathbf{u}_r - \mathbf{u}_n)\, \delta/\|\mathbf{u}_r - \mathbf{u}_n\|$
3  $\mathbf{x}_j \leftarrow \psi_c(\mathbf{u}_j)$
4  new $\leftarrow$ False
5  if Valid($\mathbf{x}_j$) and Collision($\mathbf{x}_j$) then
6      Blocked $\leftarrow$ True
7  else
8      Blocked $\leftarrow$ False
9      if not Valid($\mathbf{x}_j$) or $\|\phi_c(\mathbf{u}_j) - \mathbf{x}_j\| > \epsilon$ or
       $\|\mathbf{u}_n - \mathbf{u}_j\|/\|\mathbf{x}_n - \mathbf{x}_j\| < \cos(\alpha)$ or $\|\mathbf{u}_j\| > R$ then
10         $c \leftarrow$ NewChart$(A, \mathbf{x}_n)$
11         $\mathbf{x}_j \leftarrow \mathbf{x}_n$
12         new $\leftarrow$ True
13     else
14         if $\mathbf{u}_j \notin \mathcal{A}_c$ then
15             $c \leftarrow$ NeighborChart$(c, \mathbf{u}_j)$
16             new $\leftarrow$ True
17     if new then
18         $\mathbf{u}_j \leftarrow \psi_c^{-1}(\mathbf{x}_j)$
19 Return $(\mathbf{x}_j, \mathbf{u}_j, c,$ Blocked, New$)$
```

value of δ should be small enough so that there are not undetected collisions nor large curvature changes between two consecutive configurations along a path. If the new configuration is not in collision, the algorithm checks if it triggers the creation of a new chart (Line 9) or if it is in the applicability area of a neighboring chart (Line 14). In any of these two cases, $\mathbf{u}_j$ is recomputed (Line 18) projecting $\mathbf{x}_j$ on the new or neighbor chart determined at Lines 10 and 15, respectively.

The function Path detailed in Algorithm 6 is similar to the Steer function, although in this case the goal is not a random point on the tangent space of a given chart, but a node already included in the graph, i.e., a configuration on the manifold. Thus, function Path does not include the projection and the displacement of the sample, since the goal configuration is used, whenever necessary (Lines 2 and 9). At the end of the Path procedure, the sequence of configurations connecting the query points is returned (Line 13), unless the goal configuration is not actually reached (Line 15).

Algorithm 6: The PATH procedure.

```
input  : The atlas $A$ and two nodes in the graph, $\mathbf{x}_n$ and $\mathbf{x}_r$.
output : A collision-free path connecting the two nodes, if found.
1  $c \leftarrow$ CHARTINDEX($\mathbf{x}_n$)
2  $\mathbf{u}_r \leftarrow \boldsymbol{\psi}_c^{-1}(\mathbf{x}_r)$
3  BLOCKED $\leftarrow$ FALSE
4  $\mathcal{P} \leftarrow \emptyset$
5  while not BLOCKED and $\|\mathbf{u}_n - \mathbf{u}_r\| > \delta$ do
6      $(\mathbf{x}_j, \mathbf{u}_j, c,$ BLOCKED, NEW$) \leftarrow$ NEWCONF($A, c, \mathbf{x}_n, \mathbf{u}_r$)
7      if not BLOCKED then
8          if NEW then
9              $\mathbf{u}_r \leftarrow \boldsymbol{\psi}_c^{-1}(\mathbf{x}_r)$
10         $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathbf{x}_j\}$
11         $\mathbf{x}_n \leftarrow \mathbf{x}_j$
12 if $\|\mathbf{x}_n - \mathbf{x}_r\| < \delta$ then
13     RETURN $\mathcal{P}$
14 else
15     RETURN $\emptyset$
```

The increment of computational complexity in the AtlasBiRRT* planner caused by the use of the atlas appears in Algorithm 5 and concentrates on the computation of the mapping $\boldsymbol{\psi}_c$ (Line 3) and in the addition of new charts to the atlas (Line 10). 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 a new chart and to identify the neighboring charts in the atlas to avoid the overlaps. The former operation is $O(n^3)$ since it is implemented using a QR decomposition and the latter can be implemented using hierarchical structures, with a cost that is logarithmic in the number of charts in the atlas. Since n is constant and small compared to $|\mathcal{V}|$ and the number of charts in the atlas is bounded in the long term, the added complexity does not affect the asymptotic computational performance of the planner.

