People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Power and Control

Final Year Project Report Presented in Partial Fulfilment of
the Requirementsfor the Degree of

MASTER

In **Electrical and Electronic Engineering**

Option: **Control**

Title:

Motion planning for a tethered mobile robot

Presented by:

AMMOUR Manel

Supervisor:

Dr. GUERNANE

Registration Number:……/2019

Acknowledgements

I would like to express appreciation and indebtedness to Dr.Guernane for his support, patience, and motivation. I am deeply grateful to him for encouraging me to take on this project. I owe him a very important debt for offering the necessary background and knowledge I needed.

I also would like to show my greatest appreciation to all the teachers, relatives, parents, and others who in one way or another encouraged me in the entire process.

Abstract

Recently there has been surge of research in motion planning for tethered robots. In this problem a planar robot is connected to a base via a flexible cable assumed to be always stretched of limited length L. The existence of the cable causes additional constraints on the motion of the robot. Our method consists of finding a shortest path for a robot having an initial cable layout from a given position to the goal. The shortcut algorithm is used to estimate the cable configuration by smoothing and refining the path constituted of the concatenation of the initial cable layout and the path candidate without violating the homotopy class. This heuristic determines whether a target position can be reached for a given tether configuration.

The algorithm was evaluated and validated on Matlab by planning trajectories in an environment with obstacles of different shapes, and by setting different cable lengths. The results of our path planning approach have shown that the obtained smoothed path between base and goal position matches exactly the cable's configuration. In addition, homotopy classes are preserved when performing the smoothing operation over the path.

Keywords: Tether, Motion planning, Probabilistic sampling based Algorithm, shortcutting Algorithm, RRT*, Homotopy classes.

Table of Contents

List of Figures

List of Tables

1.1. Introduction

Robotics is a combination of various disciplines such as electrical and electronics, computer science and engineering, mechatronics and mechanical design, control and automation systems. Progress in the field of science and technology has emerged new innovative and interesting fields. The domain of robotics is an example of one of such fields.

History of robotics dates back to 1920s. However, the last two decades witnessed technological as well as social revolution in this domain widening the spectrum of robot applications dramatically.

Today, robots are being used actively in rehabilitation, motion assistance, cognition, target detection and tracking in addition to Nuclear Power Plants (NPP), Space and numerous other industrial applications. However, the accomplishment of these tasks is limited by the batteries 'life span. In order to overcome this problem, a tethered robot to a base station plugged-in to a power source would guarantee extended operational periods. Eliminating on-board power also acts to reduce platform size, an important consideration for systems that typically must traverse through narrow openings. Nevertheless, the tether introduces topological constraints due to the presence of obstacles in the environment of the robot, and metric constraints related to the finite length of the cable that limits the reachable workspace.

In this chapter, we will discuss the advantages of using tethered robots in different fields and recite some of the existing robots. We will state the path planning problem of tethered mobile robots and discuss about related work to our approach.

1.2. Motivation

The robotic motion planning problem has received a considerable amount of attention, especially over the last decade, as robots started becoming a vital part of modern industry as well as our daily life.

Mobile robots are typically untethered. This is not always desirable in high-power robotics. Wireless communication can be unreliable and batteries need to be charged

regularly. These challenges can be solved by using cables for communication and power.

There are important applications in which mobile robots must be tethered to a base station such as rescue, underwater, space, volcanic exploration and a number of other terrestrial applications. A tether can provide a communication link enabling teleoperation in environments where wireless signals either may not work or be sufficient .This was the case when robots were deployed in the reactor building after the tragic accident in the Fukushima because the radioactive environment disrupted communication links .In disaster recovery operations a tether may also facilitate high bandwidth communications between the robot and a remotely located human user. As in many mobile robot applications, energy can be a scarce resource and a tether allows a robot to be connected to a power source allowing for longer missions [1].

Moreover, on-board power robots requires stronger-duty motors which make them expensive to build and overweighted due to the bulkiness and heaviness of the batteries.so, tethering will result in decreasing the robot size significantly and enabling it to maneuver in narrow spaces[2].

1.3. Problem Formulation

Consider the problem of path planning for a tethered mobile robot in a two dimensional workspace with arbitrarily shaped and fixed obstacles (O_i). The robot is considered to be holonomic ,also arbitrarily shaped, and is tethered to a fixed point b in the workspace. The tether has a maximum length L and is assumed to be flexible and retractable.

The robot's initial departure is at the anchor point thus; as it moves towards the starting position the configuration of the tether may follow exactly the shortest path between the base and the starting point or may have a given cable configuration.This is considered as the initial configuration of the tether defined as (Y_i) which does not exceed the cable's length L.

Our problem consists of finding the shortest path that connects the starting position (S) to the goal position (G) which yields to a given cable configuration that does not violate the length constraint .

1.4. Motivational Examples

We have selected some of the tethered mobile robots operating in different fields:

1.4.1. Axel Rover

Axel is an exploratory robot used to access extreme terrains in order to probe, sample, and measure.

The minimalistic Axel robot is a two-wheeled robot that can rappel down steep terrain using a tether in order to carry out science at close range (Fig. 1.1) .The tether is routed through the arm, and wound around a reel in the center of its body. A solo Axel is conceived as a daughter ship in a mother-daughter configuration, with Axel's tether anchored in the mother ship (a rover or lander at the top of the extreme terrain) (Fig. 1.2) .In addition to providing mechanical support; the tether can provide power and communication [3].

Fig. 1.1: A picture of Axel descending steep, rough terrain in Arizona desert. Tether highlighted in blue.

Fig 1.2: The DuAxel rover on rough terrain. DuAxel consists of two Axel-class rovers docked with a central module.

1.4.2. Atlas Robot

Atlas is a tethered massive humanoid robot that was built by Boston Dynamics whose goal is to develop robots that can help people respond to natural and other disasters. Atlas is intended to aid emergency services in search and rescue operations, performing

tasks such as shutting off valves, opening doors and operating powered equipment in environments where humans could not survive [4].

Fig. 1.3: Atlas DRC Robot with tether for networking and electric power.

1.4.3. Tethered quadrator

Unmanned quadrotor helicopter cannot continuously work for a long time because it is powered by batteries.

In [5] the author proposed an unmanned quadrotor helicopter attached to a cable. This kind of cable has a variable length, pay-out and reel-in from a winch that can be put on a base station. It also has the function that transports electric energy.

The unmanned quadrator helicopter is designed to perform aerial tasks in the field of surveillance, photographing, surveying, detection and many other fields.

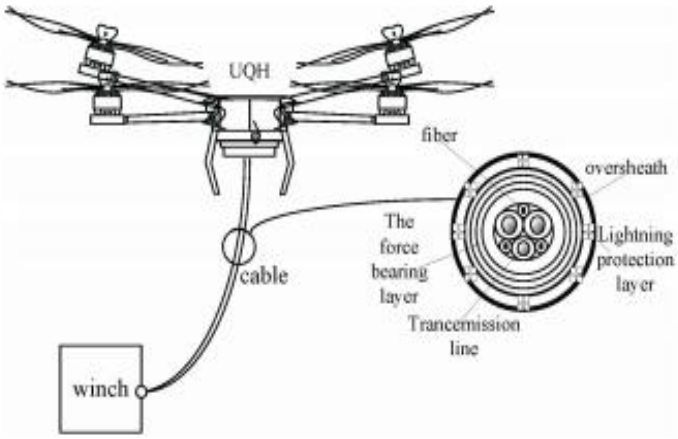


Fig 1.4: The tethered unmanned quadrotor helicopter and sectional view of the cable.

1.5. Related work

In motion planning for tethered robots, along with the usual constraints that are considered in any motion planning problem (collision avoidance, distance.),two important additional constraints are imposed on the motion of the robot due to the existence of the cable:

i) The radius of the robot's movement is limited by the cable's length.

ii) Topological constraints are imposed by the cable and obstacles in the environment.

The two additional constraints increase the complexity of the path planning.

Different approaches were proposed to treat this kind of path planning in two dimensions; here we will provide a list of the closely related work.

Homotopy class of the cable was considered in [6] to solve a similar problem. The authors considered the path length to the node to distinguish paths in different homotopy classes. Some important information may be lost by using the metric information of the path for the representation of homotopy. Due to this loss of information, the distinction between paths of different homotopy classes my fail which may result in infeasible paths. For example, they will not be able to differentiate between two paths of the same length even if one passes an obstacle on the right and the other one on the left.

The problem in [1] has been solved in a similar manner as in [6].However the novelty in this approach is that a true homotopy invariant (h-signature) is used to construct what we call a h-augmented graph that explicitly bears the topological information .The workspace had been represented using discrete graph representation of the environment through cell decomposition. The drawback of this method is that the robot size must be smaller than cell size. In the opposite case, it is not possible to uniquely determine the robot position. This decreases the possible range of the grid. There are also situations in which infeasible solutions are generated when moving from one cell to an adjacent cell [7].

In [8] the approximate grid approach in [1] is replaced by an optimal visibility-graph based solution, which is significantly faster but constrained to only polygonal obstacles.

In [9] the authors triangulated the environment by using all the edges of polygonal obstacles as edges of triangles. Then the authors built the visibility graph to find the shortest path. However, the suggested algorithm suffers from the computational complexity associated with the number of vertices of obstacles.

In [10] the workspace can be calculated by considering the topology class of the path and cable to guide a heuristic search in the form of heuristics. Recently-developed Multi-Heuristic A* allows to deal with numerous inadmissible heuristic functions while guaranteeing the suboptimality bound .As there could be too many possible topology classes and corresponding heuristic functions and not all of them are useful during the search, a Topology-based Multi-Heuristic A*is proposed, which starts as a normal weighted A* but shifts to Multi-Heuristic A* by adding new heuristic functions to escape from local minima.

