electronics

MDPI

Article

Learning the Metric of Task Constraint Manifolds for Constrained Motion Planning

Fusheng Zha [1], Yizhou Liu [1], Wei Guo [1], Pengfei Wang [1,2,*], Mantian Li [1] and Xin Wang [2] and Jingxuan Li [1]

[1] State Key Laboratory of Robotics and Systems, Harbin Institute of Technology, Harbin 150001, China; zfsh751228@163.com (F.Z.); sduliuyz@163.com (Y.L.); wguo01@hit.edu.cn (W.G.); limt@hit.edu.cn (M.L.); jingxuanli_chn@163.com (J.L.)

[2] Shenzhen Academy of Aerospace Technology, Shenzhen 518000, China; xinwanghit07s@gmail.com

* Correspondence: wangpengfei@hit.edu.cn, Tel: +86-451-86414462

Received: 11 September 2018; Accepted: 3 December 2018; Published: 5 December 2018

Abstract: Finding feasible motion for robots with high-dimensional configuration space is a fundamental problem in robotics. Sampling-based motion planning algorithms have been shown to be effective for these high-dimensional systems. However, robots are often subject to task constraints (e.g., keeping a glass of water upright, opening doors and coordinating operation with dual manipulators), which introduce significant challenges to sampling-based motion planners. In this work, we introduce a method to establish approximate model for constraint manifolds, and to compute an approximate metric for constraint manifolds. The manifold metric is combined with motion planning methods based on projection operations, which greatly improves the efficiency and success rate of motion planning tasks under constraints. The proposed method Approximate Graph-based Constrained Bi-direction Rapidly Exploring Tree (AG-CBiRRT), which improves upon CBiRRT, and CBiRRT were tested on several task constraints, highlighting the benefits of our approach for constrained motion planning tasks.

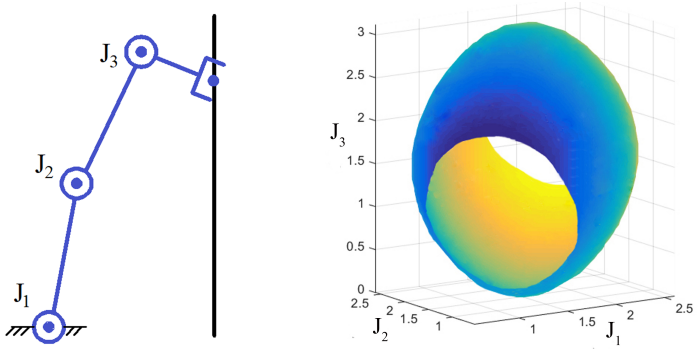
Keywords: motion planning; constraint manifolds; approximate metric; projection

1. Introduction

Since multi-degree-of-freedom (multi-DOF) robots are often subject to some hard kinematic constraints in real tasks, each constraint forms a low-dimensional manifold embedded in high-dimensional space, which poses a great challenge to traditional sampling-based motion planning techniques. For example, a robot may need to keep the orientation fixed when carrying an object, a robot may form a closed kinematic chain with the environment (when opening a door or pulling a drawer), or some tasks require cooperation of multiple manipulators. Due to task constraints, robots cannot move freely in the original configuration space.

Compared with artificial potential field methods [1], heuristic search techniques [2] and optimization-based methods [3,4], sampling-based motion planning algorithms such as RRT [5], PRM [6] and RRT* [7] have been shown to be effective for high-dimensional robots, and become the standard for industrial solutions [8]. They rely on the feasibility information of configurations provided by the collision query module, and connect a series of collision-free configurations to generate a feasible path from the starting point to the target region, which avoids explicit description of obstacles in high-dimensional space. However, the manifold generated by hard kinematic constraints introduces several challenges to the sampling-based motion planners. First, it is almost impossible to obtain configurations satisfying the constraints by direct sampling, because the constraint manifold has zero measure with respect to the ambient space metric, as shown in Figure 1. Second, the manifold is

implicitly defined by constraints in robot's workspace, and the mapping between the workspace and the configuration space is nonlinear, which means that the constraint manifolds cannot be explicitly described by an analytic formula. Third, sampling-based motion planners cannot interpolate locally under task constraints to form a continuous path on manifolds.



(**a**) 3-DOF robot (**b**) Constraint manifold

Figure 1. (**a**) The end effector of a three-DOF robot is constrained on the black line, which means that the robot has only two-DOF in the workspace. (**b**) The configurations satisfying the constraint form a two-dimensional manifold in the C-space. Obviously, This manifold has zero measure with respect to ambient space metric and can not be expressed by an analytic formula, because the mapping between configuration space and workspace is nonlinear.

To solve the above-mentioned problems, various methods of motion planning on constraint manifold are widely proposed, including relaxation [9,10], projection [11–13], tangent-space [14] and atlas [15,16]. The characteristics of these methods are summarized and compared in [17]. Relaxation is to relax the surface of the constraint manifold by increasing the allowed tolerance of the constraint function so that the sampling-based planners can handle the constraints without additional machinery. However, relaxation transmutes task constraint into narrow passage problem, which poses a challenge to the efficiency of constrained motion planning. Tangent space and atlas both create a piece-wise linear approximation of the manifold, which can then be used for generating satisfying configurations or local motion. However, the piece-wise linear approximation of the manifold breaks down when the manifold becomes highly curved, as tangent movement rapidly drifts away from the surface of the manifold. Additionally, the two methods also break down near singularity points, due to the Jacobian losing rank and no longer maintaining a surjective mapping to the ambient configuration space. By comparison, projection is easier to implement and captures the structure of the constraint function within the planning process, which is the most widely implemented methodology. The projection process is introduced in Section 3.5.

Constrained Bi-direction Rapidly-Exploring Random (CBiRRT) [11,18] is a typical projection-based constrained motion planning method. Based on the bidirectional extension of BiRRT, the configurations which are sampled uniformly in ambient space are projected on constraint manifold step by step, and the continuous path satisfying the task constraints is obtained. Because the process of sampling and projection is online, the method lacks the prior knowledge of constraint manifold, which leads to its blind expansion. The failure of the projection process and the exploration of invalid regions in the C-space greatly limits the efficiency and the success rate of the planning algorithm within the allowed time.

In recent years, many successful applications of machine learning methods in the field of robot motion planning show that effective utilization of prior knowledge of the C-space can greatly improve the efficiency of various planners [19–22]. Therefore, this paper obtains the approximate model of high dimensional constraint manifolds by machine learning methods and selects the ideal extension direction based on the approximate metric for random tree. This process avoids the exploration of the

invalid region in C-space, thus greatly improving the efficiency of the constrained motion planning algorithm. The proposed framework, which is developed from CBiRRT, and CBiRRT itself were applied to the motion planning of single-arm robot and dual-arm robot under task constraints.

The main contributions of this paper is to learn an approximate metric for constraint manifolds. The paper is organized as follows: Section 2 introduces the existing work related to the framework of the article. In Section 3, the construction and metric learning methods of approximate model for constraint manifold are introduced, and a motion planning framework under constraints is proposed. In Section 4, AG-CBiRRT and CBiRRT are applied to deal with the task constraints, including keeping a cup upright, dual-arm cooperation and the closed passive chains. The results and analysis of many performance indicators including the planning time, the success rate, the path length and the number of extended nodes are provided. In Section 5, we make conclusions and provide directions for future work.