4.3. Asymptotic Optimality on Manifolds

As previously mentioned, the asymptotic optimality of a planner on manifolds relies on the ability to define the right percolation threshold, and to connect configurations with geodesic paths. Next, we show how these issues are solved thanks to the atlas structure.

With respect to the percolation threshold, the existing results for globally-parametrizable spaces can be directly applied to the local parametrization provided by a given chart. Using Eq. (7) 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

$$\|\mathbf{x}_i - \mathbf{x}_j\| \leq \sec(\alpha)\, \|\mathbf{u}_i - \mathbf{u}_j\|. \tag{10}$$

Thus, there is a bounded distortion between points in the tangent space and the associated points on the manifold. Therefore, the volume of the part of the manifold covered by a given chart is a scaled factor of the volume of the corresponding applicability area. Adapting Theorem 38 and algorithm 6 in [6], the critical percolation value for chart c, γ_c^*, and the associated search radius at each step, γ_c, are such that

$$\gamma_c^* > \left[2\left(1+\frac{1}{k}\right)\frac{\mu_F(\mathcal{A}_c)}{\zeta_k}\sec(\alpha)\right]^{1/k}, \tag{11}$$

and

$$\gamma_c = \gamma_c^*\left[\frac{\log|\mathcal{V}_c|}{|\mathcal{V}_c|}\right]^{1/k}, \tag{12}$$

where $\mu_F(\mathcal{A}_c)$ is the Lebesgue measure of the applicability area of the chart corresponding to collision free configurations, ζ_k is the volume of the unitary k-dimensional ball and $\mathcal{V}_c$ the set of nodes in the chart.

However, taking into account only one chart gives a myopic view of the whole space to explore and, thus, a too small value for the search radius. Actually, the underestimation can be analytically derived as follows. From [6], the global search radius would be

$$\gamma = \left[2\left(1+\frac{1}{k}\right)\frac{\mu_F(\mathcal{F})}{\zeta_k}\frac{\log|\mathcal{V}|}{|\mathcal{V}|}\right]^{1/k}. \tag{13}$$

Now, if r_v is the ratio between the volume of the collision-free region covered by the considered chart and the full collision-free configuration space, that can be approximated as

$$r_v = \frac{\mu_F(\mathcal{A}_c)\sec(\alpha)}{\mu_F(\mathcal{F})}, \tag{14}$$

and r_s is the ratio of samples included in this chart,

$$r_s = \frac{|\mathcal{V}_c|}{|\mathcal{V}|}, \tag{15}$$

then, the global search radius can be rewritten as

$$\gamma = \gamma_c\left[\frac{r_s}{r_v}\frac{\log|\mathcal{V}|}{\log|\mathcal{V}_c|}\right]^{1/k} \tag{16}$$

$$= \gamma_c\left[\frac{r_s}{r_v}\frac{\log|\mathcal{V}|}{\log r_s + \log|\mathcal{V}|}\right]^{1/k}. \tag{17}$$

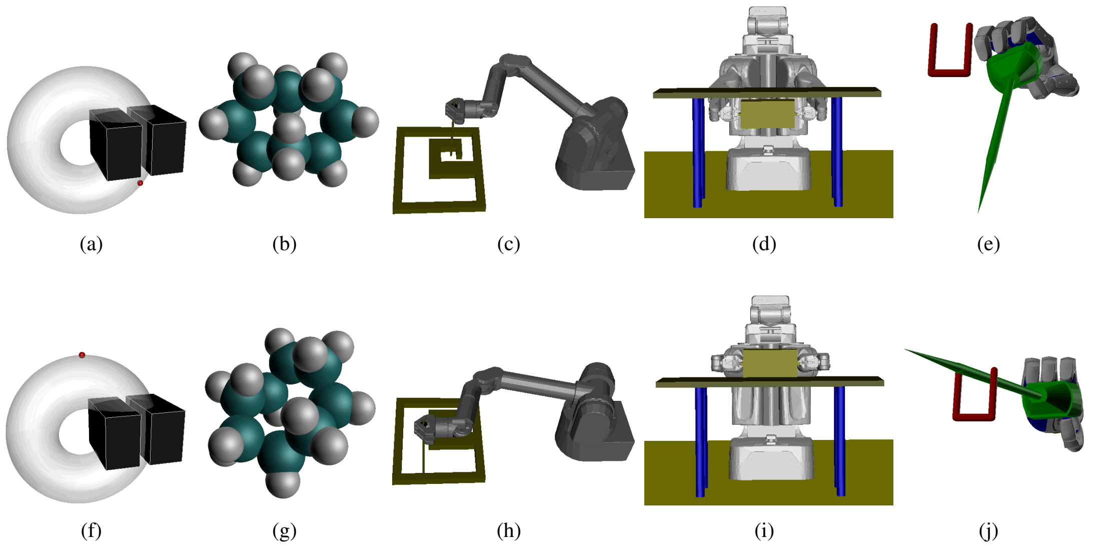

Figure 4: The five benchmarks used in this paper. (a,f) A ball moving on a torus. (b,g) The cyclooctane molecule. (c,h) The Barrett arm with a peg escaping a dead-end. (d,i) The PR2 service robot moving a box with the two arms. (e,j) A robot anthropomorphic hand moving a needle among obstacles. The figures in the top and bottom rows correspond to the start and goal configurations, respectively.