Our approach differs from what had been suggested in previous work. A heuristic is used to represent the cable configuration consisting of a path, obtained after smoothing the initial cable layout with the path candidate in the same homotopy class. In case the goal is not reachable from the starting position, the robot is allowed to retract back to find the shortest accessible path.

1.6. Summary

This chapter presented our problem formulation, motivation for this work and some of the relevant work. In Chapter 2, we will present the preliminary materials and the necessary notions to do our study .Chapter 3 contain our proposed algorithm to solve this problem. Simulation and results are held in Chapter4.

2.1. Introduction

The necessary background is provided in this chapter. The Basic mathematical concept and the important notion of homotopy classes will be introduced in the upcoming sections.

Since our proposed algorithm is based on sampling-based algorithms; detailed information will be given about these approaches.

The advantages of using sampling-based methods are highlighted and a comparison between the different approaches is made.

2.2. Mathematical overview

The following sections introduce the necessary material required to understand our proposed approach for solving the motion planning problem which will be discussed in detail in the upcoming chapter.

2.2.1. Obstacles and the Configuration Space

The configuration space, or C-space, of the robot system is the space of all possible configurations of the system. Thus a configuration is simply a point in this abstract configuration space.

Throughout this thesis, we use 'x' to denote a configuration and "X" to denote the configuration space which is assumed to be a planner (R^2) that contains obstacles.

The path-planning problem is defined to be that of determining a continuous mapping,

c :[0 ;1] $\longrightarrow$ X, such that no configuration in the path causes a collision between the robot and an obstacle.

The configuration space obstacle X_{obst} is the set of all configurations occupied by obstacles in the C-space.

The free configuration space X_{free} is the set of configurations at which the robot does not intersect any obstacle, i.e.:

X_{free}=$X \backslash X_{obst}$ where the operator "\" is a subtraction operator.

2.2.2. Homotopic curves

Both cable configuration and robot paths are 1-dimensional curves in X_{free}.They can thus be defined as continuous maps from the interval [0, 1] to X_{free}. We will need to consider the homotopy class of both cables and robot trajectories.

Two curvesγ_1,γ_2 : [0, 1]⟶ X_{free} connecting the same start and end points, are homotopic (or belong to the same homotopy class) iff one can be continuously deformed into the other without intersecting any obstacle (see Figure 2.1).

Formally, if γ_1 : [0, 1] ⟶$X_{free}$and γ_2 : [0, 1] ⟶X_{free} represent the two trajectories (with$\gamma_1(0) = \gamma_2(0) = q_s$ and γ_1 (1) =γ_2 (1) =q_g), then γ_1 is homotopic to γ_2 iff there exists a continuous map η : [0, 1] ×[0, 1]⟶ X_{free} and such that:

- ✓ η (α , 0) = γ_1(α) ∀ α ∈ [0, 1]
- ✓ η (β, 1) =γ_2 (β) ∀ β ∈ [0, 1]
- ✓ η (0, μ) =q_s and η(1, μ) =q_g ∀ μ ∈ [0, 1] [1].

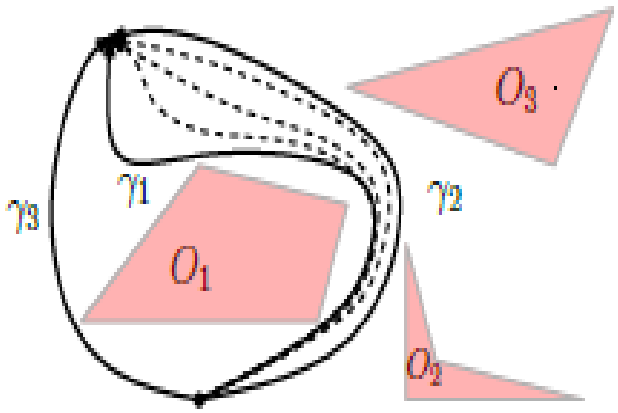


Figure 2.1: The two curvesγ_1, γ_2 are homotopic as there is a continuous deformation between the two that does not intersect any obstacle. γ_3 is not homotopic to γ_1, $\gamma_2$as any deformation will intersect an obstacle.

2.3. Collision Avoidance

The choice of the collision detection algorithm applied in motion planning significantly influences the overall performance of a planner due to its complexity. One approach to figure out whether a certain configuration is in collision is to check the interference between the robot and obstacle geometries. This can be accomplished by finding the minimum distance between the geometries. The configuration is said to be in collision if the distance is found to be negative. However, robots are often made up of complex geometries and it becomes a computationally expensive operation to find the minimum distance. Therefore, the robot geometry is usually approximated by simple geometric shapes, such as cylinders and spheres, in order to speed up the computation. Additionally these shapes are enlarged to obtain a so called safety margin.

2.4. Potential Based Path Planning

This method treats the environment as a potential field such that the goal point attracts and the obstacles repulse the agent. In spite of being useful as a real-time path planning, these methods suffer from trapping into local minima. Thus, potential field methods need additional effort to overcome the problem of local minima as well as to find a minimum length path to the goal point [11].

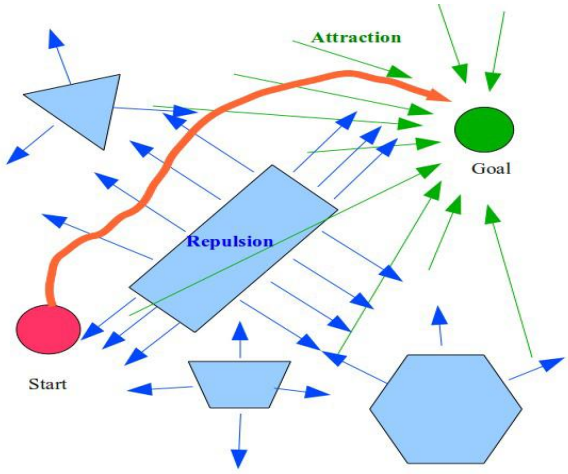


Figure 2.2: Potential based path planning.

2.5. Graph Based Path Planning

The methods associated with this approach usually make a grid of the environment and apply real-time versions of A* to it. Some of the methods simply divide the environment into simple polygonal grids. On the other hand, there are some methods that use Voronoi diagram to build a graph representing the environment. One drawback of graph-based path planning is that while the environment is being explored and a graph representing it is being constructed, further processing is needed to extract the path from the graph, such as A*. In multi-query tasks, the entire graph is searched to find a path to different goal points [11].

2.6. Sampling-based Algorithms

The right implementation of the cell decomposition methods and potential fields approach may relax the completeness requirement to, for example, resolution completeness. These approaches showed significant performance in accomplishing different tasks within acceptable time limits in complex environments. However, since decomposition-based methods suffered from large number of cells, and potential field methods from local minima, their state spaces are limited to up five dimensions in practical applications.

The above methods construct a solution from the explicit representation of the obstacles in the configuration space. Environments described by a large number of obstacles may result in an excessive computational burden. The idea to develop sampling-based algorithms came to avoid such a representation. These algorithms proved to be very effective for motion planning in high-dimensional spaces, and attracted significant attention over the last decade.

For this reason we will focus in the following sections on two algorithms, the Probabilistic Roadmap Planner and the Rapidly Exploring Random Trees (RRT) Planner, as representatives of multi-query and single-query planning algorithms.

2.6.1. Probabilistic Roadmaps (PRM)

Creating a path in the configuration space using the Probabilistic Roadmap Planner is considered as a two phase process: planning, and query. In the planning phase the planner keeps on sampling the configuration space until a certain number of collision free configurations has been generated. Each sampled configuration, or simply said point in C, is connected to the neighbors lying within an area of predefined size. Here, the connection between two points is established by a straight line path that is only maintained by the planner if it is found to be collision-free. The resulting network, called a roadmap, stored by the planner is then used to solve all subsequent motion planning queries.

A motion planning query is specified by a start and goal configuration, q_s and q_g. In order to solve a query, both configurations are connected in a first step to the respective closest configuration of the roadmap. Then, starting from q_s a solution path is found by moving iteratively to the neighboring node providing the minimum distance to the goal configuration. The distance metric used, usually corresponds to the Euclidean norm. The solution for a given start and goal configuration is shown in Figure 2.3 as a thick, solid path [12].

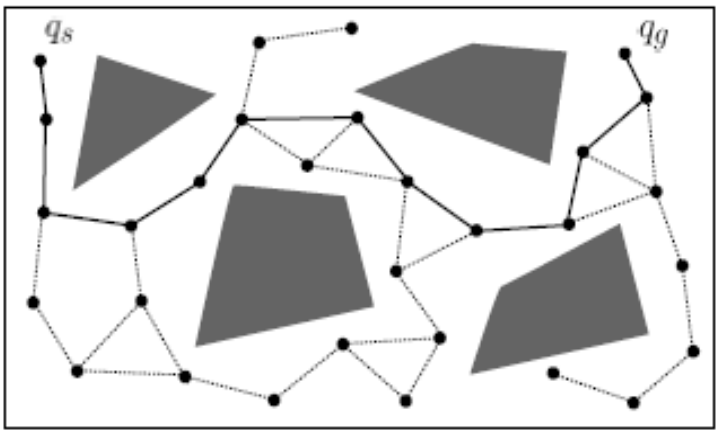


Figure 2.3: Probabilistic Roadmap Planner. Solution to a motion planning query (solid path).