2. Background and Related Work

In this paper, the problem is to find a collision-free path in configuration space, satisfying the task constraints of the end effectors. Because of the existence of hard kinematic constraints, the valid configuration set is restricted to a manifold embedded in C-space. Because the constraint manifold has zero measure with respect to the ambient space metric, the probability of generating valid configurations by direct sampling method is very low. A commonly used method to solve this problem is a projection operator, which can be realized by a Gauss–Newton process based on Jacobian pseudo inverse $J(q)^+$ [23].

The projection operator is used in the process of sampling and local extension. This process allows configurations to be sampled arbitrarily in the ambient space where the manifold is located. Because a projection is easy to implement, it is utilized to solve the problem of tracking the trajectory of industrial robots [24]. Berenson et al. [18] proved that, when the planning time approaches infinity, the configuration set that satisfies the constraints will be fully covered by the projection sampling process. This means that the constrained motion planning method has probabilistic completeness, which is similar to the RRT-like planners. The probabilistic completeness of sampling-based motion planning algorithms means that, if there is a path solution between the starting point and the target point, a path solution must be found as long as the planning time is long enough. Yakey et al. [12] proposed using a projection operator to enable sampling-based motion planners such as RRT and PRM to expand under constraints. They used the framework to solve the motion planning problem of closed kinematic chains. Stilman [13] defined a generic task constraints description method, and presented three projection methods based on the Jacobian pseudo inverse: Tangent Space Sampling (TS), First-Order Retraction (FR) and the Random Gradient Descent (RGD). In recent years, some humanoid robots such as HRP2 [25] and Humanoid Path Planner System [26] also adopt projection to meet task constraints.

Although the Projection-based planning techniques are easy to implement and have probabilistic completeness, the unknown structure of the constraint manifold will lead to many extension steps towards invalid regions in the C-space. To solve this problem, off-line sampling and machine learning methods are used to obtain prior knowledge of manifolds. Off-line sampling means obtaining a configuration set that satisfies constraints by precalculation. Kagami [27] and Burget [28] et al. utilized this method to realize the whole body motion planning of a humanoid robot under dynamic equilibrium constraints. Sucan et al. [29] connected the configurations sampled off-line to generate an approximate graph of the manifold, and then used a variety of sampling-based motion planners to search on the graph directly. However, because of the limited accuracy of the approximate graph, it is difficult to ensure that the local expansion process satisfies constraints. Roy N [30] and Phillips M [31] et al. proposed that the experience graph (E-graph) can be learned from constrained movements (such as opening doors, or pulling drawers) which are demonstrated by users. The E-graph can guide the planner to perform new planning tasks and achieve remarkable effects in multi-DOF robots' motion

planning such as PR2. However, the manifold prior model mentioned above will need to be resampled and relearned when the environment changes. No researcher uses an adapted manifold metric to guide the constrained motion planning.

3. Methods

We introduce the process of constructing an approximate graph of constraint manifolds by off-line sampling, and utilize an approximate model to learn the approximate metric on the manifold. This metric is combined with the projection operator to achieve fast and high-quality motion planning under task constraints.

3.1. Problem Statement

The configuration space of the robot is defined by Q. A path in the space is defined by $\tau : [0,1] \rightarrow Q$. We consider constraints evaluated as a function of a configuration $q \in Q$ in τ. The location of q in τ decides which constraints are active at the configuration. Thus, a constraint is defined as the pair $\{C(q), s\}$, where $C(q) \in \mathbb{R} \geq 0$ is the constraint-evaluation function to decide if the constraint is satisfied at q, and $s \subseteq [0,1]$ is the domain of this constraint, i.e., where in the path τ the constraint is active. We assume that τ satisfies the given constraint when $C(q) = 0, \forall q \in \tau(s)$. Each constraint defined in this way implicitly defines a manifold in Q where $\tau(s)$ is allowed to exist. Configurations that satisfy the constraint form the manifold $\mathcal{M}_C \subseteq Q$, which is defined as:

$$\mathcal{M}_C = \{q \in Q : C(q) = 0\} \tag{1}$$

For τ to satisfy a constraint, all the elements of $\tau(s)$ must lie within $\mathcal{M}_C$. If there exists a configuration q not on $\mathcal{M}_C$, q is said to violate the constraint. In general, we can define any number of constrains for a given task, each with their own domain. Let a set of n constraint-evaluation functions be $\mathcal{C}$ and the set of domains corresponding to those functions be S. Then, the constrained motion planning problem is defined as:

$$\begin{aligned} \text{find } \tau : q \in \mathcal{M}_{C_i} \quad & \forall q \in \tau(S_i) \\ & \forall i \in \{1 \ldots n\} \end{aligned} \tag{2}$$

When the end effector of the three-DOF manipulator in Figure 1a is constrained to be on a straight line, the path satisfying the constraint must be located on the manifold in Figure 1b. Note that, if two or more constraints' domains overlap, the way-points of the path τ need to lie within two or more constraint manifolds.

3.2. Approximating Constraint Manifolds

Given a specific task constraint, we can construct an approximate graph of the constraint manifold. Task constraints and environment constraints (e.g., caused by geometry of the robots) are decoupled, which means that collision avoidance constraints are not considered in the process of constructing a manifold approximate graph. The form of the approximate graph is $\mathrm{G} = (V, E, D)$, where V is the set of nodes on the approximate graph corresponding to the configurations satisfying task constraint which are sampled off-line, and E is the edges on the graph, which refer to the node connections on the manifold. D is the distance matrix of nodes on the approximate graph. The construction process of manifold approximate graph is shown in Algorithm 1.

V can be sampled off-line by robot inverse kinematic calculation, projection [11–13], tangent space [14] or atlas [15,16] methods. Samples are drawn to cover the area of interest in the satisfying subset defined by the constraint and placed within V. Then, the KNN [32] algorithm is used to connect each node to the k nearest neighbors in V to generate edges on the approximate graph, as shown in Lines 2–7 of Algorithm 1. KNN is a lazy learning method for data classification by measuring the

distance between samples in feature space. KNN is used to construct the relationship between adjacent samples to generate undirected graph. In this process, the k nearest neighbors of each sample need to be retrieved by Kdtree [33]. As shown in Figure 2a, the torus is a two-dimensional manifold in the three-dimensional space. An approximate graph covering the entire manifold can be obtained by uniform sampling and adjacent samples connection process on the torus.

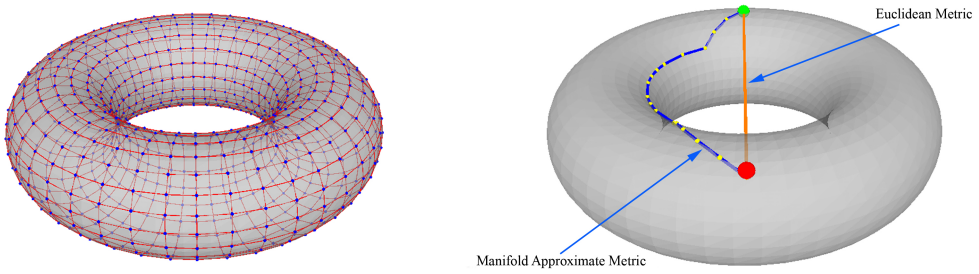
Algorithm 1: Construct approximate graph.