This offers the possibility of using chart-based perlocation thresholds, which only require local uniform distribution of samples. However, the estimation of r_v can be troublesome. If it was feasible to generate a uniform distribution of samples on the manifold, then r_v and r_s would be the same and, therefore, γ_c^* and γ^* would converge to the same value, in the long term.

Fortunately, the atlas allows obtaining a close to uniform distribution of samples in the part of the manifold already explored. To this end, Algorithm 3 selects a chart at random, samples 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 which translates to a close to uniform distribution on the manifold, with the distorsion bounded by Eq. (10). The fact that the atlas is built together with the exploration trees has an effect on the distribution of the samples. In the worst case, though, the distribution will be uniform from the moment the atlas fully parametrizes the free configuration space, and thus, this factor has influence on the initial stages of the search, but not on the long-term optimality.

Finally, the atlas parametrization provides 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 c and the corresponding path on the manifold $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ with $\mathbf{x}_i = \boldsymbol{\psi}_c(\mathbf{u}_i)$, $i \in \{1, \ldots, m\}$. Then, the length of the path can be approximated by

$$p = \sum_{i=2}^{m} \|\mathbf{x}_{i-1} - \mathbf{x}_i\|, \tag{18}$$

and its length in the parameter space is

$$l = \sum_{i=2}^{m} \|\mathbf{u}_{i-1} - \mathbf{u}_i\| = \|\mathbf{u}_1 - \mathbf{u}_m\|. \tag{19}$$

Note that $l \leq p$ and that, using Eq. (10), $p \leq \sec(\alpha)\ l$. Moreover, assume that p^* is the length of the geodesic path connecting the two points and l^* is its length in the parameter space. Since paths are defined as straight lines in this space, we have $l < l^*$ and, thus

$$l \leq l^* \leq p^* \leq p \leq \sec(\alpha)\ l \leq \sec(\alpha)\ l^*. \tag{20}$$

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

$$\frac{p - p^*}{p^*} \leq \frac{\sec(\alpha)\ l - l}{p^*} \leq \sec(\alpha) - 1. \tag{21}$$

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 beyond the scope of a given chart by generating new charts, when necessary.

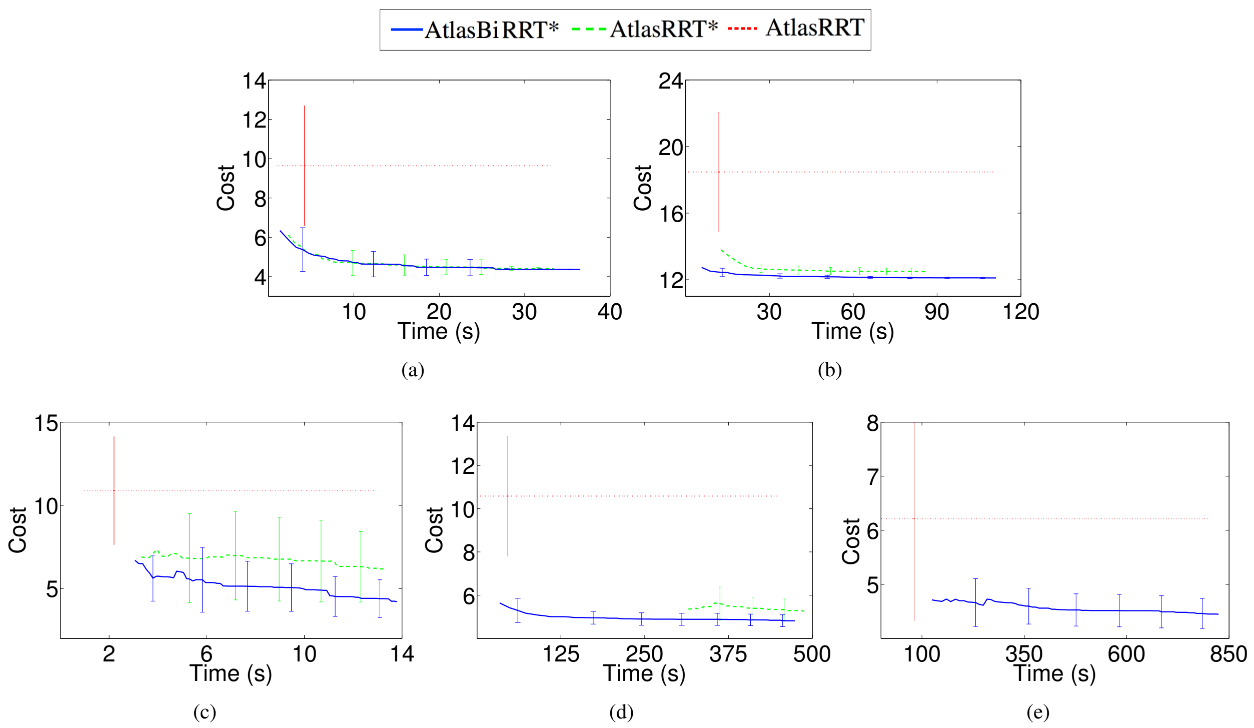

Figure 5: Path cost versus execution time for the AtlasBiRRT*, AtlasRRT*, and AtlasRRT planners on (a) the torus, (b) the cyclooctane, (c) the Barrett, (d) the PR2, and (e) the Robot Hand problems averaged over 25 executions. Costs are given once a solution is found in at least 70% of the repetitions. In the Robot Hand problem the AtlasRRT* planner founds the solution in only one of the 25 repetitions and, thus, it is not shown in the corresponding plot. In the plots, the vertical bars give the standard deviation.

Benchmark	k	n	Relative Error	γ^*
