2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)
September 12-15, 2023. Sinaia, Romania

Exploring feasibility maps for trajectory planning of redundant manipulators using RRT

Marc Fabregat-Jaen
Instituto de Investigacion en Ingenieria de Elche (I3E)
Universidad Miguel Hernandez de Elche
03202 Elche, Spain
mfabregat@umh.es

Adrián Peidro
Instituto de Investigacion en Ingenieria de Elche (I3E)
Universidad Miguel Hernandez de Elche
03202 Elche, Spain
apeidro@umh.es

Arturo Gil
Instituto de Investigacion en Ingenieria de Elche (I3E)
Universidad Miguel Hernandez de Elche
03202 Elche, Spain
arturo.gil@umh.es

David Valiente
Instituto de Investigacion en Ingenieria de Elche (I3E)
Universidad Miguel Hernandez de Elche
03202 Elche, Spain
dvaliente@umh.es

Oscar Reinoso
Instituto de Investigacion en Ingenieria de Elche (I3E), Universidad Miguel Hernandez de Elche, 03202 Elche, Spain
ValgrAI: Valencian Graduate School and Research Network of Artificial Intelligence, 46022 Valencia, Spain
o.reinoso@umh.es

***Abstract*—Redundant manipulators offer several advantages, including improved manipulability, singularity avoidance, and obstacle evasion. However, kinematic redundancy also introduces additional challenges, such as the need to solve an underdetermined inverse kinematic problem to control the manipulator. This paper introduces a novel approach for motion planning of redundant manipulators, based on the exploration of feasibility maps. The proposed method is an extension of the RRT algorithm, modified to explore the redundant space in order to find a suboptimal feasible path in the joint space, sacrificing optimality for scalability to higher degrees of redundancy. The method is able to follow a given task trajectory while considering other constraints, such as joint limits, self-collisions, and obstacles.**

***Index Terms*—redundant manipulator, inverse kinematics, motion planning**

I. Introduction

Kinematic redundancy occurs when a manipulator has more degrees of freedom (DOF) than necessary to perform a task. This is a common feature in many robotic systems, such as industrial manipulators, collaborative arms, or mobile manipulators. The presence of redundant DOF allows the manipulator to perform the task in multiple ways, which can be exploited to improve the performance of the robot [1]. For instance, redundancy can be used to avoid singularities [2], [3], improve manipulability [4], or avoid obstacles [5].

However, the existence of redundant DOF also introduces additional challenges, such as the need to solve the inverse kinematic problem (IKP) to obtain the joint configuration of the manipulator from a given task. It is well known that the IKP is an underdetermined problem for redundant manipulators, resulting in an infinite number of solutions.

The most common way to address this issue is by solving the IKP at a differential level, by means of the Moore-Penrose pseudoinversion of the Jacobian matrix $\mathbf{J}$ [6], in order to express joint velocities $\dot{\mathbf{q}}$ in terms of task velocities $\dot{\mathbf{x}}$. However, this approach does not ensure the avoidance of kinematic singularities, as proved in [7]. The authors of [2] and [3] proposed to use a damped least-squares approach to solve the IKP, which results in a nonsingular Jacobian throughout the workspace.

Task-space augmentation, proposed in [8], is another approach to solve the IKP for redundant manipulators. It adds additional parameters to the task, which are used to satisfy additional constraints. This way, the task vector can be augmented to equal the number of DOF of the manipulator, resulting in an augmented task for which the manipulator is nonredundant.

The recent trend is to use optimization-based approaches to solve the IKP, which allows to take into account additional constraints. Gradient descent is employed in [9] to maximize the distance to obstacles while minimizing the end-effector's orientation error. However, the typical approach is to formulate the IKP as a quadratic programming (QP) problem and obtain a solution using a QP solver [10] or [11]. In [12], this approach was used to achieve robust constrained control that can be used to incorporate hard joint constraints (e,g,. joint limits or obstacles). Joint torque optimization is achieved in [13] by formulating a QP problem subject to physical constraints.

This work is part of the project PID2020-116418RB-I00, funded by MCIN/AEI/10.13039/501100011033; part of the grant PRE2021-099226, funded by MCIN/AEI/10.13039/501100011033 and by the ESF+; and part of the project TED2021-130901B-I00, funded by MCIN/AEI/10.13039/501100011033 and by the European Union "NextGenerationEU/PRTR".

Neural networks (NN) have also been used to solve the IKP of redundant manipulators. In [14], the authors present a dual network architecture to solve the IKP of redundant manipulators. The authors in [15] propose using an ANFIS architecture [16], which combines fuzzy logic and NN, to obtain the joint configuration for a given task. For further information on the use of NN for solving the IKP, the reader is referred to [17].

Based on the concept of task-space augmentation, reference [18] presents feasibility maps, which are used to determine whether a manipulator is able to track a given trajectory while avoiding obstacles. A subsequent work [19], uses feasibility maps to plan collision-free trajectories for redundant manipulators minimizing the change in the redundant parameter. The concept is also expanded in [20] to plan the trajectory of a redundant parallel manipulator by transitioning among working modes. However, the path is not planned completely autonomously, as it needs a human operator to select some intermediate waypoints. In [21], an exhaustive grid-search on feasibility maps is performed using dynamic programming to find the globally optimal trajectory.

In this paper, we present a novel approach for motion planning of redundant manipulators, based on the exploration of feasibility maps. The proposed method is an extension of the Rapidly-exploring Random Trees (RRT) algorithm [22], modified to explore the redundant parameters in order to find a feasible path in the joint space. Like previous similar methods, the method is capable of handling constraints such as joint limits, self-collisions, or obstacles while following a given task trajectory. However, instead of performing exhaustive grid searches to find globally optimal trajectories, our proposed method uses an RRT-based sampling to find a suboptimal path in the space of redundant parameters, sacrificing global optimality for computational efficiency, which allows the method to scale to higher degrees of redundancy.