2.6.2. Rapidly Exploring Random Trees (RRT)

A RRT Planner is a single-query probabilistic motion planning algorithm that searches for a path by incrementally expanding trees T in the configuration space. From an algorithmic point of view, trees are represented by a data structure, also referred to as a Rapidly Exploring Random Tree. As opposed to the PRM planner, this kind of algorithm only explores a subset of the configuration space, relevant for solving the query in mind. Furthermore, the tree growing procedure is repeated for each new query, for which reason this approach is referred to as a single-query method. The routine to be repeatedly applied for growing the trees in C proceeds as follows.

At the beginning of each iteration a random configuration $q_{rand}$is sampled from C according to a uniform probability distribution. Then, the closest configuration to q_{rand} in T, also called the nearest neighborq_{near}, is found and the local planner generates a new configuration q_{new} at a distance ε from q_{near} along the segment connecting q_{near} and q_{rand} as illustrated in Figure 2.4.

Afterwards, both $q_{new}$and the segment joining it to q_{near} are checked for collisions. If they are found to be collision-free, the search tree T is expanded by the new configuration and that segment.

The simplest form of a RRT Planner grows a single tree rooted at the start configuration q_{start}. Then, during the search the tree tries occasionally to connect to the goal configuration by using it asq_{rand} in the tree expansion procedure (see Figure 2.5) [12].

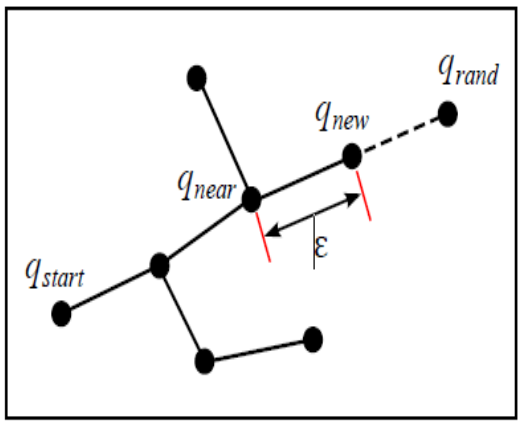


Figure 2.4: Example of a tree expansion step.

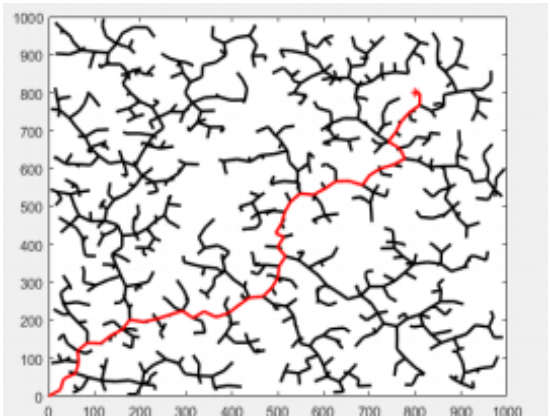


Figure 2.5: The tree constructed by RRt.The red line is a Path connecting the start to the goal configuration.

2.6.3. Advantages of Probabilistic Sampling Based Methods

Unlike most complete motion planning algorithms, sampling-based methods save large amounts of computations by avoiding explicit construction of obstacles in state space. These approaches rely on a collision checking module that gives information about the feasibility of candidate trajectories. Furthermore, they connect a set of points sampled from the obstacle-free space to build a graph (roadmap) or a tree structure of feasible trajectories. Even though these algorithms are not complete, they provide probabilistic completeness guarantees in the sense that the probability that the planner fails to return a solution, if one exists, decays to zero as the number of samples approaches infinity [13].

2.6.4. Optimal Sampling-based Algorithms

It has been proven that the standard PRM and RRT algorithms are not asymptotically optimal, and that the simplified PRM algorithm is asymptotically optimal, but computationally expensive.

In order to address the limitations of previously discussed algorithms, new algorithms were proposed in [13], i.e.,PRM*, RRG, and RRT*, and proven to be probabilistically complete, asymptotically optimal, and computationally efficient.

- PRM* is a batch variable-radius PRM, applicable to multiple-query problems, in which the radius is scaled with the number of samples in a way that provably ensures both asymptotic optimality and computational efficiency.
- RRG is an incremental algorithm that builds a connected roadmap, providing similar performance to PRM_ in a single-query setting, and in an anytime fashion (i.e., a first solution is provided quickly, and monotonically improved if more computation time is available).
- The RRT* algorithm is a variant of RRG that incrementally builds a tree, providing anytime solutions, provably converging to an optimal solution, with minimal computational and memory requirements. The details of this method will be found in chapter 3.

It should be remembered that it can not be concluded that there is no solution to the pro blem when complete probabilistic or resolution algorithms fail to find a path within a preset time or resolution.

2.7. Summary

This chapter covers some of the methods that precedes the sampling-based algorithms and contribute in its development. Those approaches have been shown to be impractical as the complexity of the environment increases. The limitations found in the geometric motion planning gave rise to sampling-based methods that are probabilistically complete and efficient in high dimensional C-space.

3.1. Introduction

Rapidly Exploring Random Tree (RRT) has been introduced recently and it is one of the fastest and the most efficient obstacle free path finding algorithm. Despite ensuring probabilistic completeness, finding the most optimal path cannot be guaranteed. Rapidly Exploring Random Tree Star (RRT*), a newly proposed extension of RRT, claims to achieve convergence towards the optimal solution thereby ensuring asymptotic optimality along with probabilistic completeness. Our proposed algorithm is based on this method.

In this chapter we will discuss the RRT* method and the smoothing algorithm which represent the building blocks of our proposed approach used to plan the optimal path for a tethered mobile robot.

3.2. Optimal Rapidly-exploring Random Trees (RRT*)

The RRT* is used as an essential part in our proposed algorithm for planning optimal path for tethered mobile robot. This tree finds an initial path very quickly and then later keeps on optimizing it as the number of samples increases.

3.2.1. Characteristics of RRT*

The RRT* has many features that makes it the most suitable method for path generation of tethered mobile robot. Our selection was based on the following characteristics:

- In contrast to the PRM algorithm, the RRT* algorithm is primarily aimed at single-query applications.
- Compared to PRM* and RRG (Rapidly-exploring Random Graph), a tree structure is economical in memory.
- Although the problem at hand is geometric, the RRT* can be used to solve planning problems with differential constraints.
- The RRT* does not require a search algorithm since the path given from the root to any root is optimal.

- ❖ The RRT* is probabilistically complete, i.e, with enough samples; the probability that it finds an existing solution converges to one.
- ❖ The RRT* algorithm is capable of finding the real optimal solution since no discretization is needed which would result in a resolution optimal solution.

3.2.2. RRT* Methodology

This section introduces important path planning concepts related to RRT* in order to provide a better understanding of this study.

RRT* is an incremental sampling based algorithm which finds an initial path very quickly. The path is later optimized as the execution takes place.

Let *X* define the configuration space in which X_{Obst} is the obstacle region,X_{free} = *X* / X_{obst} is the obstacle-free region, $x_{initial} \in X_{free}$ is the starting point and $x_{goal} \in X_{free}$ is the goal.

The path planning problem consists of finding a feasible path following the system constraints with minimum cost from an initial start x (0) = $x_{initial}$ to the goal $x(1) \in x_{goal}$ such that:

$$x: [0,1] \rightarrow X_{free}\ Where\ x(0) = x_{initial}\ \wedge x(1) = x_{goal}\ \wedge\ x(s) \in X_{free} \forall\ s \in [0,1]$$

While finding the solution, RRT* maintains a tree T= *(V, E)* of vertices *V* sampled from the obstacle-free state space X_{free} and edges *E* that connect these vertices together.

The RRT* algorithm grows a tree structure in the search space and this tree is optimized given a certain cost function. The root of the tree is located at the start point and the tree is grown toward the goal. The algorithm does not stop when a feasible trajectory to the goal is found, but it continues optimizing the tree and the trajectory to the goal. The algorithm consists of a number of sub-functions of which the steering function is the most crucial one. This function is responsible for returning optimal trajectories between states. Next to that a sampling function is needed, as well as a nearest neighbor procedure and a collision checker. For the nearest neighbor procedure a distance metric is needed that represents the length of the trajectories between states in the search space.

This algorithm makes use of a set of procedures which are explained in the next section.

3.2.3. Primitive Procedures

Sampling: It randomly samples a state $x_{rand} \in X_{free}$ from the obstacle-free configuration space.

Distance: This function returns the cost of the path between two states assuming the region between them is obstacle free. The cost is in terms of Euclidean distance.

Nearest Neighbor: The function *Nearest (*T, x_{rand}*)* returns the nearest node from T=(V, E) to x_{rand} in terms of the cost determined by the distance function.

Steer: The function Steer (x_{rand}, $x_{nearest}$) solves for a control input *u:*[0,T] that drives the system from *x(0)*= x_{rand} to *x(T)*= $x_{nearest}$ along the path *x: [0,T]* → *X* giving x_{new} at a distance Δq from $x_{nearest}$ towardsx_{rand} where Δq is the incremental distance.

Collision Check: The function *Obstaclefree (x)* determines whether a path *x*:[0,T] lies in the obstacle-free region X_{free} for all *t=0* to *t=T*.

Near-by Vertices: The function *Near (*T, x_{rand}, *n)* returns the nearby neighboring nodes that lie in a ball of volume (*β(logn/n)*) aroundx_{rand}, where β is a constant that depends on the planner.