Torus	2	3	0.01	10
Cyclooctane	2	8	0.007	12
Barrett	3	9	0.05	2.5
PR2	4	16	0.14	4
Robot Hand	5	23	0.07	2.5

Table 1: Dimension of the configuration and ambient spaces, and relative errors respect to the optimal path obtained with AtlasBiRRT* after 1000 iterations using the given values for γ^*. In these experiments, collisions are not considered.

5. Experiments and Results

Figure 4 shows the five benchmarks used to evaluate the AtlasBiRRT* algorithm. The first one involves a small ball moving on a implicitly-defined torus with two obstacles forming a narrow corridor. The simplicity of this benchmark facilitates the interpretation of the results. The second test case is the cyclooctane, a molecule that can be modelled with eight revolute joints forming a kinematic loop that defines a configuration space with a Klein bottle topology [51]. In this problem, collisions correspond to steric clashes that appear whenever two atoms are closer than the sum of their Van der Waals radii. This example is used to illustrate the ability of AtlasBiRRT* to determine the optimal path among many feasible ones. In the third example, the Barrett arm has to move a peg out of a dead-end. The task is constrained because the peg can not rotate about its vertical axis and it must remain orthogonal to and in contact with the maze plane. The fourth test case is the PR2 robot executing a coordinated manipulation task to move a box from underneath a table and to place it on the table. Finally, the last test case is a robotic anthropomorphic hand moving a needle avoiding a U-shaped obstacle that introduces local minima in the path planning process. The last three 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 $\delta = 0.05$, $R = 0.4$, $R_s = 1$, $\epsilon = 0.1$, and $\alpha = 0.45$ rad. With such parameters, the error factor with respect to geodesic given by Eq. (21) is below $\sec(\alpha) - 1 = 0.1$, which is reasonably small. The value of γ^* depends on the volume of the collision-free space, which is hard to evaluate. Therefore, the value for this parameter is determined experimentally for each case. Finally, the Euclidean distance in the embedding joint ambient space is used as heuristic, when necessary. 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 [52].

Exceptionally, in the first set of experiments summarized in Table 1, collisions are not considered and, thus, the constraints only arise from the manifold structure of the configuration spaces. The table gives the dimension of the configuration space, k, the dimension of the ambient space, n, and the average relative error of AtlasBiRRT* with respect to the optimal path after $N = 1000$ iterations using the given values for γ^*. The optimal path is obtained by smoothing the best path returned by AtlasBiRRT*. We can see that AtlasBiRRT* converges to the optimal path in all cases, within the tolerance given by parameter α.