The rest of the paper is organized as follows. Section II reviews the concept of feasibility maps. The proposed method is presented in Section III, which uses RRT to explore the redundant space of the manipulator. In Section IV, the proposed method is evaluated in a simulated example. Finally, Section V presents the conclusions and future work.

II. Feasibility maps

The IKP refers to the task of obtaining the joint configuration $\mathbf{q}$ of an n-DOF manipulator from a given m-dimensional task $\mathbf{x}$, which is typically its end-effector position and orientation. The forward kinematic problem consists in computing the position and orientation $\mathbf{x}$ of the end-effector from the joint angles $\mathbf{q}$, which can be written as follows for serial robots:

$$\mathbf{x} = \mathbf{f}(\mathbf{q}) \tag{1}$$

When solving the IKP, it would be desirable to be able to invert (1) and obtain an expression as follows, which provides the joint coordinates as a function of the task variables:

$$\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x}) \tag{2}$$

However, in general it is not possible to obtain a global inverse function, because normally, for nonredundant manipulators ($n = m$), a single $\mathbf{x}$ can be achieved by a finite number of different solutions for $\mathbf{q}$, i.e., the inverse kinematic mapping is multivalued. This is even worse for kinematically redundant manipulators ($n > m$), for which equation (1) admits infinitely many different solutions for $\mathbf{q}$ for a given $\mathbf{x}$. The degree of redundancy r can be defined as the difference between the DOF of manipulator and task:

$$r = n - m \tag{3}$$

One way to address this issue is by using an augmented task space, which causes the IKP to become determined and can be solved by specifying the redundant parameters of the task. For a manipulator performing a task $\mathbf{x}$ with r degrees of kinematic redundancy, the augmented task $\mathbf{x}_a$ is defined as an n-dimensional vector, for which the manipulator becomes nonredundant [23]:

$$\mathbf{x}_a = \begin{bmatrix} x_1 & \cdots & x_m & x_{m+1} & \cdots & x_{m+r} \end{bmatrix}^\intercal = \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_r \end{bmatrix} \tag{4}$$

where $\mathbf{x}_r$ is an additional r-dimensional task vector, whose components are independent of every other in $\mathbf{x}_a$, and can be freely selected depending on the application needs. The redundant parameters vector $\mathbf{x}_r$ is typically comprised by joint coordinates or a differentiable function in terms of the joints vector, such as the position or orientation of a tracked point of the manipulator.

In the presence of multiple constraints, such as joint limits or obstacles that must be avoided, the set of feasible configurations that satisfy the desired task can be depicted in a $(r+1)$-dimensional map. The extra dimension corresponds to the time dimension on which the task is defined, which often is the trajectory followed by the end-effector over time. These maps are known as feasibility maps and were first introduced in [18]. A feasibility map $\mathcal{FM}$ is defined as the set of points in the $(t, x_{m+1}, \ldots, x_{m+r})$ space, where t is the time, for which there exists a feasible joint configuration that satisfies the commanded task and the imposed constraints.

To illustrate the concept, let us consider a 2R planar manipulator ($n = 2$), shown in Fig. 1, whose link lengths are $l_1 = 1$ and $l_2 = 1$, and whose joint configuration is defined by $\mathbf{q} = \begin{bmatrix} q_1 & q_2 \end{bmatrix}^\intercal$. It executes a task $\mathbf{x} = \begin{bmatrix} p_y \end{bmatrix}$ defined by the position of the $\mathbf{Y}$ coordinate of its end-effector p_y over time t ($m = 1$), where the desired trajectory is $p_y = -6.662t^2 + 8.162t - 1.5$, with t varying from 0 to 1. This scenario results on the manipulator having $r = 1$ degree of redundancy for achieving the commanded task. Therefore, an augmented task can be defined by adding an extra parameter to the task vector $\mathbf{x}$. In this example, the first revolute joint q_1 is chosen as the redundant joint, resulting in the augmented task vector $\mathbf{x}_a$:

$$\mathbf{x}_a = \begin{bmatrix} p_y \\ q_1 \end{bmatrix} \tag{5}$$

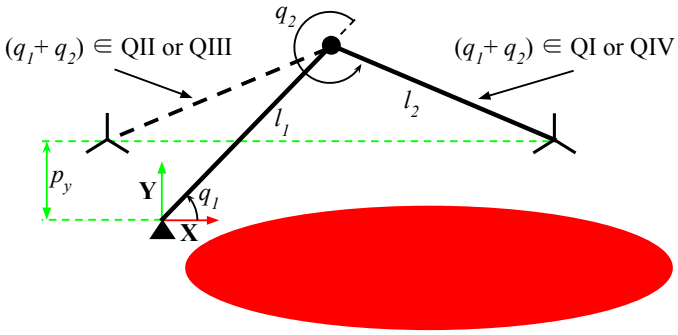


Fig. 1. 2R planar manipulator with an elliptic obstacle.

In order to solve the IKP, the remaining joint coordinates must be expressed in terms of the augmented task parameters. Following the example, q_2 must be defined in terms of p_y and q_1:

$$q_2 = \arcsin\left(\frac{p_y - l_1 \sin(q_1)}{l_2}\right) - q_1 \tag{6}$$

or

$$q_2 = \pi - \arcsin\left(\frac{p_y - l_1 \sin(q_1)}{l_2}\right) - q_1 \tag{7}$$

where l_i is the length of i-th link that constitutes the manipulator. Note that there exist two solutions to the IKP depending on the quadrant where $(q_1 + q_2)$ is located. When the angle lies in the first (QI) or fourth quadrant (QIV), the value of q_2 is calculated as in (6) and the resulting joint configuration corresponds to the one drawn in solid line in Fig. 1. Otherwise, the angle belongs in the second (QII) or third quadrant (QIII) and (7) is employed. The solution for the latter case is plotted in dashed line. Consequently, two different feasibility maps can be extracted from the task, one when (6) is used for solving q_2, and another when using (7).