Input: Task constraint $C(q)$;
Output: Manifold approximate graph $\mathrm{G}(V, E, D)$;
$V \leftarrow$RobotIK$(C(q))$ or Projection$(C(q))$ or Atlas$(C(q))$ or TangentSpace$(C(q))$;
for $i = 0$ *to* Size(V) **do**
  $V[i].list \leftarrow$ K-NearestNeighbors$(V[i], V)$;
  for $j = 0$ *to* k **do**
    $E \leftarrow$AddEdge$(V[i].list[j], V[i])$;
  end
end
for $i = 0$ *to* Size(V) **do**
  for $j = 0$ *to* Size(V) **do**
    $D[i][j] \leftarrow$Dijkstra$(E, V[i], V[j])$;
  end
end
return $\mathrm{G}(V, E, D)$;

The above undirected graph can be used to realize the estimation of distance metric on manifold. Distance metric plays an important role in motion planning. For example, the asymptotically optimal motion planning methods, such as RRT* [7] and RRTX [34], use the distance metric to calculate the current cost and heuristic value of the path, which realizes the continuous optimization of the path. The geodesic distance on constraint manifolds cannot be directly calculated in practice, but it can be approximated piecewise linearly by graph distance [35]. The problem of calculating the shortest arc length between two points on a manifold can be simplified to the problem of calculating the shortest path between two points on an approximate graph. The Euclidean distance between the join points in the approximate graph is labeled as the weight of the edges to generate an undirected weight graph. Since the shortest path length on the undirected weighted graph satisfies the properties of a metric, namely being non-negative, symmetric and triangular inequalities, it can be used as an approximate metric of the manifold.

The remaining problem is constructing the distance matrix D between all nodes in the undirected weight graph, which can be realized by repeatedly calling the Dijkstra algorithm [36]. The Dijkstra algorithm can calculate the shortest distance between a source node and all other nodes on the weight graph. As shown in Figure 2b, the approximate geodesic distance between two points on the torus can be obtained by searching the approximate graph in Figure 2a with Dijkstra algorithm.

Constructing distance matrix between all nodes on the manifold approximate graph is the process of metric learning for the constraint manifolds. It is noteworthy that the distance matrix is a real symmetric matrix with a size of $|V| \times |V|$. When the dimension of planning problems are relatively high, the number of samples required is large, resulting in large memory occupied required by the distance matrix. Thus, in practice, we use a constant lookup data structure to read the distance matrix online.



(**a**) Manifold approximate graph (**b**) Manifold approximate matric

Figure 2. The torus is a two-dimensional manifold in three dimensions. The blue points in (**a**) are the samples that are uniformly sampled on the torus surface. Each point is connected to the five nearest points (red line), thus forming an approximate graph of the manifold. In (**b**), the green and red points are the points needed to perform metric calculation. The blue line is the shortest path that the Dijkstra algorithm searches on the graph, and the yellow points is the way-points on the shortest path. The length of the path can be used as an approximate metric between the two points on torus surface. The length of the orange line is the Euclidean metric between these two points.

3.3. Planning on the Constraint Manifold

In this part, the manifold prior knowledge will be used for constrained motion planning. Similar to CBiRRT [11], the method proposed in this paper implements the exploration of configuration space by growing two trees from the starting point and the goal point, respectively. However, the difference is that the method is based on the approximate graph and manifold metric to select the local optimal expansion direction, rather than by random sampling expansion. Since this algorithm is based on CBiRRT and utilizes the manifold approximate graph for motion planning, we named the algorithm the Approximate Graph CBiRRT (AG-CBiRRT). As shown in Algorithm 2, the input includes the starting point q_{start}, goal point q_{goal} of the planning problem and the approximate graph G of the constraint manifold. The output is the path τ that satisfies the task constraints.

Algorithm 2: AG-CBiRRT.

Input: q_{start}, q_{goal}, G;
Output: τ;
$T_a.init(q_{start}), T_b.init(q_{goal})$;
while TimeRemaining() **do**
  $q_{rand} \leftarrow$ RandomConfig();
  $q^a_{near} \leftarrow$ NearestNeighbor(T_a, q_{rand});
  $q_{rand} \leftarrow$SelectNeighbor$(q^a_{near}, q_{rand}, \text{G}, T_a)$;
  $ExtendResult, q^a_{reached} \leftarrow$ ConstraintExtend$(T_a, q^a_{near}, q_{rand})$;
  if $ExtendResult \neq$TRAPPED **then**
    $q^b_{near} \leftarrow$ NearestNeighbor$(T_b, q^a_{reached})$;
    $ExtendResult, q^b_{reached} \leftarrow$ ConstraintExtend$(T_b, q^b_{near}, q^a_{reached})$;
  end
  if $q^a_{reached} = q^b_{reached}$ **then**
    $\tau \leftarrow$ ExtractPath$(T_a, q^a_{reached}, T_b, q^b_{reached})$;
    return τ;
  end
  else
    Swap(T_a, T_b);
  end
end

As shown on Lines 3–5 of Algorithm 2, the configurations obtained by random sampling are not set as local expansion targets directly. Algorithm 2 searches the nearest neighbors q_{near}^{a} on the current tree T_a, and uses SelectNeighbor function, which is explained in Algorithm 3, to select the optimal extension direction for the q_{near}^{a}.

As shown on Line 2 of Algorithm 3, the function NearestNeighbor is utilized to search the nearest neighbor of the other tree T_{other}'s root node, and set it as the global extension target q_g. Then, the function K-NearestNeighbors is utilized to retrieve the k nearest neighbors of q_{near} on the approximate graph, which are stored in the data structure $q_{near}.list$. In this way, k candidate extension targets are obtained. The geodesic distance between each candidate nearest neighbor and the global extension target q_g is difficult to calculate in practice, but, as mentioned above, it can be estimated by the metric GraphMetric(a, b) on the manifold approximate graph. Since the distance matrix of the approximate graph is obtained by off-line computing, the GraphMetric(a, b) can be read directly so that it will not generate additional computing overhead. Since the manifold $\mathcal{M}_C$ has the property of its ambient Euclidean space Q in local, the geodesic distance between candidate neighbors and q_{near} can be estimated by Euclidean metric function EuclideanMetric(a, b). In this way, as shown on Line 11 of Algorithm 3, the path cost between each candidate neighbor and q_g can be calculated. The candidate nearest neighbor with the lowest path cost will be chosen as the target of this extension. As shown in Figure 3, through this process, it can select the most promising local extension target for q_{near}.