Next, to assess the performance of the proposed planner, we compared it with AtlasRRT [12], a sampling-based path planner that efficiently determines feasible paths on manifolds, but without optimizing them in any sense. Moreover, we also compared the results with the AtlasRRT* planner introduced in [45], which, up to our knowledge, is the only previous asymptotically-optimal planner able to deal with manifolds. Figure 5 shows the cost of the paths obtained with the three planners for the five benchmarks of Fig. 4, considering collisions and executing $N = 10000$ iterations. In the figures, the costs are plotted when at least 70% of the repetitions are successful and, thus, the eventual increments of the cost are caused by the different data averaged at each iteration.

In all the cases, the path obtained with AtlasRRT has a high cost and it is not improved once discovered. The path obtained with AtlasRRT* would eventually converge to the optimal path, but the convergence rate is slow and the time required to identify a first feasible solution can be long. In contrast, AtlasBiRRT* identifies such solution in few iterations and rapidly improves it until the optimal path is determined. Whereas in simple problems, like the torus one, the difference between AtlasRRT* and AtlasBiRRT* planners is minor, in more complex problems the advantage of using the new planner is remarkable. For instance, in the Robot Hand example, AtlasRRT* only was successful once in 25 executions, while AtlasBiRRT* finds the solution in all cases. In the experiments, the eventual extra time required by AtlasBiRRT* with respect to AtlasRRT* to complete the 10000 iterations is due to the management of the bidirectional trees and, specially, to the iterative rewiring procedure. However, the results show that the computational burden of these additional operations is not significant, specially in complex problems.

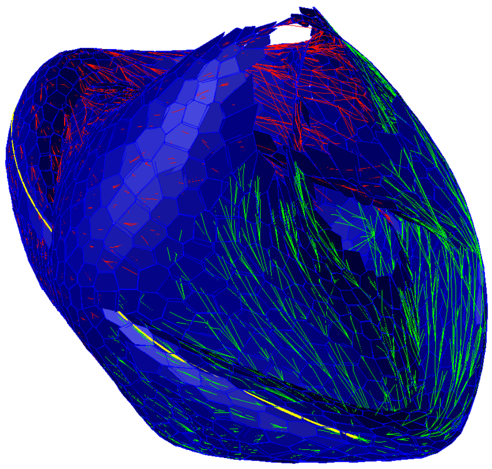
Figure 6: The atlas generated in the cyclooctane example (in blue) where each polygon is a chart, the two RRT* (in green and red), and the optimal path (in yellow).

Figure 6 shows the typical atlas and RRTs obtained for the case of the cyclooctane. Here, the configuration space is only two-dimensional and thus, the atlas and the trees end up covering all the region accessible from the root nodes. However, note the presence of charts not entirely enclosed by other charts. These charts are in the frontier of the collision regions, which are not parametrized by the atlas.

Table 2 provides an account of the memory used by the tree planners compared in this paper. First, we can note that AtlasRRT uses a significantly small amount of memory since it stops as soon as a solution is found and, thus it generates few samples and builds a small atlas. In contrast, both AtlasRRT* and AtlasBiRRT* refine the path after finding a first solution and, thus, they generate more samples and charts. Since AtlasBiRRT* actually maintains a graph and not just a tree, its uses more memory. This increment, though, is worth taking into account the performance of this planner.

Finally, as described along this paper, a minimal value of γ^* is required to ensure the asymptotic optimality. However, an overly conservative value for this parameter will strongly degrade the performance of the planner. Figure 7 shows the performance of AtlasBiRRT* with different values for γ^* in the torus example for $N = 10000$ iterations. Clearly, the larger the value of this parameter, the better the approximation of the optimal path path. However, as shown in Figs. 7 (b) and (c), the execution time and the memory used increase significantly when γ^* grows. Thus, this parameter provides a mechanism to balance the optimality and the performance of the planner.