Insert node: The function *Insertnode (*x_{parent}, x_{new}, T*)* adds a node x_{new} to *V* in the tree T =*(V, E)* and connects it to an already existing node x_{parent} as its parent, and adds this edge to *E*. A cost is assigned to x_{new} which is equal to the cost of its parent plus the Euclidean cost returned by the Distance function between $x_{new}$and its parent x_{parent}.

Rewire: The function *Rewire (*T,$x_{near,}$$x_{min}$, x_{new}*)* checks if the cost to the nodes in $x_{near,}$ is less through x_{new} as compared to their older costs. If it is for a particular node, its parent x_{parent} is changed to x_{new}[15].

3.2.4. Tree Expansion in RRT*

RRT* constructs multiple short paths randomly organized as tree instead of one long path. It originates tree from initial state $x_{initial}$ to find a path towards goal state x_{goal}. The tree gradually improves with iterations. In each iteration, a sampling process selects a random state say x_{rand} from configuration space X. The random sample x_{rand} is rejected if it lies in X_{obst}. However, if it lies in X_{free} then a nearest node say $x_{nearest}$ is searched in tree T according to a defined metric ρ. If x_{rand}lies in X_{free} and is also accessible to $x_{nearest}$according to predefined step size, then a local planner inserts it in tree by connecting x_{rand} and $x_{nearest}$. Otherwise, planner returns a new node $x_{new}$by using a steering function and adds it in tree by connecting it with $x_{nearest}$. A collision checking process is performed to ensure collision free connection between x_{new} and$x_{nearest}$ .The Node expansion process is illustrated in Fig 3.1.

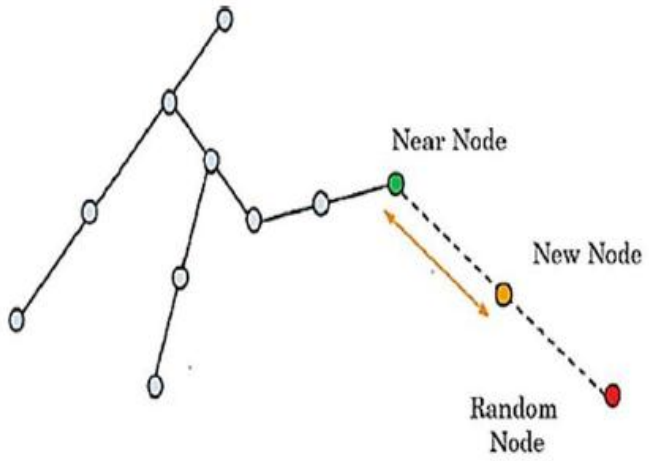


Figure 3.1: RRT* Tree expansion process [16].

If x_{new} is found collision free then near neighbors of x_{new} are searched within the area of a ball of radius defined by

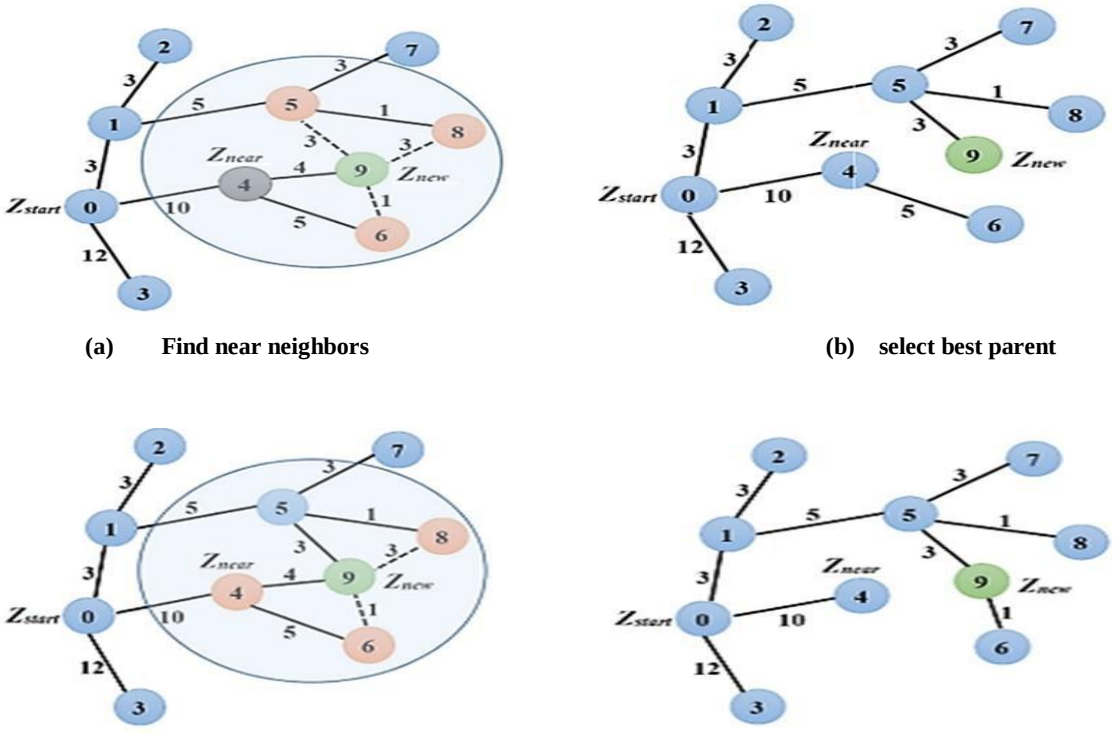
$$k=\gamma\,(\log(n)/n)^{1/d} \quad [13] \qquad (1)$$

where *d* is the configuration space dimension, in our case it is two since the robot has two degrees of freedom(it can translate along the x and y axis) and γ is the planning constant based on environment which can be calculated using the following formula:

$$\gamma= 2(1+\tfrac{1}{d})^{1/d}(\mu(X_{free})/\zeta_d)^{1/d} \quad [13] \qquad (2)$$

Where d is the dimension of the space X, $\mu(X_{free})$ denotes the Lebesgue measure (i.e., volume) of the obstacle-free space but since we are working in 2D; $\mu(X_{free})$ is the area of X_{free} and $\zeta_d$is the volume of the unit ball in the d-dimensional Euclidean space which will be replaced by the area of the unit disk.

Within the area defined by (1), neighbor x_{min} with least cost is selected to be parent of x_{new}. Procedure of near neighbor search is similar to *k*-near neighbor problem to find out the best parent node x_{min} of new node x_{new} before its insertion in tree. New node x_{new} is inserted as child of x_{min} in tree. Further, the cost of near neighbor's parent node is also compared with the cost of x_{new}. If x_{new} gives less cost as parent, then rewiring process rebuilds the tree for minimum parent cost within the area identified by (1). This process is shown in Fig 3.2.



(a) Find near neighbors **(b) select best parent**

(c) Check cost for rewiring **(d) Rewired tree**

Figure 3.2: Near neighbor search and rewiring operations in RRT*[16].

The RRT* is featured by the process of choosing least cost parent and rewiring tree step. These two characteristics contribute in the asymptotic optimality of the RRT* [14]. Though best parent selection and rewiring of tree improve the path quality. However, as the number of nodes in the tree increases ;the convergence is slowed. When x_{goal} is found, a path connecting $x_{initial}$ and x_{goal} is established. This path is improved as planner continues until a predefined number of iterations are executed or given time expires. The RRT* Algorithm is described in the following section.

3.2.5. Algorithm Steps

In this part we will discuss the algorithm steps in details:

Algorithm 1:

The RRT* is initially empty. The first node inserted in the tree is the initial state (line 1-2).

The tree is built and refined through N iterations (lines 3-11). The RRT* algorithm begins in the same way as the RRT. It incrementally builds the tree by sampling a random state x_{rand} from the obstacle-free region (line 4) and solves for a path y_{new} that extends the closest node in the tree $x_{nearest}$ toward the sample (lines 5– 6). If this path is obstacle-free (line 7), instead of selecting the nearest node as the parent, the RRT* selects the set of nodes in the tree that are in the neighborhood of x_{new}(line 8) then the ChooseParent() function (line 9)will select the best parent from the neighborhood of nodes .This is the first major difference between RRT* and the RRT.

Algorithm 2:

This function maintains the node with the lowest total cost for reaching x_{new}.the node that yields the lowest cost becomes the parent node of x_{new} as the new node is added to the tree (Alg 1, line 10).the lowest cost represents the additive combination of the distance associated with reaching the potential parent node and the distance of the path to x_{new}.

Algorithm 3:

The Rewire () function is the second major difference between RRT and RRT*.It changes the tree structure based on the newly inserted node x_{new} . This function uses the nearby neighborhood of nodes,X_{near}, as candidates for rewiring. If the path from x_{new} to the nearby node is obstacle free and the total cost of this path is lower than the current cost to reach x_{near} (line 3). Then the new node x_{new} is a better parent than the current parent of x_{near}. The tree is then rewired to remove the edge to the current parent of x_{near}, and add an edge to make $x_{new}$the parent of x_{near}(line 4).