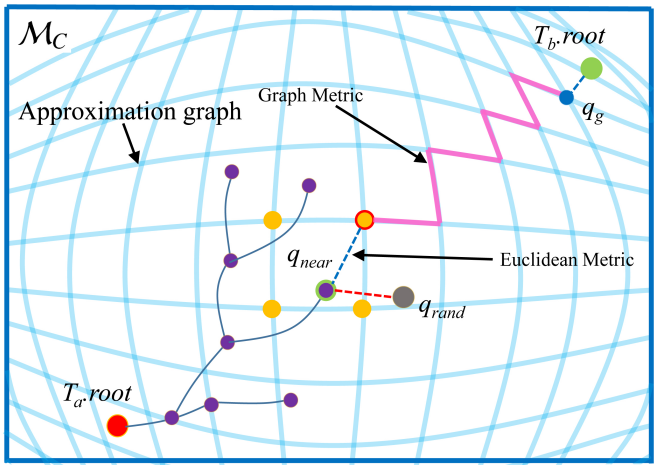


Figure 3. The diagram of selecting the nearest candidate neighbors. The grey point q_{rand} is a random sampling configuration in the C-space (not on the constraint manifold). The purple point with green contour q_{near} is the nearest neighbor of q_{rand} on T_a. The yellow point is one candidate neighbor of q_{near} on the manifold approximate graph. Through the Euclidean metric (blue dotted line) between the candidate nearest neighbors and q_{near}, and the graph metric (pink line) between the candidate nearest neighbors and the q_g, the estimated cost of each candidate nodes can be calculated. The candidate neighbor with the lowest cost (yellow point with red contour) is chosen as the extension target of q_{near}.

Note that the manifold approximate graph obtained by the previous learning process only considers task constraints, and does not consider the collision avoidance constraint of environment. Therefore, the expansion failure caused by environmental obstacles may cause the planning algorithm to fall into dead circulation (an infeasible candidate neighbor is repeatedly selected). Therefore, it is necessary to ensure that:

- Each node on the search tree only performs once k nearest neighbors searching, as shown on Line 3 of Algorithm 3.
- Each candidate nearest neighbor can only be selected once (but it can be selected as other nodes' candidate neighbor). As shown on Line 16 of Algorithm 3, when the candidate neighbor is selected, it will be erased from the data structure $q_{near}.list$.

- If q_{near}'s k nearest neighbors have been searched but the $q_{near}.list$ is empty (Line 7 of Algorithm 3), which indicates that all the neighbors have been selected as q_{rand} and erased from $q_{near}.list$, the q_{rand} sampled in configuration space will be returned without change.

Algorithm 3: SelectNeighbor(q_{near}, q_{rand}, G, T;).

```
   Output: q_rand;
1  minimumcost=MAX;
2  q_g ← NearestNeighbor(G, T_other.root);
3  if !q_near.searched then
4  |  q_near.list ← K − NearestNeighbors(q_near, G);
5  |  q_near.searched ←True;
6  end
7  if q_near.list = empty then
8  |  return q_rand;
9  end
10 for i = 0 to Size(q_near.list) do
11 |  cost ← GraphMetric(q_near.list[i], q_g)+EuclideanMetric(q_near.list[i], q_near);
12 |  if minimumcost > cost then
13 |  |  minimumcost ← cost;
14 |  |  q_rand ← q_near.list[i];
15 |  end
16 |  Erase(q_near.list, q_rand);
17 end
18 return q_rand;
```

After completing the selection of candidate neighbors, the ConstraintExtend function is used to implement local extensions from q_{near} to target q_{rand} under constraints. If the extension is successful, the ConstraintExtend function is called again to extend the other tree T_b to the $q^a_{reached}$ which is reached last time. If $q^a_{reached} = q^b_{reached}$, it means that two search trees are connected and the planning task is completed. Otherwise, the trees will be swapped and expanded until the allowed planning time is reached.

3.4. Local Extension under Constraint

Although the selection of an optimal candidate neighbor on the manifold approximate graph can make most of the extension on the manifold, the Projection operation is still necessary. The main reasons are as follows:

- The number of samples is limited, which will lead to the error between an approximate graph and the constraint manifold.
- When the $q_{near}.list$ is emptied, q_{rand} will be returned as local extension target, but the configuration does not satisfy the task constraints, as shown on Lines 7–9 of Algorithm 3. Therefore, to ensure that all way-points on the path satisfy the task constraints, the ConstrainedExtend function is used to implement local extension under constraints.

As shown in Algorithm 4, The ConstrainedExtend function works by iteratively moving from q_{near} to the configuration q_{target} with step size $\triangle q_{step}$. q_{target} represents the target of local extension under task constraints. During each move, the ConstrainConfig function is used to project the new configuration q_s onto the constraint manifold, so that q_s satisfies the task constraint. The ConstrainConfig function is defined by specific problems and is discussed in Section 4. The CollisionFree function checks whether the interval between the two configurations is feasible by

calling the collision query module. After each move, we whether the new configuration q_s reaches q_{target} is checked. There are three cases of local extension results: REACHED represents the local extension target q_{target} is reached; ADVANCED indicates that it moves toward the target, but hasn't arrived; and TRAPPED indicates that there is no successful movement. Because the planner may fail to reach the local expansion target q_{target}, we need to keep track of where the planner actually reached in $q_{reached}$, which could be used as the local expansion target of next round as shown on Line 9 of Algorithm 2.

Algorithm 4: ConstrainedExtend(T, q_{near}, q_{target}).

Output: $ExtendResult, q_{reached}$;
1 $q_s \leftarrow q_{near}$; $q_s^{old} \leftarrow q_{near}$;
2 **while** True **do**
3 **if** $q_{target} = q_s$ **then**
4 $q_{reached} \leftarrow q_{target}$;
5 **return** REACHED, $q_{reached}$;
6 **end**
7 **if** $|q_{target} - q_s| > |q_s^{old} - q_{target}|$ **then**
8 $q_{reached} \leftarrow q_s^{old}$;
9 **return** ($q_{reached} = q_{near}$?TRAPPED;ADVANCED), $q_{reached}$;
10 **end**
11 $q_s \leftarrow q_s + \min(\triangle q_{step}, |q_{target} - q_s|) \frac{(q_{target} - q_s)}{|q_{target} - q_s|}$;
12 $q_s \leftarrow$ ConstrainConfig(q_s^{old}, q_s);
13 **if** $q_s \neq$ NULL ***and*** CollisionFree(q_s^{old}, q_s) **then**
14 T.AddVertex(q_s);
15 T.AddEdge(q_s^{old}, q_s);
16 **end**
17 **else**
18 $q_{reached} \leftarrow q_s^{old}$;
19 **return** ($q_{reached} = q_{near}$?TRAPPED;ADVANCED), $q_{reached}$;
20 **end**
21 **end**

3.5. Projection Operation to Satisfy Constraints

In this section, the method of satisfying task constraints by projection technique is briefly introduced. CBiRRT [11] extends the Bi-directional RRT (BiRRT) algorithm by using projection techniques to explore the configuration space manifolds that correspond to constraints and to find bridges between them. The projection technique used in this article is similar to that of CBiRRT.