The environment on which the manipulator operates is shown in Fig. 1. The red ellipse represents a forbidden region for the end-effector, and it is defined by the following inequation:

$$\frac{(p_x - 1.1)^2}{1^2} + \frac{(p_y + 0.2)^2}{0.25^2} \leq 1 \tag{8}$$

where p_x and p_y are the coordinates of the end-effector in the reference system of the base, identified by $\mathbf{X}$ and $\mathbf{Y}$ in Fig. 1. The ellipse is an obstacle that the end-effector must avoid colliding with (p_x and p_y must not satisfy (8)), but the rest of the manipulator can intersect with it.

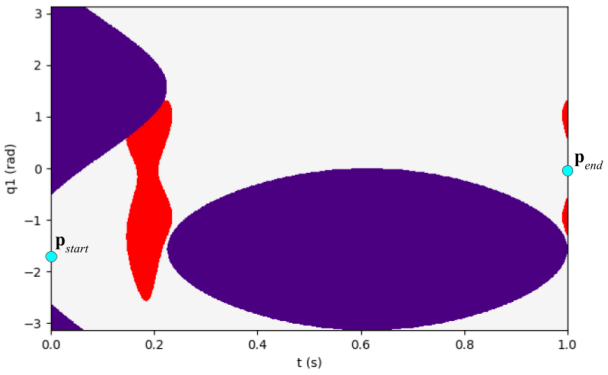
Fig. 2 shows the feasibility maps of the 2R planar manipulator, which are $r + 1 = 2$-dimensional and can be plotted in a 2D space. The horizontal axis represents time t, while the vertical axis represents the joint coordinate q_1, which is the only parameter in the additional task vector $\mathbf{x}_r$. The elliptic obstacle in the task space (Fig. 1) appears as red regions in the plot. Points in the (t, q_1) space that lead to a complex solution of the IKP (i.e., points for which the argument of the arcsin in (6) and (7) is higher than 1 or smaller than -1) are represented in purple.

It is important to note that the actual feasibility maps are strictly limited to the uncolored regions of the plot and the colored regions do not pertain to them because they imply forbidden configurations (either collisions or complex solutions). The upper and lower boundaries of the feasibility maps are also established by the joint limits in case the values of the vector $\mathbf{x}_r$ of redundant parameters are chosen as such. For example, in Fig. 2, it is assumed that q_1 is unbounded, so the displayed limits are $\pm\pi$ radians.

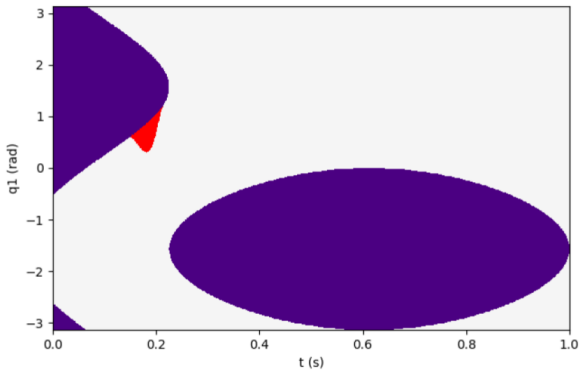
III. Proposed Motion Planning Algorithm

Feasibility maps depict every feasible point in a $(t, \mathbf{x}_r)$ space that meets specific constraints to accomplish a given task. By exploring feasibility maps, it is possible to determine a continuous trajectory in the redundant space $\mathbf{x}_r$ over time t, for a prescribed desired task trajectory $\mathbf{x}(t)$ parameterized in terms of t (which typically represents time, or an arc-length parameter).

As an example, consider the feasibility map $\mathcal{FM}$ shown in Fig. 2a and a pair of start $\mathbf{p}_{start} = [t_{start}, q_{1,start}]$ and



(a) $(q_1 + q_2) \in$ QI or QIV



(b) $(q_1 + q_2) \in$ QII or QIII

Fig. 2. Feasibility maps of the 2R planar manipulator for the given task and constraints.

end points $\mathbf{p}_{end} = [t_{end}, q_{1end}]$. Note that the augmented task vector $\mathbf{x}_a$ is uniquely defined for each point in the map, since it is defined by the task p_y, which only depends on t, and the redundant joint q_1. Consequently, the joint configuration $\mathbf{q}$ can be calculated at any point in the map by means of (6).

A feasible path in the redundant space over time can be established if a set of points that belong to the feasibility map connect the initial and final configurations. In other words, a feasible path $\mathcal{P} = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$ exists if there is a continuous set of points $\mathbf{p}_i = [t_i, \mathbf{x}_{ri}^\intercal]$ such that $\mathbf{p}_i \in \mathcal{FM}$ for all i, where $t_i > t_{i-1}$, $\mathbf{p}_0 = \mathbf{p}_{start}$ and $\mathbf{p}_n = \mathbf{p}_{end}$.

Fig. 3 depicts two feasible paths connecting the start and end points in the feasibility map. The solid-line path navigates a narrow passage between the red obstacle region and the purple region where the IKP has complex solutions. On the other hand, the dashed-line path takes advantage of angular wrapping in q_1 through $\pm\pi$, resulting in a more direct path that avoids the narrow passage. If the joint limits for q_1 constrained the joint to remain within the range of $[-\pi, \pi]$, then the solid-line path would be the only feasible option, as the wrapping would be impossible.

Furthermore, if the end configuration, determined by $\mathbf{p}_{end}$, is irrelevant as long as it is feasible, then any path ending in the vertical green line shown in Fig. 3, where the time is equal to t_{end} ($t_{end} = 1$ s in this case), would be considered valid. If the final time t_{end} is reached, it means that the task, determined by the trajectory $\mathbf{x}$ has been completed. This fact will be exploited in the proposed algorithm.