Algorithm 1: $\boldsymbol{T = (V, E) \leftarrow RRT^{*}(x_{init})}$

1 $T \leftarrow InitializeTree()$
2 $T \leftarrow InsertNode(\emptyset, x_{init}, T)$
3 **for** $i = 1$ *to* $i = N$ **do**
4 $x_{rand} \leftarrow Sample(i);$
5 $x_{nearest} \leftarrow Nearest(T, x_{rand});$
6 $(y_{new}) \leftarrow Steer(x_{nearest}, x_{rand})$
7 **if** $ObstacleFree(y_{new})$ **then**
8 $X_{near} \leftarrow Near(T, x_{new}, |V|)$
9 $x_{min} \leftarrow ChooseParent(X_{near}, x_{neasest}, x_{new}, y_{new});$
10 $T \leftarrow InsertNode(x_{min}, x_{new}, T)$
11 $T \leftarrow ReWire(T, X_{near}, x_{min}, x_{new})$
12 **return** T

Algorithm 2: $x_{min} = ChooseParent(X_{near}, x_{nearest}, x_{new})$

$x_{min} \leftarrow x_{nearest}$;
$c_{min} \leftarrow Cost\ (x_{\text{nearest}}) + c(y_{\text{new}})$;
for $x_{near} \in X_{near}$ **do**
    $(y') \leftarrow Steer(x_{near}, x_{new})$;
    if $ObstacleFree(y')$ **then**
        $c' = Cost\ (x_{\text{near}}) + c(y')$;
        $if\ c' < Cost\ (x_{\text{new}}) and\ c' < c_{\text{min}} then$
            $x_{\text{min}} \leftarrow x_{\text{near}}$;
            $c_{\text{min}} \leftarrow c'$;
return x_{min}

Algorithm 3: $T \leftarrow ReWire(T, X_{near}, x_{min}, x_{new})$

for $x_{near} \in X_{near} \backslash \{X_{min}\}$ **do**
    $(y') \leftarrow Steer(x_{new}, x_{near})$;
    if $ObstacleFree(y') and$
        $Cost\ (x_{\text{new}}) + c(y') < Cost\ (x_{\text{near}})$ **then**
        $T \leftarrow ReConnect(x_{new}, x_{near}, T)$
return T

3.3. Path Smoothing

The shortcutting method will be used in our algorithm to give an improved result of the concatenated paths obtained from the RRT*. The path smoothing is obtained by the following procedure:

Given a path *P*, perform a certain number of iterations. At each iteration, pick two random configurations q_1 and q_2 from *P* and try to connect them by using the local planner. If the new path segment between q_1 and q_2 is valid, i.e., it does not violate any constraints and is shorter than the original path segment in *P*, the original path

segment is replaced by the new one. By repeatedly performing this operation the raw solution path becomes smoother and smoother. The smoothness of the final solution path obtained obviously depends on the maximum number of iterations specified.

A general form of shortcutting is shown in Algorithm4.

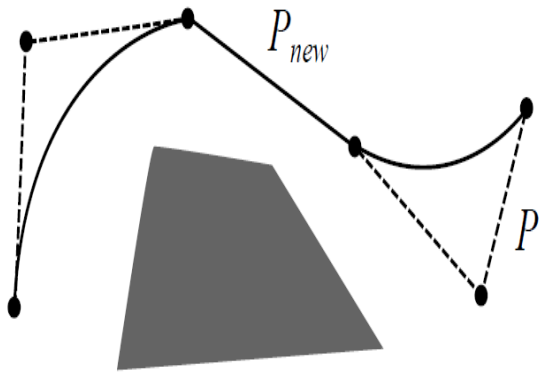


Figure 3.3: Example of the smoothing operation applied to the raw solution path [12].

Algorithm4: Random Shortcut

Input: ***P***: path, ***w***: Number of shortcutting attempts.

Output: A path with potentially shorter length

1. $w' \leftarrow 0$
2. ***while*** $w' < w$ ***do***
3. $\widetilde{ad} \leftarrow P.randomSegment()$
4. $\overline{ad}$ ←straight path connecting endpoints of $\widetilde{ad}$
5. If$|\overline{ad}| < |\widetilde{ad}|$ and if *Obstacle-Free* $(\overline{ad})then$
6. $P.substitute(\overline{ad}, \widetilde{ad})$
7. $w' \leftarrow w' + 1$
8. **return** P

3.4. Proposed algorithm

Our algorithm aims to find the shortest path for a robot cabled at the base, having an initial cable configuration from base to start without violating the cable's length constraint.

The overall path traversed by the robot is the concatenation of the initial shortest path in the cable initial homotopy and the solution path from start to goal generated by RRT*. The solution to the path planning problem is a continuous function such that:

$$P: [0,1] \rightarrow X \; Where \; c(0) = s \; \wedge c(1) = g \; \wedge \; P(t) \in X_{free} \forall \; s \in [0,1$$

We denote a path P concatenated with a path Y as $P \circ Y$; where the start of the path Y is the end of the path P.

The presence of the obstacles in the environment increases the complexity of the problem and gives rise to trajectories that connect the same ending points but differ in the homotopy class. The first building concept of our proposed algorithm is the preservation of the homotopy class meaning that:

- Given a concatenated path $P \circ Y_i$ where P is the path from start to goal and Y_i is the initial shortest path in the cable initial homotopy. The cable configuration denoted as Y_f is the shortest path connecting the base position to the goal position within the same homotopy class as the concatenated path $P \circ Y_i$.

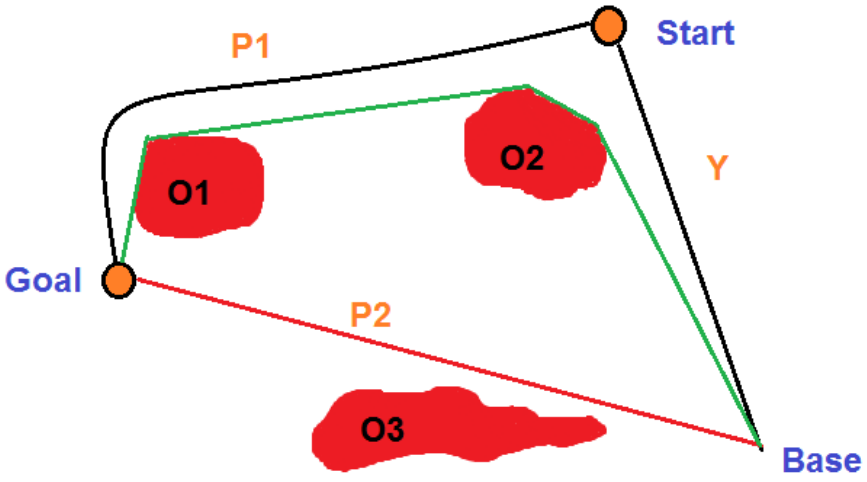


Figure 3.4: Solution to the path planning of a circular robot. Y is the initial cable configuration.P1 is the shortest path from start to goal. The corresponding cable configuration is shown in green. P2 is the shortest path from base to goal.

- Let Random Shortcut (P) be a smoothing function that uses random shortcuts considered to be sufficiently small; i.e.no longer than any workspace obstacle cross section. The smoothing of the concatenated path $P_{b.s}{}^{\circ}P_{s,g}$ gives the shortest path from base to goal which represents the shortest length of the cable from base to goal belonging to the same homotopy.

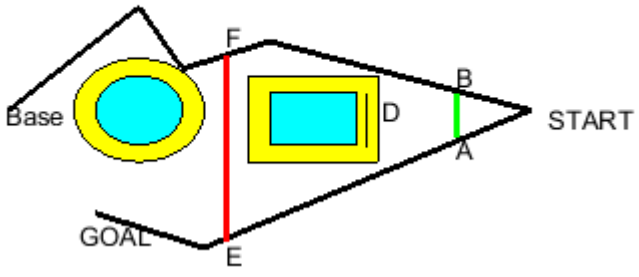


Figure 3.5: Random shortcuts on a path segment. [E-F] represents violation of homotopy class due to the inappropriate choice of the shortcut range while [A-B] illustrates a good shortcut.

- Having Y_i as initial cable configuration and $P_{s.g}$ being the shortest path from start to goal; the goal is reachable via the same homotopy class as $Y_i{}^{\circ}P_{s.g}$

$$if\ |Y_{b,g}| = \left| RandomShortcut\{(Y_i{}^{\circ}P_{s,g})\}\right| < L$$

Where L is the length of the tether.

- If the shortest path connecting the anchor position and the goal position is greater than the length of the tether, then, the problem has no solution. Otherwise, a solution path exists.
- Applying the shortcut method in the configuration space of the robot will lead to an over estimates of the tether and false conclusions on the goal's reachability especially if the robot goes around many obstacles. This is due to the inflation of the obstacles. The shortcut method should be applied on the same path but on the workspace taking into account the obstacles without inflation. Figure 3.6 (a) depicts a smoothed path in the configuration space of the robot; this will result to a configuration of the cable similar to Figure 3.6 (b) which gives an over estimation of the length of the cable. Applying the shortcut in the workspace will result to a configuration of the cable as shown in figure 3.7(a) which will result in a correct estimation of the cable's robot in the workspace as shown in Figure 3.7 (b).

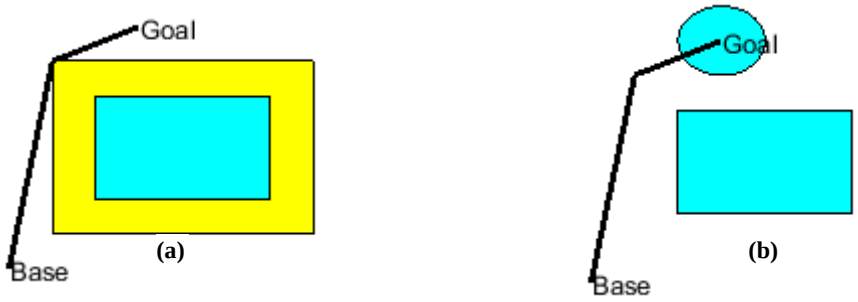


Figure 3.6: Shortcut method using the Robot C space (a) Representation of the tether in the robot C-space (b) Representation of the tether in the workspace.