Firstly, $\mathbf{T}_c^0$ is defined as the relative transformation matrix of the task constraints relative to the world coordinate system. $\mathbf{T}_{obj}^0$ is the relative transformation matrix between the end effector and the world coordinate system, which can be calculated by the positive kinematics. Constraints in Cartesian space are defined according to the allowable tolerance between $\mathbf{T}_{obj}^0$ and $\mathbf{T}_c^0$, as shown in Equation (3).

$$\mathbf{C} = \begin{bmatrix} x_{min} & x_{max} \\ y_{min} & y_{max} \\ z_{min} & z_{max} \\ \psi_{min} & \psi_{max} \\ \theta_{min} & \theta_{max} \\ \phi_{min} & \phi_{max} \end{bmatrix} \tag{3}$$

The first three lines of $\mathbf{C}$ provide upper and lower boundaries for the translation transformations along X, Y, and Z axes, respectively, and the last three lines of $\mathbf{C}$ provide upper and lower bounds for the rotation transformations around these three axes.

Given a configuration q_s, the DisplacementFromConstraint$(\mathbf{C}, \mathbf{T}_c^0, q_s)$ function is defined as follows:

First, we use forward kinematics at q_s to get $\mathbf{T}_{obj}^0$. The relative coordinate transformation between the end-effector and the task constraint can be calculated by Equation (4):

$$\mathbf{T}_{obj}^c = (\mathbf{T}_c^0)^{-1}\mathbf{T}_{obj}^0 \tag{4}$$

Then, $\mathbf{T}_{obj}^c$ can be converted to a six-dimensional vector d^c, including X, Y, Z, Roll, Pitch and Yaw, which represents displacement vector in Cartesian space:

$$d^c = \begin{bmatrix} \mathbf{t}_{obj}^c \\ \arctan2(\mathbf{R}_{obj32}^c, \mathbf{R}_{obj33}^c) \\ -\arcsin(\mathbf{R}_{obj31}^c) \\ \arctan2(\mathbf{R}_{obj21}^c, \mathbf{R}_{obj11}^c) \end{bmatrix} \tag{5}$$

where $\mathbf{t}_{obj}^c$is the 3×1 translation vector in $\mathbf{T}_{obj}^c$, and $\mathbf{R}_{obj}^c$ is the 3×3 rotation matrix in $\mathbf{T}_{obj}^c$.

Finally, by comparing with the constraint bounds in Equation (3), we can get the displacement from the constraint in the Cartesian space $\triangle\mathbf{x}_i$:

$$\triangle \mathbf{x}_i = \begin{cases} d_i^c - \mathbf{C}_{imax}, \text{ if } d_i^c > \mathbf{C}_{imax} \\ d_i^c - \mathbf{C}_{imin}, \text{ if } d_i^c < \mathbf{C}_{imin} \\ 0, \text{otherwise} \end{cases} \quad i = x, y, z, \psi, \theta, \phi; \tag{6}$$

To satisfy the constraint, we use a gradient descent projection method based on the Jacobian pseudo inverse, which is consistent with the method used in CBiRRT (as shown in Algorithm 5).

Algorithm 5: ProjectConfig$(q_s^{old}, q_s, \mathbf{C}, \mathbf{T}_c^0)$.

```
   Output: q_s
1  while true do
2      △x ← DisplacementFromConstraint(C, T_c^0, q_s);
3      if ‖△x‖ < ε then
4          return q_s;
5      end
6      J ← GetJacobian(q_s);
7      △q_error ← J^T(JJ^T)^{-1} △x
8      q_s ← (q_s − △q_error);
9      if |q_s − q_s^old| > 2 △ q_step or OutsideJointLimit(q_s) then
10         return NULL;
11     end
12 end
```

Through the ProjectConfig function, the configuration q_s which satisfies the task constraints of the end effector can be obtained. Due to the manifold approximate graph based local extension, most of the way-points are satisfied with the task constraint, and the times of Projection operations is reduced greatly (as shown on Lines 3–5 of Algorithm 5).

4. Results

In this section, the simulation and experiment results of the proposed method in the constrained motion planning problem of keeping a cup upright, dual-arm cooperation and passive chains are shown. All of the above algorithms were implemented in the framework of the Open Motion Planning Library (OMPL 1.3.1) [37]. All experiments were implemented on an Intel Core i5-4590 3.3GHz personal computer. ROS Indigo and Gazebo were used to build the simulation platform, and the Flexible Collision Library (FCL) [38] was used for collision query. In all experiments, search step length was $\triangle q_{step} = 0.1$, and the constraint tolerance was $\epsilon = 0.1$.

4.1. Constraint Motion Planning for Keeping a Cup Upright

For single-arm robots, the end effector holding a cup and keeping it upright is a typical task constraint. Firstly, we determined the constraint boundary **B** of the task in the form of Equation (3):

$$\mathbf{B} = \begin{bmatrix} -\infty & +\infty \\ -\infty & +\infty \\ -\infty & +\infty \\ 0 & 0 \\ 0 & 0 \\ -\infty & +\infty \end{bmatrix} \tag{7}$$

The two degrees of freedom under Cartesian space are restricted, so the feasible configuration of a 6-DOF manipulator under this task is constrained on a continuous four-dimensional manifold. In this task, the constraint's coordinate system $\mathbf{T}_c^0$ can be set to any fixed point in the space. For convenience, we set it as the base coordinate system of the manipulator. Next, we designed the ConstrainConfig function for the task, as shown in Algorithm 6.

Algorithm 6: ConstrainConfig(q_s^{old}, q_s).

Output: q_s

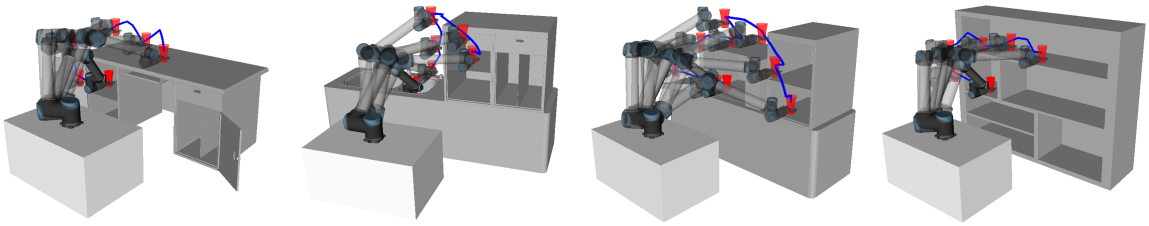
1 $\mathbf{C} = \mathbf{B}$;
2 $\mathbf{T}_c^0 =$ BaseofManipulator();
3 $q_s \leftarrow$ ProjectConfig($q_s^{old}, q_s, \mathbf{C}, \mathbf{T}_c^0$);
4 **return** q_s;

By traversing the workspace of the robot and the inverse kinematics, a configuration satisfying the task constraint could be obtained. The approximate graph $G(V, E, D)$ of the four-dimensional manifold was constructed using 4183 samples in 49.1 min, and occupied a memory of 189.2 Mb. As shown in Figure 4, a 6-DOF UR10 manipulator was used to build the simulation platform. AG-CBiRRT, which is proposed in this paper, and CBiRRT were used for motion planning under constraints in four scenarios, respectively. Note that, since our method decouples the task constraints from collision avoidance constraints, the obtained manifold metric is applicable in all four scenarios and does not need to be relearned.

As shown in Table 1, the performance comparison between CBiRRT and AG-CBiRRT algorithm under four planning scenarios could be obtained through multiple tests. The success rate refers to the probability that a feasible path could be searched within the 10 s allowed planning time. The path length refers to the sum of Euclidean distances between adjacent path points on the final path. The number of nodes refers to the number of feasible nodes on the tree when planning was successful.