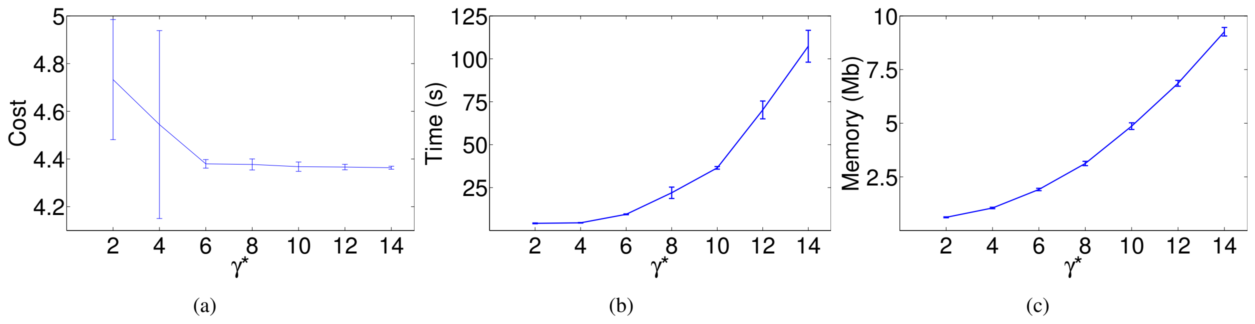

Figure 7: Path cost, execution time (in seconds), and memory use in megabytes (Mb) for the AtlasBiRRT* planner on the torus example for different values of γ^* after 10000 iterations.

Benchmark	AtlasRRT	AtlasRRT*	AtlasBiRRT*
Torus	0.03	0.45	4.51
Cyclooctane	0.09	1.18	2.70
Barrett	0.004	0.25	0.48
PR2	0.007	19.14	21.98
Robot Hand	0.24	4.25	5.41

Table 2: Average memory used in megabytes for the three planners compared in this paper.

6. Conclusions

In this paper we have introduced an asymptotically-optimal sampling-based path planner able to efficiently operate on implicitly-defined configuration spaces. This is achieved by using a bidirectional search strategy, a full propagation of the eventual improvements introduced with each new sample, and relying on higher-dimensional continuation tools to define an atlas parametrizing the configuration space manifold. Thanks to the atlas, we can be characterize the distribution of the samples on the manifold and we can determine close-to-geodesic paths for generic manifolds.

Several issues arise from the work presented in this paper. First, it would be necessary to investigate practical ways to determine an adequate value for the γ^* parameter for each problem or to provide mechanisms to automatically tune it. Moreover, we would like to apply the proposed planner with cost functions other than the length of the path. In this sense, we are already working on singularity-free path planning [53] where the cost is given by the proximity to the singularities, i.e., the set of configurations where the kinetostatic performance of a manipulator gets dramatically altered. Finally, we would like to explore the possibility of including dynamical aspects in the planner so that the resulting paths are adequate for direct execution, without requiring any futher post-process.

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 European Social Fund.

References

[1] S. M. LaValle, Planning Algorithms, Cambridge University Press, New York, 2006.

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

[3] L. E. Kavraki, P. Svestka, J.-C. Latombe, M. H. Overmars, Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces, IEEE Transactions on Robotics and Automation 12 (1996) 566–580.

[4] S. M. LaValle, J. J. Kuffner, Rapidly-Exploring Random Trees: Progress and Prospects, in: Algorithmic and Computational Robotics - New Directions, 2000, pp. 293–308.

[5] Kineo Computer Aided Motion web page, http://www.kineocam.com, Last accessed March-2013.

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

[7] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, D. Kragic, Dual Arm Manipulation: A Survey, Robotics and Autonomous Systems 60 (10) (2012) 1340–1353.

[8] N. Daoud, J. Gazeau, S. Zeghloul, M. Arsicault, A Real-Time Strategy for Dexterous Manipulation: Fingertips Motion Planning, Force Sensing and Grasp Stability, Robotics and Autonomous Systems 60 (3) (2012) 377–386.

[9] S. Koenig, M. Likhachev, D. Furcy, Lifelong Planning A*, Artificial Intelligence 155 (1–2) (2004) 93–146.

[10] J. M. Porta, L. Jaillet, Path Planning on Manifolds using Randomized Higher-Dimensional Continuation, Workshop on the Algorithmic Foundations of Robotics (2010) 337–353.

[11] J. M. Porta, L. Jaillet, O. Bohigas, Randomized Path planning on Manifolds based on Higher-Dimensional Continuation, The International Journal of Robotics Research 31 (2) (2012) 201–215.

[12] L. Jaillet, J. Porta, Path Planning Under Kinematic Constraints by Rapidly Exploring Manifolds, IEEE Transactions on Robotics 29 (1) (2013) 105–117.