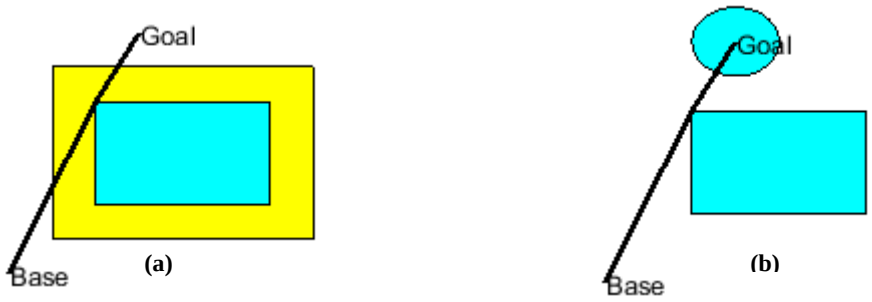


Figure 3.7: Shortcut method using a point like robot in the workspace. (a) Representation of the tether in the robot C-space (b) Representation of the tether in the workspace.

3.4.1. Discussion

The suggested algorithm determines whether a solution to the problem exists through (lines 1-6).The RRT* is kept running long enough then the shortest path connecting the base point to the goal is compared with the length of the cable .The algorithm returns the path calculated from base to goal when the starting point represents the anchor point (8-9).

Algorithm 6 will be executed if the goal can be reached from the base. The shortest path is determined from starting position to the goal which is then concatenated to the cable's configuration.

The length of the new concatenated path is compared with the original length of the cable to check if the goal position is reachable by the robot (line 1-4). In case the condition is not satisfied, the new path is smoothed using random shortcuts with appropriate range to obtain the final cable layout (line 6-11).

In case no solution is determined, the RRT* is run from the goal to determine the shortest distance from each waypoint of the initial cable configuration to the goal. The cost to retract back to each waypoint from the start is also calculated. The waypoint to which the robot will retract back is the one which has the minimum additive combination of the two costs (Algorithm 5 line (13-18)).The RetractBack function in algorithm 7 goes n nodes back from the starting node, removing a portion from the initial configuration of the cable which is appended to the solution path.

The flowchart of the algorithm is depicted in figure 3.8.

Algorithm5: ***The proposed Algorithm***

Input: $b: anchorpoint; s: starting\ position\ ;\ g: goal\ position;\ L: Cablelength$

Output: *A solution if one exists*

$path \leftarrow empty$

$T_b \leftarrow emptyTree$

$T_b \leftarrow RRT^*(b)$

$P_{b,g} \leftarrow T_b.ShortestPath(g)$;

$P_{b,g} \leftarrow RandomShortcut(P_{b,g})$

$Y \leftarrow T_b.ShortestPath(s)$;

if $|length(P_{b,g})| > L$ **then**

  return $NoSolution$

else if $s = b$;

  $path \leftarrow P_{b,g}$

else

  $T_g \leftarrow RRT^*(g)$

  While $!IsGoalReachable\ (s, T_g, Y, L, path)$

    $N \leftarrow number\ path.segments$

    for $i = 1\ to\ i = N$ **do**

      $Cost\ \left(T_g.ShortestPath(Y.segments\ (i))\right)$

      while $Y.parent(s) \sim Y.segments\ (i)$

        Cost RB ←Cost(s,*Y.parent(s)*)

        $s \leftarrow Y.parent(s)$

    $k \leftarrow indx(\min\left(Cost\ (T_g.ShortestPath(Y.segments)) + CostRB\right))$

      n←N-k

    $Y \leftarrow RetractBack(Y, s, n)$

return $path$

Algorithm 6: $\boldsymbol{IsGoalReachable}\ (\boldsymbol{s}, \boldsymbol{T_g}, \boldsymbol{Y}, \boldsymbol{L}, \boldsymbol{path})$

$P_{s,g} \leftarrow T_g.ShortestPath(s);$
if $|Y^{\circ}P_{s,g}| < L$ **then**
  $path \leftarrow path + P_{s,g}$
  $return\ True$
else
  $Y \leftarrow RandomShortcut(Y^{\circ}P_{s,g}, range)$
  if $|Y| < L$ **then**
    $path \leftarrow \overline{path + P_{s,g}}$;
    return $True$
  else
    return false

Algorithm7: $\boldsymbol{RetractBack}(\boldsymbol{Y}, \boldsymbol{s}, \boldsymbol{n})$

$i = 1$
while $(i \leq n$ **and** $(s\ in\ Y))$
$s_{new} \leftarrow Y.Parent(s)$
$Y \leftarrow Y.removechild(snew)$
$s \leftarrow s_{new}$
$return\ path$

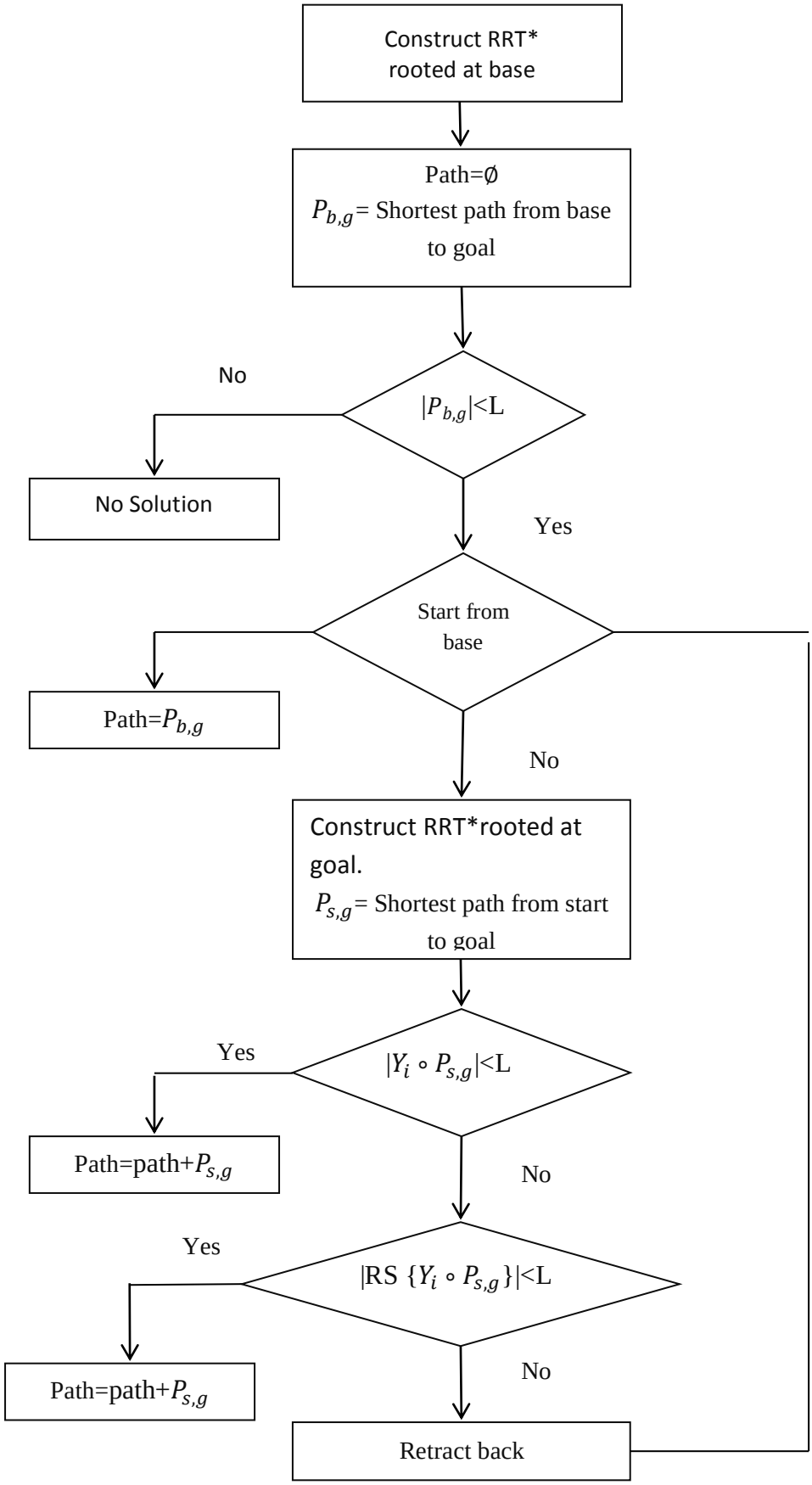


Figure 3.8: flowchart of the proposed algorithm.

3.5. Summary

In this chapter we presented our proposed algorithm to find the shortest path from starting position to goal position using RRT* algorithm. The smoothing of the concatenated path between the initial cable configuration and the path candidate estimates the new cable configuration. In case solution has been found, the robot is allowed to retract back along its waypoints to look for a shorter path.

4.1. Introduction

Using the algorithm described in the previous chapter, different situations dealing with the shortest path planning for tethered mobile robot issue are simulated. Our major contribution consists in using the heuristic presented in section III.4 to estimate the cable configuration when the robot reaches the target.
Various scenarios are simulated using Matlab; the results are shown and discussed in the upcoming sections. The last section treats the results of the anytime motion planning approach.

4.2. SIMULATION

4.2.1. Solution path to different queries

All the computations were performed on a laptop with an Intel Core i3 processor clocked at 2.4 GHz equipped with 4GB of RAM.
The simulation results are displayed in a simple 200×200 discretized environment where both the workspace and the configuration space are considered.
The workspace obstacles are illustrated in dark blue and the robot is modeled as circle of radius 5 depicted in yellow. Whereas the configuration space obstacles are in light blue.