To show the algorithm's effect more intuitively, we present box-plots of the planning time and the path length. As shown in Figure 5a, since the extension of AG-CBiRRT was carried out on the manifold approximate graph, many projection operations and the exploration of the invalid regions

in C-space were avoided. The planning time was controlled within 1 s, and the success rate in each planning scene was up to 100%. In most of the scenes, the planning efficiency was improved by an order of magnitude. As shown in Figure 5b, the method proposed in this paper used the approximate graph to select the lowest cost expansion direction in each step. Compared with the CBiRRT, the path length was significantly shortened.

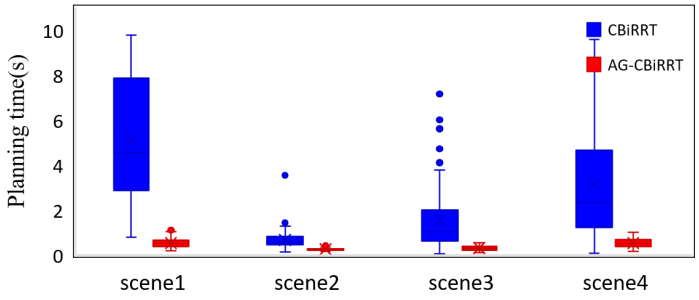


(**a**) Scene 1: Desk (**b**) Scene 2: Kitchen (**c**) Scene 3: Microwave (**d**) Scene 4: Cabinet

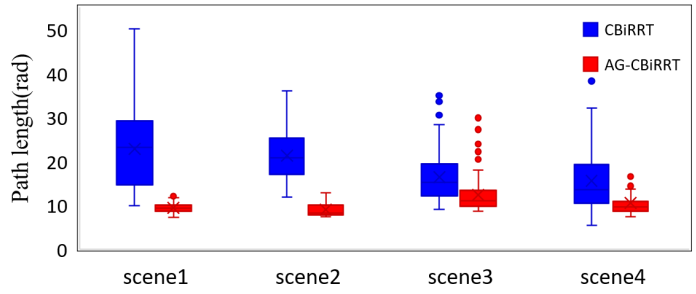
Figure 4. The moving trajectory of the manipulator in four scenes. The end effector is required to always keep the cup upright during the movement. To clearly display the environment model, the robot states on the path are not all displayed.

Table 1. Single manipulator motion planning under task constraints.

	CBiRRT				AG-CBiRRT			
	Scene 1	**Scene 2**	**Scene 3**	**Scene 4**	**Scene 1**	**Scene 2**	**Scene 3**	**Scene 4**
Success Rate (%)	54.2	100	100	89.6	100	100	100	100
Average Planning Time (s)	5.11	1.25	2.87	3.13	0.54	0.25	0.31	0.53
Average Path Length (rad)	22.76	39.30	30.28	15.49	9.35	8.90	12.25	10.46
Average Nodes Number	1738	646	1142	1155	369	144	172	293



(**a**) Planning time



(**b**) Path length

Figure 5. Box-plots of motion planning performance.

4.2. Constraint Motion Planning for Dual-Arm Cooperation

A dual-arm robot forms a kinematic closed chain when it manipulates an object, which is a common task constraint. For the task of holding an object and maintaining its levelness with the end effectors of the dual-arm robot, the configurations that satisfy the constraint constitute a continuous four-dimensional manifold in a $2d$-dimensional space (d is the single manipulator's DOF). For this task, similar to the case of the single-arm robot, the constraints' coordinate system $\mathbf{T}_c^0$ could be set to any fixed point in work space. Since the end effectors of the two arms were connected by the manipulated objects, the coordinate system $\mathbf{T}_{right}^0$ of the right end effector had the following relationship with the left end effector's coordinate system. $\mathbf{T}_{left}^0$:

$$\mathbf{T}_{left}^0 = \mathbf{T}_{obj}\mathbf{T}_{right}^0 \tag{8}$$

where $\mathbf{T}_{obj}$ is the coordinate transformation between the two end effectors, which is related to the size and shape of the object.

The motion constraints of the manipulated object were decomposed into two end effectors' constraints:

$$\mathbf{B}_{right} = \begin{bmatrix} -\infty & +\infty \\ -\infty & +\infty \\ -\infty & +\infty \\ 0 & 0 \\ 0 & 0 \\ -\infty & +\infty \end{bmatrix} \qquad \mathbf{B}_{left} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{9}$$

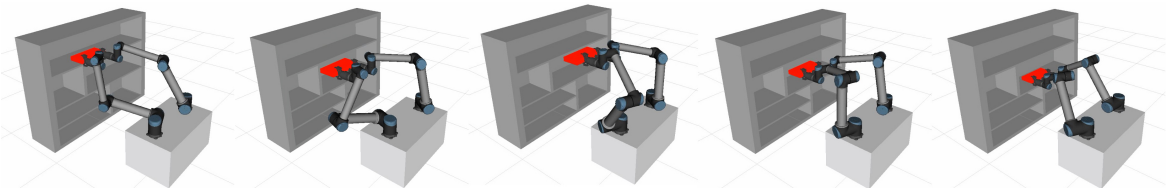
$\mathbf{T}_{right}^0$ needs to satisfy the constraint of keeping the object level. $\mathbf{T}_{left}^0$ could be solved according to Equation (8), and its projection operation was equivalent to solving inverse kinematics by numerical iteration. Therefore, the task constraint of the dual-arm robot needed to be satisfied by two projection operations (as shown in Algorithm 7).

Algorithm 7: ConstrainConfig(q_s^{old}, q_s).

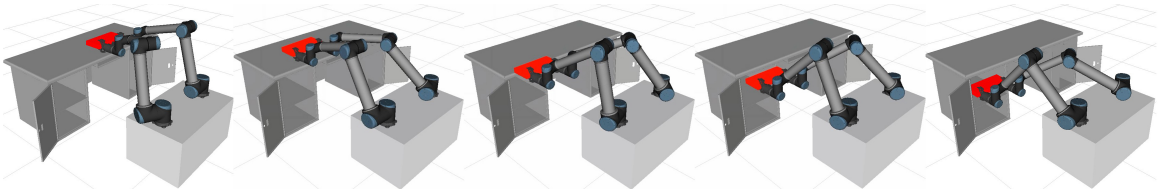
Output: q_s

$\mathbf{T}_c^0 = \text{BaseofRightManipulator}()$;
$q_s.right \leftarrow \text{ProjectConfig}(q_s^{old}.right, q_s.right, \mathbf{B}_{right}, \mathbf{T}_c^0)$;
$\mathbf{T}_{right}^0 = \text{ForwardKinematic}(q_s.right)$;
$\mathbf{T}_{left}^0 = \mathbf{T}_{obj}\mathbf{T}_{right}^0$;
$q_s.left \leftarrow \text{ProjectConfig}(q_s^{old}.left, q_s.left, \mathbf{B}_{left}, \mathbf{T}_{left}^0)$;
if $q_s.right = \text{NULL}$ ***or*** $q_s.left = \text{NULL}$ **then**
    return NULL;
end
$q_s \leftarrow (q_s.right, q_s.left)$;
return q_s;

Subsequently, AG-CBiRRT and CBiRRT were used for the constrained motion planning tasks of dual-arm robots, respectively, as shown in Figure 6. Two 6-DOF UR10 robots formed a dual-arm robot simulation platform, which aimed to maintain the levelness of the object in the process of holding and transferring the object. The time limit for the motion planning was set to 200 s. The approximate graph of the constraint manifold was constructed using 4286 samples in 53.6 min, and occupied a memory of 204.9 Mb.