Path-planning is a well-established problem in robotics and there are many algorithms that can be employed to solve it. Search-based algorithms, like Dijkstra's [24], A* [25] and its variations [26]–[28], are capable of finding the optimal path in a graph by exploring and evaluating paths using a heuristic function that guides the search. However, as the dimensionality of the problem grows, their computational costs become prohibitively expensive.

In contrast, sampling-based algorithms, such as Probabilistic Roadmaps (PRM) [29] and Rapidly-exploring Random Trees (RRT) [22], can identify a feasible path more efficiently, regardless of the dimensionality of the search space. Nonetheless, they do not guarantee the optimality of the found path.

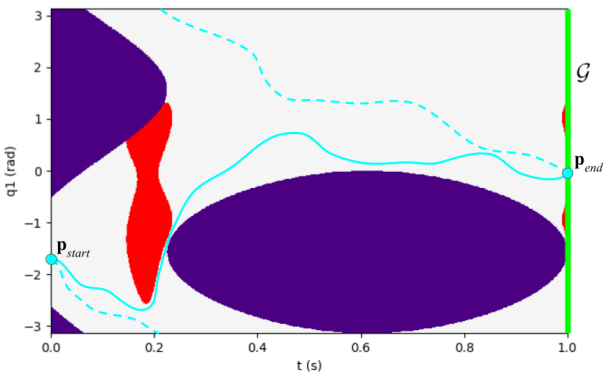


Fig. 3. Two possible paths in the feasibility map.

RRT is a popular sampling-based motion-planning algorithm that explores a given space and constructs a tree by randomly sampling points and connecting them to the nearest point in the tree. Starting from an initial state $\mathbf{p}_{start}$, the RRT algorithm generates a tree $\mathcal{T}$ by iteratively adding new nodes to the tree until a path that reaches a final state $\mathbf{p}_{end}$ is found.

RRT was first introduced in [22] as a randomized data structure for path planning in high-dimensional spaces in an incremental way. Since then, many variations of the original algorithm have been proposed to address different challenges and requirements. RRT-Connect extends the RRT algorithm to allow for bidirectional search between the start and end points [30]. In [31], ERRT is proposed to make the original RRT algorithm suitable for real-time applications. Dynamic RRT [32] provides a method to replan the path when the environment changes. RRT* [33] and Anytime RRT [34] are extensions that ensure the optimality of the found path when given enough time.

Due to its scalability to a higher number of dimensions, RRT has been widely employed for motion planning of redundant manipulators with collision avoidance [35], [36]. Recent works have proposed various enhancements, such as using an artificial potential field on the target configuration to guide the tree expansion towards the goal [37], a bidirectional RRT-based algorithm that can directly connect both generated trees to avoid obstacles [38], and another variant that leverages the scalability of the method to plan the path of a 17-DOF robot while avoiding obstacles and adhering to the joint deflection angles [39].

A. Modified RRT algorithm

A modified RRT algorithm that explores the feasibility map of a redundant manipulator to determine a feasible path in the redundant space is proposed in this subsection. It is an adaptation of the original RRT with some modifications to account for the particularities of the problem at hand.

The main difference between the original RRT algorithm and the proposed algorithm is that the latter does not have a final state $\mathbf{p}_{end}$. Instead, our algorithm searches for a feasible path that reaches the goal subset $\mathcal{G} = \{[t, \mathbf{x}_r^\intercal] \in \mathcal{FM} : t = t_{end}\}$, regardless of the final configuration $\mathbf{x}_r$, which is not of interest, as long as it is feasible. From the set of feasible paths found, the algorithm will select the one that minimizes a cost function $c(\mathbf{p})$ that will be defined later.

Moreover, unlike most RRT-based algorithms [40], the proposed algorithm uses online planning. That is, the algorithm does not require to precompute the whole feasibility map in advance, it exists implicitly. In Figure 2, the feasibility maps, which are the white obstacle-free regions due to constraints, have been precomputed in advance only for illustrative purposes. However, during the online execution of the algorithm, the map is generated and explored in real time as required. Algorithm 1 presents the proposed algorithm and Fig. 4 illustrates its most important steps, which are described over the next paragraphs.

Algorithm 1: Modified RRT for feasibility maps

$\mathcal{T} \leftarrow$ `InitializeTree`$(\mathbf{p}_{start})$
for $i = 1, 2, \ldots, i_{max}$ **do**
  $\mathbf{p}_{rand} \leftarrow$ `RandomNode`
  $\mathbf{p}_{parent} \leftarrow$ `BestParent`$(\mathbf{p}_{rand})$
  if conditions (C1), (C2) and (C3) are met **then**
    Add node $\mathbf{p}_{rand}$ with parent $\mathbf{p}_{parent}$ to $\mathcal{T}$
    $\mathbf{p}_{ext} \leftarrow$ `ExtendPath`$(\mathbf{p}_{parent}, \mathbf{p}_{rand})$
    if conditions (C2) and (C3) are met **then**
      Add node $\mathbf{p}_{ext}$ with parent $\mathbf{p}_{rand}$ to $\mathcal{T}$
$\mathcal{P}_{best} \leftarrow$ `BestPath`$(\mathcal{T}, c(\mathbf{p}))$
$\mathcal{P}_{smooth} \leftarrow$ `SmoothPath`$(\mathcal{P}_{best})$
return $\mathcal{P}_{smooth}$

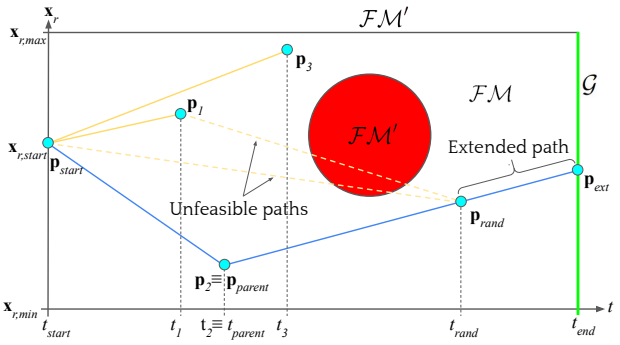


Fig. 4. Proposed modified RRT algorithm for feasibility maps, where $\mathcal{FM}'$ is the complement of $\mathcal{FM}$.