These obstacles represented in the C space are the inflated versions of the original given obstacles, so as to avoid collisions, and thus letting us consider a point model for the robot.

The tree resulting from running the RRT* is represented in green branches inside the reachable region of the robot which is a circle of radius equal to the cable's length and centered at the Base.

The path segments are represented in red lines where:

- The path segment from Base to Start represents the initial cable configuration.
- The path segment from Start to Goal represents the solution path candidate.

The final cable configuration when reaching the goal is shown in Magenta.

L is the given length of the cable and L' is the estimated length of the cable.

This section deals with all feasible instances that may be followed by the robot to find the shortest path to the goal if one exists.

First scenario:

Case 01: Starting at the anchor point with **L=160**

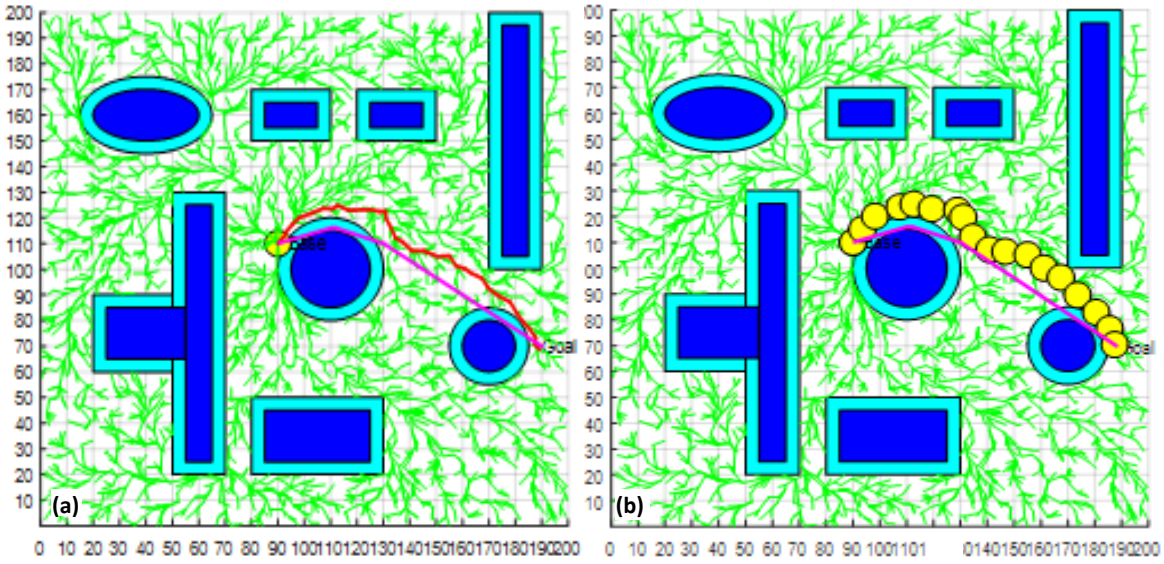


Figure 4.1: The robot starting from the base position a)-path candidate with the estimated tether configuration for a tethered point robot. b)-Motion planning for a tethered circle-shaped mobile robot (depicted in yellow).

The starting position is the base of the tether, then RRT* is launched with 5000nodes. The shortest path from the base point to the goal is generated as in Figure 4.1-a, then the smoothing operation performed on the path using10000 attempts resulting in the cable configuration shown in Magenta with an estimated cable length of L'= **113.9653**<L. This implies that the target is reachable. Hence, the robot moves toward the goal as shown in Figure 4.1-b.The overall algorithm lasts for 252.17 seconds.

Case 02: Starting from a different position than the base with **L=170**

The initial cable configuration is the shortest path from Base to start which the configuration is found from the previous case. Then, the RRT* is launched from the

goal position. The shortest path to the goal is concatenated to the initial cable configuration resulting in a total length of 250.0767. The length of the estimated cable layout found by performing the smoothing is L'=168.9684.

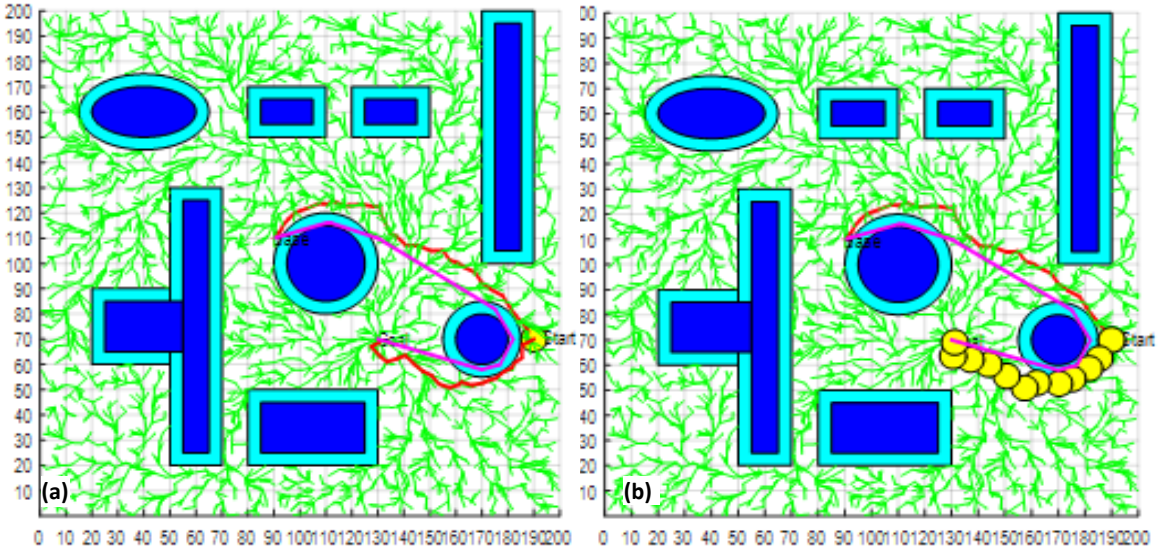


Figure 4.2: The robot starting from a different position than the base a) - initial cable configuration and path candidate with the estimated tether configuration of a point robot. b) - Motion planning for a tethered circle-shaped mobile robot (depicted in yellow).

Since the length of the estimated cable configuration is less than the given cable length L' <L, then the robot can move toward the goal as shown in Figure 4.2-b.The time that has elapsed to display the output for this case is 259.35 seconds.

Case 03: Starting from a different position than the base with **L=160**

Let's assume the same configuration as in the previous case. This implies that the goal is not reachable since L'=168.9684>L. For this situation, retraction of the robot is needed.

The table below depicts the way points constituting the initial cable configuration. The additive combination of the cost from the goal and the retracting back cost of each way point is shown in the third column of the table.

Table 4.1: Way points representing the initial cable configuration and their corresponding costs.

1	90.000; 110.000	3.7875×10^{2}
2	92.9098; 113.5611	3.5502×10^{2}
3	93.4667; 115.0766	3.5301×10^{2}
4	95.8937; 117.9149	3.3730×10^{2}
5	98.2213; 120.1258	3.3318×10^{2}
6	102.8857; 121.8504	3.0648×10^{2}
7	107.3828; 123.7365	3.0311×10^{2}
8	110.7230; 123.4877	3.0002×10^{2}
9	112.6339 ; 124.8468	2.8792×10^{2}
10	116.8599; 122.8976	2.8113×10^{2}
11	119.3648; 123.0213	2.7608×10^{2}
12	124.3432; 123.2864	2.7036×10^{2}
13	128.8372; 122.4286	2.7281×10^{2}
14	130.8796; 122.5391	2.5628×10^{2}
15	130.9855; 119.7635	2.4244×10^{2}
16	132.8063; 116.2361	2.3567×10^{2}
17	134.6039; 112.1026	2.2762×10^{2}
18	137.3083; 110.5841	2.1566×10^{2}
19	140.7947; 107.2141	1.8648×10^{2}
20	145.5456; 107.1043	2.0312×10^{2}
21	146.7813; 106.4900	1.8987×10^{2}
22	150.6053;104.9495	1.8241×10^{2}
23	155.0237; 105.1904	1.8210×10^{2}
24	157.5461; 101.3583	1.7935×10^{2}
25	161.0451; 100.0853	1.8807×10^{2}
26	163.1976; 98.4994	1.7807×10^{2}
27	167.3885; 96.4480	1.8890×10^{2}
28	169.9013; 92.4395	1.7332×10^{2}
29	173.6054; 89.6014	1.5667×10^{2}
30	178.0090; 87.2332	1.7709×10^{2}
31	180.5312; 82.9160	1.5287×10^{2}
32	183.3984; 78.8197	1.6088×10^{2}
33	185.5469; 76.5802	1.5040×10^{2}
34	187.8390;73.0784	1.6756×10^{2}
35	187.4808 ; 70.4441	1.4403×10^{2}
36	189.5291; 68.3875	1.4630×10^{2}

The robot will retract to the point that has minimum cost (shown in yellow).Hence; the robot retracts back to the point [187.4808; 70.4441] as shown in Figure 4.3-a. The length of the concatenated path $P_{b,s}{}^{\circ}P_{s,g}$=235.0785 and the estimated length of the cable after doing the smoothing operation is L'= 76.7773.

Since L'<L; the robot can reach the goal as shown in Figure 4.3-c.