(**a**) Scene 1



(**b**) Scene 2

Figure 6. The snapshots of a dual-arm robot performing constrained motion tasks. The object had to be kept level in the process of transferring.

Table 2 shows the performance comparison of the two motion planning algorithms. In Scene 1, since the AG-CBiRRT algorithm used the manifold metric, the planning success rate significantly improved compared with the CBiRRT. In addition, the time required to complete the planning task was reduced by two orders of magnitude, and the path length was greatly shortened. In Scene 2, because of the smaller shelf space in the feasible space (as shown in Figure 6b), CBiRRT's success rate within limited time was very low. However, AG-CBiRRT still had 100% planning success rate in this task.

Table 2. Dual-arm robot motion planning under task constraint.

	CBiRRT		**AG-CBiRRT**	
	Scene 1	**Scene 2**	**Scene 1**	**Scene 2**
Success Rate (%)	66.3	4.0	100	100
Average Planning Time (s)	2.97	144	0.043	17.3
Average Path Length (rad)	7.82	16.79	5.65	12.15
Average Nodes Number	87	1,121	42	715

4.3. Constraint Manipulation Planning for Passive Chains

The task in this problem was for the robot to pull a drawer or open a door. In these processes, the robot and the operated object formed a closed kinematics chain. The operated object was assumed to be completely passive and the kinematics were assumed to be known.

In the task of pulling a drawer, the end effector was constrained on a line that is parallel to the opening direction. For convenience, we set the constraint's coordinate system $\mathbf{T}_c^0$ at the midpoint of the sliding axis of the drawer. To ensure a smooth opening, the orientation of the end effector was fixed during the motion. We could determine the constraint boundary $\mathbf{B_1}$ of the pulling task in the form of Equation (3).

In the task of opening a door, the end effector was constrained on an arc centered on the door hinge. The constraint's coordinate system $\mathbf{T}_c^0$ was set on the door hinge. Thus, the end effector can only rotate around $\mathbf{T}_c^0$'s Z axis and the constraint boundary $\mathbf{B_2}$ can be determined:

$$\mathbf{B_1} = \begin{bmatrix} 0 & 0 \\ -\infty & +\infty \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \mathbf{B_2} = \begin{bmatrix} r\cos(P) & r\cos(P) \\ -r\sin(P) & -r\sin(P) \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\infty & \infty \end{bmatrix} \tag{10}$$

where P is the Pitch of end effector relative to the constraint's coordinate system $\mathbf{T}_c^0$, and r is the door's revolving radius.

Except for $\mathbf{T}_c^0$'s position, the ConstrainConfig function for these two tasks was exactly the same as Algorithm 6.

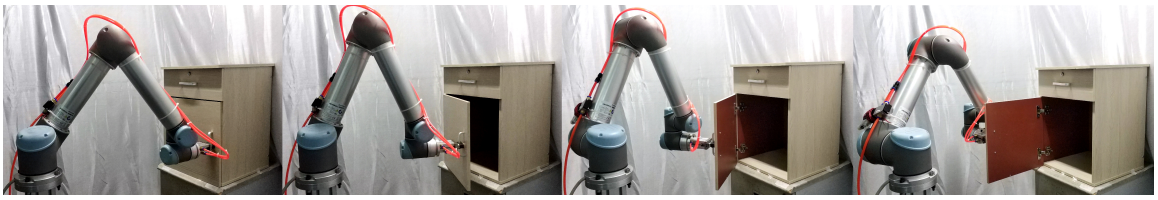
Then, we prepared the manifold approximate graph for the two tasks, respectively, according to Algorithm 1, and the graph information is shown in Table 3.

Table 3. The information of approximate graphs for passive chain constraint manifolds.

	Pulling the Drawer	**Opening the Door**
Node Number	512	422
Construction Time (s)	2.97	2.51
Occupied Memory (Mb)	8.75	5.95

Subsequently, AG-CBiRRT and CBiRRT were used for constrained motion planning tasks of passive chains, respectively, as shown in Figure 7. The time limit for the motion planning was set to 100 s.

(**a**) Pulling the drawer



(**b**) Opening the door

Figure 7. Snapshots of the robot performing constrained manipulation on passive chains.

Table 4 shows the performance comparison of the two motion planning algorithms for passive chains. In pulling a drawer task, since the AG-CBiRRT algorithm makes use of the manifold approximate metric, the planning success rate significantly improved compared with the CBiRRT algorithm. In addition, the time required to complete the planning task was reduced by one order of magnitude, and the path length was shorter. In opening a door task, CBiRRT could not complete the planning task. However, AG-CBiRRT still had 100% success rate within limited time.

Table 4. Robot performing constrained manipulation on passive chains.

	CBiRRT		AG-CBiRRT	
	Drawer	**Door**	**Drawer**	**Door**
Success Rate (%)	37.5	0	100	100
Average Planning Time (s)	3.777	–	0.236	2.112
Average Path Length (rad)	1.747	–	1.076	1.536
Average Nodes Number	211	–	8	26

5. Conclusions

In this paper, a motion planning method based on manifold metric learning is proposed to solve the problem of robot motion planning under task constraints. The off-line sampling method is used to obtain the configurations satisfying the task constraints, and the KNN algorithm is used to construct a graph which can approximate the constraint manifold. Then, the shortest path length between the nodes on the approximate graph is calculated using the Dijkstra algorithm, and the distance matrix is constructed off-line to realize the metric learning of the constraint manifolds. During on-line planning, the approximate graph and metric of the manifold are used to select the most promising direction in the local expansion, thus reducing the exploration of the invalid regions in C-space. Compared with CBiRRT, the proposed method is faster and has lower path length in experiments of keeping a cup upright, the dual-arm cooperation and the closed passive chains. In the dual-arm robot's constrained motion planning tasks, the planning efficiency is improved by 1–2 orders of magnitude. Therefore, the method proposed in this paper has the ability to perform various constrained motion planning tasks.

In the future, under the premise of satisfying the task constraints and environmental constraints, we will improve the algorithm to shorten the path length and make the path smoother. Besides, we plan to extend the approximate graph method to all RRT-like motion planning algorithms within the framework of OMPL.

Author Contributions: F.Z. and Y.L. made substantial contributions to the original ideas and designed the experiments. W.G., M.L. and X.W. developed the simulation platform and performed the experiments. Y.L. wrote the manuscript. P.W. supervised, analyzed the results, provided feedback and revised the manuscript. J.L. carried out the real world experiments and processed experimental data.

Funding: This work was supported by National Natural Science Foundation of China (No. U1713222, No. 61773139 and No. 61473015), the Natural Science Foundation of Heilongjiang Province, China (Grant No. F2015008) and Shenzhen Peacock Plan (No. KQTD2016112515134654).