The algorithm starts by initalizing the tree $\mathcal{T}$ with the starting state $\mathbf{p}_{start}$, which is a point in the feasibility map $\mathcal{FM}$. The tree is a data structure that establishes a hierarchical relationship between its nodes, each of which corresponds to a state $\mathbf{p} \in \mathcal{FM}$. Each node may have multiple descendants but only a single parent. The initial node $\mathbf{p}_{start}$ is the root of the tree and has no parent.

Next, the algorithm performs i_{max} iterations, where i_{max} is a predefined maximum number of iterations. In each iteration, a random node $\mathbf{p}_{rand} \in \mathcal{FM}$ is generated, guaranteeing its feasibility since it is contained in $\mathcal{FM}$. The algorithm then searches for the best parent node $\mathbf{p}_{parent}$ for $\mathbf{p}_{rand}$ (we will explain next what means to be the best parent node).

The search process must respect the requirement that time is strictly monotonically increasing, which requires that the time coordinate of the parent node t_{parent} must be less than the time coordinate of the random node t_{rand}, i.e., $t_{parent} < t_{rand}$. The search is made forward in the time dimension, starting from the root node, until a node $\mathbf{p}_{parent}$ is found such that: (C1) $t_{parent} < t_{rand}$, (C2) the straight path between $\mathbf{p}_{parent}$ and $\mathbf{p}_{rand}$ completely lies in $\mathcal{FM}$ (i.e., it does not intersect forbidden regions), and (C3) other requirements, which are specific to the problem at hand, are met. For instance, one such requirement could be that the velocity of some component $\dot{q}_i$ of $\mathbf{x}_r$ lies within its established limits $[\dot{q}_{i,min}, \dot{q}_{i,max}]$:

$$\dot{q}_{i,min} \leq \frac{q_{i,rand} - q_{i,parent}}{t_{rand} - t_{parent}} \leq \dot{q}_{i,max} \tag{9}$$

where t_j and $q_{i,j}$ are directly extracted from the node $\mathbf{p}_j$.

The most computationally demanding task of the algorithm is verifying condition (C2), i.e., whether the straight line path lies completely within $\mathcal{FM}$. Since the feasibility map is not precomputed, this verification must be performed online. The feasibility of the straight path between $\mathbf{p}_{parent}$ and $\mathbf{p}_{rand}$ is checked by discretizing it with the same resolution as the map and verifying if every discretized point lies within $\mathcal{FM}$. To improve the efficiency of the process, these discretized points are checked in random order following a continuous uniform distribution. This approach increases the probability of identifying an unfeasible point in the path early and results in faster rejection of the path if it is invalid. If the path is valid, $\mathbf{p}_{rand}$ is added to the tree as a child of $\mathbf{p}_{parent}$ and the path is extended.

The extension process consists in extending the previously-verified valid straight path from $\mathbf{p}_{rand}$ to an extended node $\mathbf{p}_{ext}$, which is the intersection of the elongated straight path with the vertical goal subset $\mathcal{G}$. Next, the algorithm tests the straight path between $\mathbf{p}_{rand}$ and $\mathbf{p}_{ext}$ by verifying conditions (C2) and (C3) as explained above. If the extended path is feasible, $\mathbf{p}_{ext}$ is added to the tree as a child of $\mathbf{p}_{rand}$, meaning that a complete feasible path $\mathcal{P}$ has been found.

In Fig. 4, the algorithm is applied to an example scenario where the feasibility map is bounded by t_{start} and t_{end} in the time dimension, $\mathbf{x}_{r,min}$ and $\mathbf{x}_{r,max}$ in the redundant parameters dimension, and a red region representing an obstacle in the feasibility map. At the current iteration shown in Fig. 4, the tree contains four nodes: $\mathbf{p}_{start}$, $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$.

The algorithm generates the random node $\mathbf{p}_{rand}$ and searches for the best parent node $\mathbf{p}_{parent}$. The straight path between $\mathbf{p}_{start}$ and $\mathbf{p}_{rand}$ is first checked for feasibility and rejected because it crosses the red region, which is not in the feasibility map. The remaining nodes are checked iteratively in the order determined by t, until $\mathbf{p}_2$ is identified as the best parent node $\mathbf{p}_{parent}$. Note that $\mathbf{p}_3$ has not been tested at the moment $\mathbf{p}_{parent}$ is determined since $t_3 > t_2$.

Then, the straight path between $\mathbf{p}_{parent}$ and $\mathbf{p}_{rand}$ is extended to $\mathbf{p}_{ext} \in \mathcal{G}$ and its feasibility is verified, resulting in a complete path $\mathcal{P}$ that connects $\mathbf{p}_{start}$ and $\mathcal{G}$.

Once the maximum number of iterations i_{max} has been reached, the algorithm evaluates the tree and selects the best path $\mathcal{P}_{best}$ using a cost function $c(\mathbf{p})$. This function evaluates the cost of the connection between a node and its parent, and should be chosen depending on the problem at hand. For instance, a simple choice would be to minimize the sum of the weighted norm of the segments of the path. For a node $\mathbf{p}_c$ with parent $\mathbf{p}_p$, this cost function would be:

$$c(\mathbf{p}_c) = \sqrt{(\mathbf{p}_c - \mathbf{p}_p)\ \mathbf{W}\ (\mathbf{p}_c - \mathbf{p}_p)^\intercal} \tag{10}$$

where $\mathbf{W}$ is a diagonal matrix, whose diagonal elements are the weights w_i associated to each component of $\mathbf{p}$. Meaning

that, for a higher w_i, the cost function will prioritize the minimization of the i-th component of $\mathbf{p}$.