[13] S. J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice-Hall, 2010.
[14] M. Dakulović, I. Petrović, Two-way D* algorithm for path planning and replanning, Robotics and Autonomous Systems 59 (5) (2011) 329–342.
[15] K. Gochev, A. Safonova, M. Likhachev, Planning with Adaptive Dimensionality for Mobile Manipulation, in: IEEE International Conference on Robotics and Automation, 2012, pp. 2944–2951.
[16] J. Kim, R. A. Pearce, N. M. Amato, Extracting Optimal Paths from Roadmaps for Motion Planning, in: IEEE International Conference on Robotics and Automation, 2003, pp. 2424–2429.
[17] R. Geraerts, M. H. Overmars, Creating High-Quality Paths for Motion Planning, The International Journal of Robotics Research 26 (2007) 845–863.
[18] P. C. Chen, Y. K. Hwang, SANDROS: a Dynamic Graph Search Algorithm for Motion Planning, IEEE Transactions on Robotics and Automation 14 (3) (1998) 390–403.
[19] N. Ratliff, M. Zucker, J. A. Bagnell, S. Srinivasa, CHOMP: Gradient Optimization Techniques for Efficient Motion Planning, in: IEEE International Conference on Robotics and Automation, 2009, pp. 489–494.
[20] R. Guernane, N. Achour, Generating Optimized Paths for Motion Planning, Robotics and Autonomous Systems 59 (10) (2011) 789–800.
[21] A. Shukla, E. Singla, P. Wahi, B. Dasgupta, A Direct Variational Method for Planning Monotonically Optimal Paths for Redundant Manipulators in Constrained Workspaces, Robotics and Autonomous Systems 61 (2) (2013) 209–220.
[22] L. Jaillet, J. Cortés, T. Siméon, Sampling-based Path Planning on Configuration-Space Costmaps, IEEE Transactions on Robotics 26 (4) (2010) 635–646.
[23] B. Akgun, M. Stilman, Sampling Heuristics for Optimal Motion Planning in High Dimensions, in: International Conference on Intelligent Robots and Systems, 2011, pp. 2640 –2645.
[24] R. Alterovitz, S. Patil, A. Derbakova, Rapidly-Exploring Roadmaps: Weighing Exploration vs. Refinement in Optimal Motion Planning, in: IEEE International Conference on Robotics and Automation, 2011, pp. 3706–3712.
[25] O. Arslan, P. Tsiotras, An Efficient Sampling-based Algorithm for Motion Planning with Optimality Guarantees, Technical Report DCSL-12-09-010, Georgia Institute of Technology, School of Aerospace Engineering (2012).
[26] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, S. Teller, Anytime Motion Planning using the RRT*, in: IEEE International Conference on Robotics and Automation, 2011, pp. 1478–1483.
[27] J. D. Marble, K. E. Bekris, Asymptotically Near-Optimal is Good Enough for Motion Planning, in: International Symposium on Robotics Research, 2011, in press.
[28] L. Han, L. Rudolph, J. Blumenthal, I. Valodzin, Convexly Stratified Deformation Spaces and Efficient Path Planning for Planar Closed Chains with Revolute Joints, The International Journal of Robotics Research 27 (11-12) (2008) 1189–1212.
[29] T. Siméon, J. P. Laumond, J. Cortés, A. Sahbani, Manipulation Planning with Probabilistic Roadmaps, The International Journal of Robotics Research 23 (7-8) (2004) 729–746.
[30] J. M. Porta, CuikSlam: A Kinematics-based Approach to SLAM, in: IEEE International Conference on Robotics and Automation, 2005, pp. 2436–2442.
[31] J.-P. Merlet, Parallel Robots, Springer, 2006.
[32] C. Rosales, L. Ros, J. M. Porta, R. Suárez, Synthesizing Grasp Configurations with Specified Contact Regions, The International Journal of Robotics Research 30 (4) (2011) 431–443.
[33] A. Rodríguez, L. Basañez, E. Celaya, A Relational Positioning Methodology for Robot Task Specification and Execution, IEEE Transactions on Robotics 24 (3) (2008) 600–611.
[34] G. Ballantyne, F. Moll, The da Vinci Telerobotic Surgical System: The Virtual Operative Field and Telepresence Surgery, Surgical Clinics of North America 83 (6) (2003) 1293–1304.