Conflicts of Interest: The authors declares no conflict of interest.

References

1. Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [CrossRef]
2. Aine, S.; Swaminathan, S.; Narayanan, V.; Hwang, V.; Likhachev, M. Multi-heuristic a*. *Int. J. Robot. Res.* **2014**, *35*, 224–243. [CrossRef]
3. Zucker, M.; Ratliff, N.; Dragan, A.D.; Pivtoraiko, M.; Klingensmith, M.; Dellin, C.M. Chomp: Covariant hamiltonian optimization for motion planning. *Int. J. Robot. Res.* **2013**, *32*, 1164–1193. [CrossRef]
4. Kalakrishnan, M.; Chitta, S.; Theodorou, E.; Pastor, P. STOMP: Stochastic trajectory optimization for motion planning. In Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; Volume 19, pp. 4569–4574.
5. Lavalle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.
6. Kavraki, L.E.; Svestka, P.; Latombe, J.-C.; Overmars, M.H. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Trans. Robot. Autom.* **1994**, *12*, 566–580. [CrossRef]
7. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]

8. Laumond, J.-P. Kineo CAM: A success story of motion planning algorithms. *IEEE Robot. Autom. Mag.* **2006**, *13*, 90–93. [CrossRef]
9. Bonilla, M.; Pallottino, L.; Bicchi, A. Noninteracting constrained motion planning and control for robot manipulators. In Proceedings of the IEEE International Conference on Robotics and Automation, Singapore, 29 May–3 June 2017; pp. 4038–4043.
10. Bialkowski, J.; Otte, M.; Frazzoli, E. Free-configuration biased sampling for motion planning. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 1272–1279.
11. Berenson, D.; Srinivasa, S.; Kuffner, J. Task Space Regions: A Framework for Pose-Constrained Manipulation Planning. *Int. J. Robot. Res.*. **2011**, *30*, 1435–1460. [CrossRef]
12. Yakey, J.H.; LaValle, S.M.; Kavraki, L.E. Randomized path planning for linkages with closed kinematic chains. *IEEE Trans. Robot. Autom.* **2001**, *17*, 951–958. [CrossRef]
13. Stilman, M. Task constrained motion planning in robot joint space. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2017; pp. 3074–3081.
14. Kim, B.; Um, T.T.; Suh, C.; Park, F.C. Tangent bundle RRT: A randomized algorithm for constrained motion planning. *Robotica* **2016**, *34*, 202–225. [CrossRef]
15. Jaillet, L.; Porta, J.M. Path Planning Under Kinematic Constraints by Rapidly Exploring Manifolds. *IEEE Trans. Robot.* **2013**, *29*, 105–117. [CrossRef]
16. Bordalba, R.; Ros, L.; Porta, J.M. Kinodynamic planning on constraint manifolds. *arXiv* **2017**, arXiv:1705.07637.
17. Kingston, Z.; Moll, M.; Kavraki, L.E. Sampling-based methods for motion planning with constraints. *Annu. Rev. Control Robot. Auton. Syst.* **2018**, *1*, 159–185. [CrossRef]
18. Berenson, D.; Srinivasa, S.S.; Ferguson, D.; Kuffner, J.J. Manipulation planning on constraint manifolds. In Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 625–632.
19. Yang, C.; Peng, G.; Li, Y.; Cui, R.; Cheng, L. Neural networks enhanced adaptive admittance control of optimized robot-environment interaction. *IEEE Trans. Cybern.* **2018**, 1–12. [CrossRef]
20. Huh, J.; Lee D.D. Learning high-dimensional Mixture Models for fast collision detection in Rapidly-Exploring Random Trees. In Proceedings of the IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 63–69.
21. Arslan, O.; Tsiotras, P. Machine learning guided exploration for sampling-based motion planning algorithms. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Hamburg, Germany, 28 September–2 October 2015; pp. 2646–2652.
22. Yang, C.; Chen, C.; He, W.; Cui, R.; Li, Z. Robot Learning System Based on Adaptive Neural Control and Dynamic Movement Primitives. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, 1–11. [CrossRef] [PubMed]
23. Buss, S.R.; Kim, J.S. Selectively Damped Least Squares for Inverse Kinematics. *J. Graph. GPU Game Tools* **2005**, *10*, 37–49. [CrossRef]
24. Oriolo, G.; Ottavi, M.; Vendittelli, M. Probabilistic motion planning for redundant robots along given end-effector paths. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 30 September–4 October 2002; Volume 2, pp. 1657–1662.
25. Kanehiro, F.; Yoshida, E.; Yokoi, K. Efficient reaching motion planning and execution for exploration by humanoid robots. In Proceedings of the IEEE Intelligent Robots and Systems (IROS), Vilamoura, Portugal, 7–12 October 2012; pp. 1911–1916.
26. Mirabel, J.; Tonneau, S.; Fernbach, P.; Seppälä, A.K.; Campana, M.; Mansard, N.; Lamiraux, F. HPP: A new software for constrained motion planning. In Proceedings of the International Conference on Intelligent Robots and Systems, Daejeon, Korea, 9–14 October 2016; pp. 383-389.
27. Kuffner, J.J.; Kagami, S.; Nishiwaki, K.; Inaba, M.; Inoue, H. Dynamically-stable motion planning for humanoid robots. *Auton. Robots* **2002**, *12*, 105–118. [CrossRef]
28. Burget, F.; Hornung, A.; Bennewitz, M. Whole-body motion planning for manipulation of articulated objects. In Proceedings of the IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 1656–1662.

29. Sucan, I.A.; Chitta, S. Motion planning with constraints using configuration space approximations. In Proceedings of the IEEE Intelligent Robots and Systems (IROS), Vilamoura, Portugal, 7–12 October 2012; pp. 1904–1910.
30. Roy, N.; Newman, P.; Srinivasa, S. E-Graphs: Bootstrapping Planning with Experience Graphs. In Proceedings of the Robotics: Science and Systems Conference, Sydney, NSW, Australia, 9–13 July 2012.
31. Phillips, M.; Chitta, S.; Chitta, S.; Likhachev, M. Learning to plan for constrained manipulation from demonstrations. *Auton. Robots* **2016**, *40*, 109–124. [CrossRef]
32. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1953**, *13*, 21–27. [CrossRef]
33. Bentley, J.L. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* **1975**, *18*, 509–517. [CrossRef]
34. Otte, M.; Frazzoli, E. RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *Int. J. Robot. Res.* **2016**, *35*, 797–822. [CrossRef]
35. Bernstein, M.; De Silva, V.; Langford, J.C.; Tenenbaum, J.B. *Graph Approximations to Geodesics on Embedded Manifolds*; Technical Report; Department of Psychology, Stanford University: Stanford, CA, USA, 2000; Volume 24, pp. 153–158.
36. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
37. Sucan, I.A.; Moll, M.; Kavraki, L.E. The open motion planning library. *IEEE Robot. Autom. Mag.* **2012**, *19*, 72–82. [CrossRef]
38. Pan, J.; Chitta, S.; Manocha, D. FCL: A general purpose library for collision and proximity queries. In Proceedings of the IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 3859–3866.