The cost function is computed recursively for each path, by starting from its final node, whose time coordinate equals the end time ($t = t_{end}$), and adding its cost to that of its parent. This operation is recursively performed until the root node is reached, accumulating the cost of each segment of the path. The path with the lowest accumulated cost is selected as the best path $\mathcal{P}_{best}$.

Due to the nature of the search algorithm, the path returned is polygonal, which means discontinuous velocities and infinite accelerations, which is not desirable in most applications. To address this, $\mathcal{P}_{best}$ is smoothed using a B-spline interpolation.

Given a path $\mathcal{P} = \{\mathbf{p}_{start}, \mathbf{p}_1, \ldots, \mathbf{p}_{n_p}, \mathbf{p}_{end}\}$, consisting of the start and end nodes and a set of n_p intermediate nodes, a set of control points $\mathcal{C}$ is established along the path:

$$\mathcal{C} = \{{}^1\mathbf{c}_1, {}^1\mathbf{c}_2, \ldots, {}^1\mathbf{c}_{n_c}, {}^2\mathbf{c}_1, {}^2\mathbf{c}_2, \ldots {}^2\mathbf{c}_{n_c}, \ldots\} \tag{11}$$

where n_c is the number of control points for each link (e.g., from $\mathbf{p}_{start}$ to $\mathbf{p}_1$) of the complete path $\mathcal{P}$. The control point ${}^i\mathbf{c}_j$ is the j-th control point of the i-th link that forms $\mathcal{P}$. The control points in each link are regularly spaced along it. The number of control points must be high enough. Moreover, the abrupt changes of direction of the original path, are smoothed with arcs that remain close to it. This ensures that the resulting path does not deviate significantly from the original one in order to maintain the feasibility of the path. The notation and resulting control points are illustrated in an example in Fig. 5.

The control points $\mathcal{C}$ are then interpolated using a cubic (degree 3) B-spline interpolation [41], which results in an smooth path $\mathcal{P}_{smooth}$, as shown in Fig. 5. This smooth path is the output of the algorithm.

The returned path includes the time values and the redundant parameters in each time step. By solving $\mathbf{q}$ from $\mathbf{x}_a$ as explained in Section II, the joint configuration can be derived from the redundant parameters at each time. Once the joint configuration is determined for each time step, the manipulator can be controlled to perform the task using a standard joint-space controller.

IV. Simulations

In order to validate the proposed algorithm, it has been simulated under two different scenarios. The first one corresponds to the one shown in Fig. 1, where a 2R planar manipulator is required to follow a commanded trajectory on the $\mathbf{Y}$ coordinate of its end-effector (p_y) while avoiding placing it inside the elliptic obstacle in red.

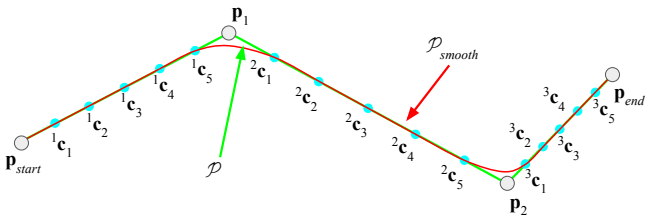


Fig. 5. B-spline interpolation $\mathcal{P}_{smooth}$ (in red) of the path $\mathcal{P}$ (in green), with $n_c = 5$.

The commanded trajectory is given by $p_y(t) = -6.662t^2 + 8.162t - 1.5$, with t varying from 0 to 1 seconds, which causes the vertical coordinate of the end-effector to describe a parabolic trajectory versus time, passing through the points $(0, -1.5)$, $(0.613, 1)$ (maximum), and $(1, 0)$ in the (t, p_y) space. The starting configuration of the manipulator is $\mathbf{q} = [-0.698, -0.331]$ rad.

Since the sum of the joint angles lies in QIV, (6) will be used to solve the IKP, corresponding to the feasibility map shown in Fig. 2a. It is appropriate to remind the reader that the algorithm does not have prior knowledge of the feasibility map, which is explored and generated in real time, and is only shown for illustrative purposes. The resolution employed for the check of the feasibility of the segments ((C2) in Section III-A) is 0.005 seconds.

The joints of the manipulator are constrained to lie in the range $[-2\pi, 2\pi]$ radians to allow for angle wrapping at $\pm\pi$, and the maximum joint velocities are limited to 13 rad/s. Equation (10) is employed as the cost function $c(\mathbf{p})$, with an identity weight matrix $\mathbf{W} = \mathbf{I}$.

The algorithm is executed with a maximum number of iterations i_{max}, which means that i_{max} random nodes $\mathbf{p}_{rand}$ will be generated, along with the likely extended nodes $\mathbf{p}_{ext}$, whose number is not affected by the maximum number of iterations. For path smoothing, $n_c = 6$ control points are employed per link of the path during the B-spline interpolation.

In Table I we show the results, averaged over 100 runs, for a selection of values of i_{max}. Simulations have been performed in Python on an Intel Core i5-10400 CPU @ 2.90GHz.