[35] W. J. Wedemeyer, H. Scheraga, Exact Analytical Loop Closure in Proteins Using Polynomial Equations, Journal of Computational Chemistry 20 (8) (1999) 819–844.
[36] R. Kimmel, J. A. Sethian, Computing Geodesic Paths on Manifolds, Proceedings of the National Academy of Sciences 95 (15) (1998) 8431–8435.
[37] F. Mémoli, G. Sapiro, Fast Computation of Weighted Distance Functions and Geodesics on Implicit Hyper-surfaces, Journal of Computational Physics 173 (2001) 730–764.
[38] D. S. Yershov, S. M. LaValle, Simplicial Dijkstra and A* Algorithms for Optimal Feedback Planning, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 3862–3867.
[39] T. Igarashi, M. Stilman, Homotopic Path Planning on Manifolds for Cabled Mobile Robots, in: International Workshop on the Algorithmic Foundations of Robotics, 2010, pp. 1–18.
[40] L. Han, N. M. Amato, A Kinematics-Based Probabilistic Roadmap Method for Closed Chain Systems, in: Algorithmic and Computational Robotics - New Directions, 2000, pp. 233–246.
[41] M. Stilman, Global Manipulation Planning in Robot Joint Space With Task Constraints, IEEE Transactions on Robotics 26 (3) (2010) 576–584.
[42] D. Berenson, S. Srinivasa, J. Kuffner, Task Space Regions: A Framework for Pose-Constrained Manipulation Planning, The International Journal of Robotics Research 30 (12) (2011) 1435–1460.
[43] Z. Yao, K. Gupta, Path Planning with General End-effector Constraints, Robotics and Autonomous Systems 55 (4) (2007) 316–327.
[44] M. E. Henderson, Multiple Parameter Continuation: Computing Implicitly Defined k-Manifolds, International Journal of Bifurcation and Chaos 12 (3) (2002) 451–476.
[45] L. Jaillet, J. M. Porta, Asymptotically-optimal Path Planning on Manifolds, in: Robotics: Science and Systems, 2012, pp. 145–152.
[46] J. J. Kuffner, S. M. LaValle, RRT-connect: An efficient approach to single-query path planning, in: IEEE International Conference on Robotics and Automation, 2000, pp. 995–1001.
[47] A. Yershova, S. M. LaValle, Improving Motion Planning Algorithms by Efficient Nearest Neighbor Searching, IEEE Transactions on Robotics 23 (1) (2007) 151–157.
[48] M. do Carmo, Differential Geometry of Curves and Surfaces, Prentice-Hall, 1976.
[49] W. C. Rheinboldt, MANPACK: A Set of Algorithms of Computations on Implicitly Defined Manifolds, Computers and Mathematics with Applications 32 (12) (1996) 15–28.
[50] S. Garrido, M. Malfaz, D. Blanco, Application of the Fast Marching Method for Outdoor Motion Planning in Robotics, Robotics and Autonomous Systems 61 (2) (2013) 106–114.
[51] S. Martin, A. Thompson, E. A. Coutsias, J.-P. Watson, Topology of Cyclo-octane Energy Landscape, Journal of Chemical Physics 132 (2010) 234115.
[52] KRD Group, The CuikSuite software, http://www.iri.upc.edu/cuik, Last accessed March-2013.
[53] O. Bohigas, M. Henderson, L. Ros, J. M. Porta, A Singularity-free Path Planner for Closed-chain Manipulators, in: IEEE International Conference on Robotics and Automation, 2012, pp. 2128–2134.

Léonard Jaillet received the Engineering degree in Mechanical Engineering from the Institut Superieur de Mecanique de Paris, Paris, France, and the Ph.D. degree in Robotics from the University of Toulouse, Toulouse, France, in 2001 and 2005, respectively. Since 2008, he has been a postdoctoral fellow at the Institut de Robòtica i Informàtica Industrial, Spanish National Research Council, Barcelona, Spain. His current research interests include motion planning for complex robotic systems and molecular simulations for structural biology.

Josep M. Porta received the Engineering degree in Computer Science in 1994, and the Ph.D. degree (with honors) in Artificial Intelligence in 2001, both from the Universitat Politècnica de Catalunya, Spain. From 2001 to 2003 he held a postdoctoral position at the University of Amsterdam, pursuing research in autonomous robot localization using vision. Currently, he is an Associate Researcher of the Spanish National Research Council at the Institut de Robòtica i Informàtica Industrial (IRI, CSIC-UPC), Barcelona, Spain. His current research interests include planning under uncertainty and computational kinematics.