To evaluate the quality of the obtained paths, the optimal path $\mathcal{P}_{opt}$ is calculated using the A* algorithm [25]. The cost of the optimal path is 3.003, and it is plotted in magenta in Fig. 6. Fig. 7 shows the relationship between the relative error of the average path per maximum number of iterations, together with the best-fit hyperbola, providing visual representation of the quality of the paths obtained by the algorithm. This error is defined as the difference between the cost of the path obtained

TABLE I
AVERAGE TIME AND COST (OVER 100 RUNS) FOR DIFFERENT i_{max}

i_{max}	Average time (s)	Average cost
100	0.217	3.974
200	0.974	3.473
300	1.232	3.415
400	1.879	3.312
500	**2.216**	**3.258**
600	3.659	3.249
700	4.489	3.208
800	5.074	3.226
900	6.267	3.208
1000	7.124	3.208
2500	37.442	3.173
5000	140.841	3.165
7500	302.732	3.163

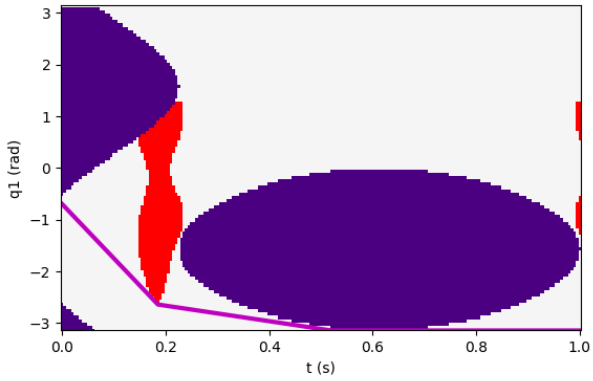


Fig. 6. Best path output by the A* algorithm.

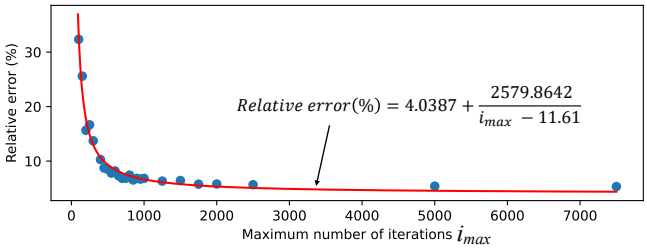


Fig. 7. Relative error of the average path per i_{max}.

by the algorithm and the cost of the optimal path. It has been observed that the failure rate, which is defined as the percentage of runs in which the algorithm does not find a feasible path, drops to 0% after $i_{max} = 70$, thus the data is not shown in the table. Considering the results, we have chosen $i_{max} = 500$ as the optimal maximum number of iterations for this example, since it provides a good trade-off between time and quality of the path returned.

Fig. 8 shows the tree generated by the algorithm with $i_{max} = 500$. Every path is colored based on its status: light red for paths that do not reach t_{end}, green for paths that reach t_{end} and blue for the smooth path $\mathcal{P}_{smooth}$ returned.

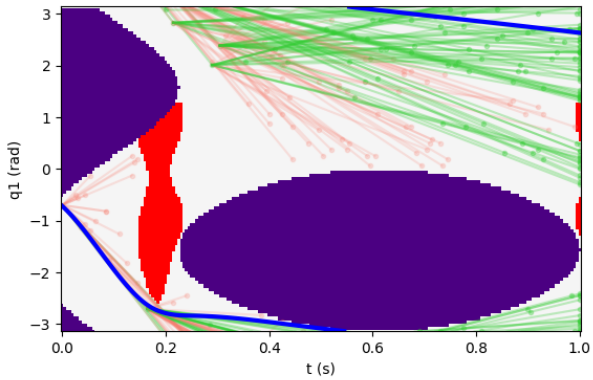


Fig. 8. Output of the modified RRT algorithm with $i_{max} = 500$.

In the following simulation, consider the same scenario (Figure 1), but the length of the first link l_1 of the manipulator depends now on the value of an additional prismatic joint q_2: $l_1 = 0.5 + q_2$, with q_2 constrained in the range $[0, 0.5]$ m. The joint stated as q_2 in Figure 1 now is q_3. With those new definitions of q_i, the degree of redundancy is $r = 2$ and the new redundant parameters vector is $\mathbf{x}_r = [q_1, q_2]^\intercal$. The starting configuration of the manipulator is now $\mathbf{q} = [-0.698 \text{ rad}, 0.5 \text{ m}, -0.331 \text{ rad}]$.

Following the same process as in the previous example, yields the optimal $i_{max} = 2100$. However, the obtained data and its analysis are not shown for brevity. Averaged over 100 runs, the algorithm takes 1.367 seconds to find a path with a cost of 3.642. Also, the failure rate is 0% for $i_{max} \geq 1600$. In Figure 9, the tree generated by the algorithm is shown along with the feasibility map, which is now three-dimensional ($r + 1 = 3$). The same color scheme as in Figure 8 is employed for representing every obtained path.

V. Conclusions

This paper has presented a new algorithm for motion planning of redundant manipulators that builds upon the RRT algorithm. The proposed method randomly samples points in the task's feasibility map subject to certain constraints and iteratively builds a tree by connecting these points to previous nodes in the tree. Paths that reach the goal are evaluated based on a cost function and the one that minimizes it is selected. The chosen path is then smoothed using a B-spline interpolation. The algorithm has been evaluated on a 2R and an RPR planar manipulator and was able to generate feasible paths efficiently.

Future work will involve extending the algorithm to manipulators with higher degrees of redundancy r, which will require exploring higher dimensional feasibility maps, for

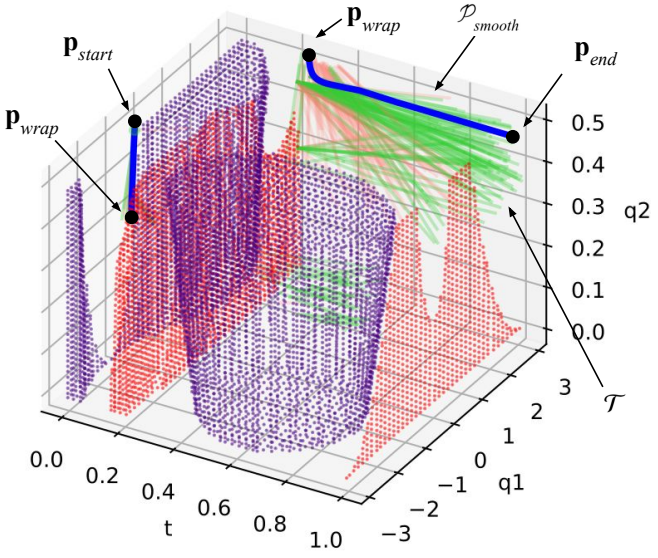


Fig. 9. Output of the modified RRT algorithm for the RPR manipulator with $i_{max} = 2100$.

which exhaustive grid-based searches, like classic A*, would require a prohibitive amount of time. Successful preliminary tests on a manipulator with $r = 5$ suggest promising scalability to more complex scenarios. Also, we will test the method on real manipulators.

REFERENCES

[1] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*. Springer, 2008, vol. 200.

[2] Y. Nakamura and H. Hanafusa, "Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 108, no. 3, pp. 163–171, 09 1986.

[3] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[4] L. Jin, S. Li, H. M. La, and X. Luo, "Manipulability optimization of redundant manipulators using dynamic neural networks," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 4710–4720, 2017.

[5] Y. Zhang and J. Wang, "Obstacle avoidance for kinematically redundant manipulators using a dual neural network," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 752–759, 2004.

[6] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on man-machine systems*, vol. 10, no. 2, pp. 47–53, 1969.

[7] J. Baillieul, J. Hollerbach, and R. Brockett, "Programming and control of kinematically redundant manipulators," in *The 23rd IEEE Conference on Decision and Control*. IEEE, 1984, pp. 768–774.

[8] L. Sciavicco and B. Siciliano, "Solving the inverse kinematic problem for robotic manipulators," in *RoManSy 6: Proceedings of the Sixth CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*. Springer, 1987, pp. 107–114.

[9] A. Sepehri and A. M. Moghaddam, "A motion planning algorithm for redundant manipulators using rapidly exploring randomized trees and artificial potential fields," *IEEE Access*, vol. 9, pp. 26 059–26 070, 2021.

[10] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpoases: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, pp. 327–363, 2014.

[11] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[12] A. M. Zanchettin and P. Rocco, "Motion planning for robotic manipulators using robust constrained control," *Control Engineering Practice*, vol. 59, pp. 127–136, 2017.

[13] Y. Zhang, S. S. Ge, and T. heng Lee, "A unified quadratic-programming-based dynamical system approach to joint torque optimization of physically constrained redundant manipulators," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, pp. 2126–2132, 2004.

[14] Y. Xia and J. Wang, "A dual neural network for kinematic control of redundant robot manipulators," *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 31 1, pp. 147–54, 2001.

[15] S. Alavandar and M. J. Nigam, "Neuro-fuzzy based approach for inverse kinematics solution of industrial robot manipulators," *Int. J. Comput. Commun. Control*, vol. 3, pp. 224–234, 2008.

[16] J.-S. Jang, "Anfis: adaptive-network-based fuzzy inference system," *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[17] A. A. Hassan, M. El-Habrouk, and S. Deghedie, "Inverse kinematics of redundant manipulators formulated as quadratic programming optimization problem solved using recurrent neural networks: A review," *Robotica*, vol. 38, pp. 1495 – 1512, 2019.

[18] P. Wenger, P. Chedmail, and F. Reynier, "A global analysis of following trajectories by redundant manipulators in the presence of obstacles," *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pp. 901–906 vol.3, 1993.

[19] J. A. Pámanes G, P. Wenger, and J. L. Zapata D, "Motion planning of redundant manipulators for specified trajectory tasks," *Advances in Robot Kinematics: Theory and Applications*, pp. 203–212, 2002.

[20] D. Reveles, P. Wenger *et al.*, "Trajectory planning of kinematically redundant parallel manipulators by using multiple working modes," *Mechanism and Machine Theory*, vol. 98, pp. 216–230, 2016.

[21] E. Ferrentino and P. Chiacchio, "A topological approach to globally-optimal redundancy resolution with dynamic programming," in *RO-MANSY 22 – Robot Design, Dynamics and Control*, V. Arakelian and P. Wenger, Eds. Springer International Publishing, 2019, pp. 77–85.

[22] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[23] L. Sciavicco and B. Siciliano, "The augmented task space approach for redundant manipulator control," in *Robot Control 1988 (Syroco'88)*. Elsevier, 1989, pp. 125–129.

[24] E. W. Dijkstra, *A note on two problems in connexion with graphs:(Numerische Mathematik, 1 (1959), p 269-271)*. Stichting Mathematisch Centrum, 1959.

[25] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[26] I. Pohl, "First results on the effect of error in heuristic search," *Machine Intelligence*, vol. 5, pp. 219–236, 1970.

[27] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, Tech. Rep., 1993.

[28] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," *Advances in neural information processing systems*, vol. 16, 2003.

[29] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[30] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[31] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *IEEE/RSJ international conference on intelligent robots and systems*, vol. 3. IEEE, 2002, pp. 2383–2388.

[32] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1243–1248.

[33] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[34] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1478–1483.

[35] K.-T. Wei and B. Ren, "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved rrt algorithm," *Sensors (Basel, Switzerland)*, vol. 18, 2018.

[36] M. V. Weghe, D. Ferguson, and S. S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2007, pp. 477–482.

[37] Z. Chen, Y. Yang, X. Xu, and A. Rodic, "Path planning of redundant series manipulators based on improved rrt algorithms," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 553–557.

[38] J. Dai, Y. Zhang, and H. Deng, "Novel potential guided bidirectional rrt* with direct connection strategy for path planning of redundant robot manipulators in joint space," *IEEE Transactions on Industrial Electronics*, 2023.

[39] L. Jia, Y. Huang, T. Chen, Y. Guo, Y. Yin, and J. Chen, "Mda+ rrt: A general approach for resolving the problem of angle constraint for hyper-redundant manipulator," *Expert Systems with Applications*, vol. 193, p. 116379, 2022.

[40] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using rrt* based approaches: A survey and future directions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016.

[41] C. De Boor and C. De Boor, *A practical guide to splines*. Springer-verlag New York, 1978, vol. 27.