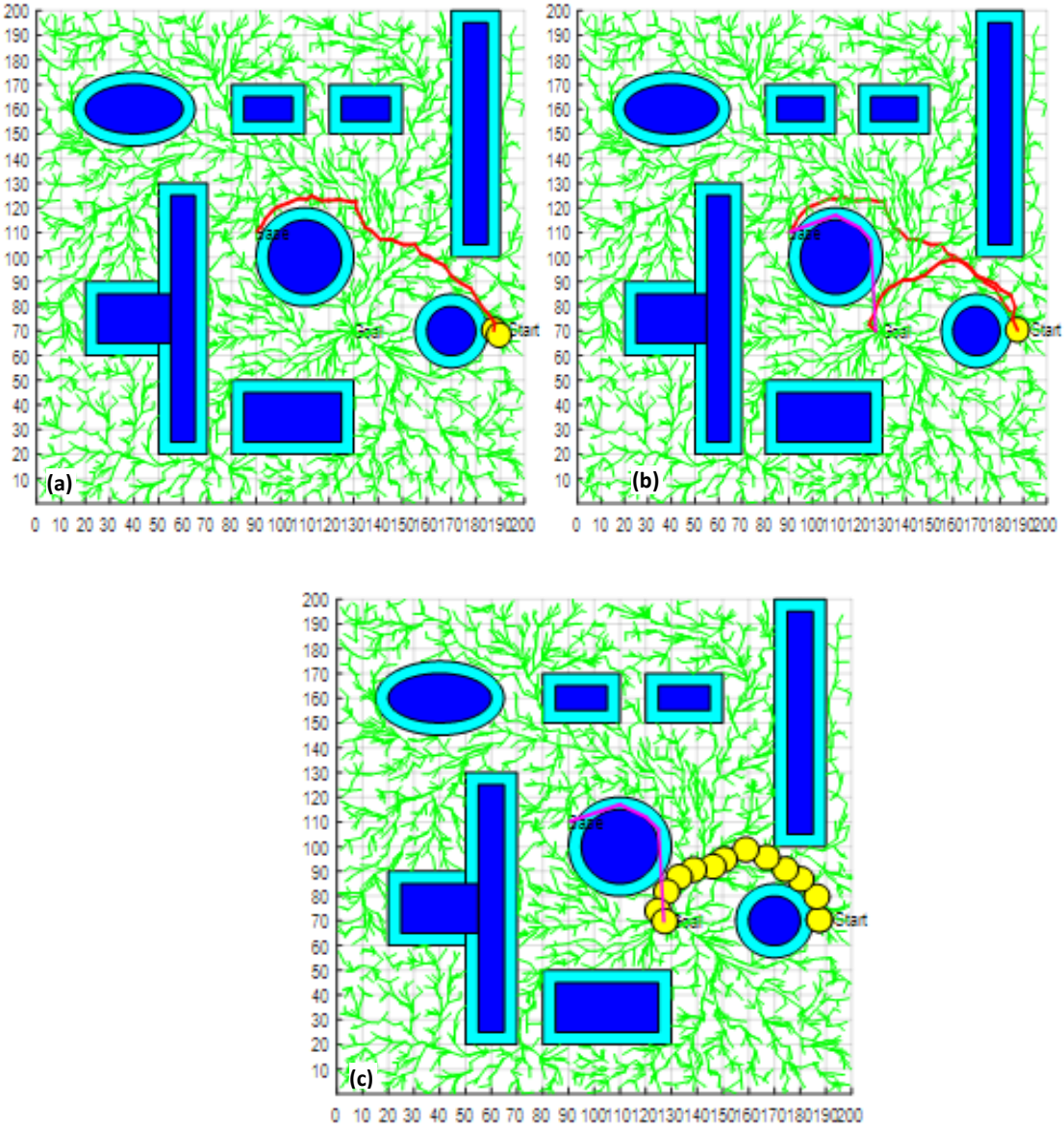


Figure 4.3: Retraction of the robot .a)-Robot retracts back by one way point. b)-the initial cable configuration and the candidate path solution with the estimated cable configuration for a point robot c)- Motion planning for a tethered circle-shaped mobile robot (depicted in yellow).

Second scenario: L=140

In this case, the robot is starting from a different position than the base where this position is not the shortest path from Base to Start. It is an arbitrary cable configuration of length L=131.5839 (Figure 4.4-a).

The RRT* is launched from the goal inside a circle of radius 140 centered at the Base. The length of the shortest path from Start to Goal is 67.3491.The estimated length of the tether is L'=116.1481.The running time of the algorithm is 249.38seconds.

The robot can follow the solution path and reach the target as shown in Figure 4.4-c.

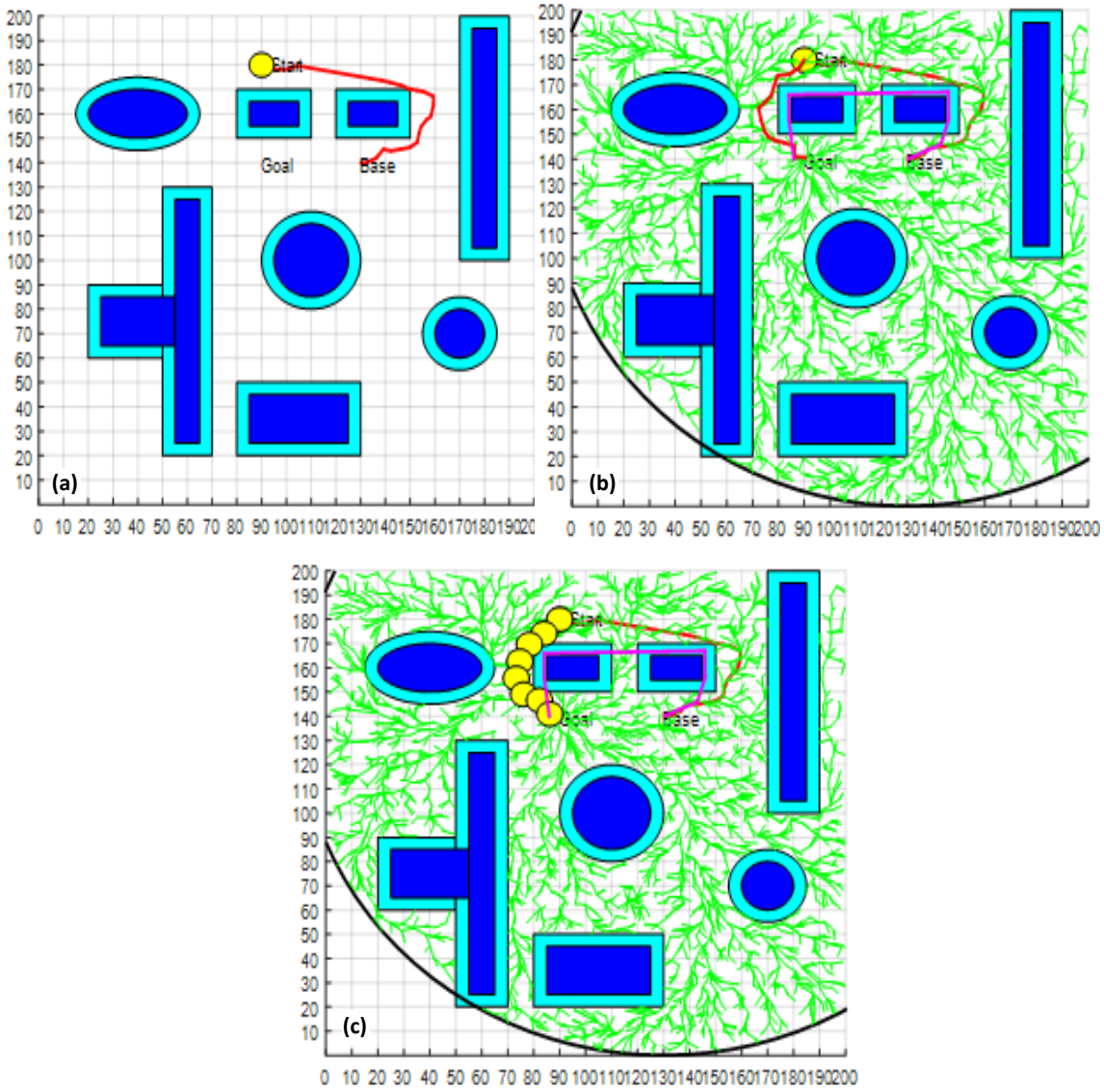


Figure4.4: The robot having an arbitrary initial cable configuration. a)-the robot at the starting position. b)-the path candidate and the estimated tether configuration of a point robot c)- motion planning for a tethered circle-shaped mobile robot(depicted in yellow).

Third scenario: L=190

In this scenario, the robot has an initial cable configuration which is assumed to be the shortest path from Base to Start. The RRT* is launched with 5000 nodes from the goal. The shortest path from goal to start is of length 149.5130 as shown in Figure 4.5-a.The estimated length of the tether is L'=224.8953.

Clearly, the goal cannot be reached .The robot needs to retract back and look for better solution.

The robot should retract to the way point [82.1958; 128.1887] as shown in Figure 4.5-b.

Table 4.2: Way points representing the initial cable configuration and their corresponding costs.

1	10.000; 50.000	3.1917×10^2
2	9.3978; 54.9636	3.2247×10^2
3	9.2220; 59.9605	3.0781×10^2
4	8.9565; 64.9535	3.1983×10^2
5	11.0936; 69.4738	3.2215×10^2
6	10.7481; 73.5021	3.2582×10^2
7	10.6485; 78.1673	3.2628×10^2
8	11.8277; 82.2574	3.2276×10^2
9	13.3273; 84.4548	3.2571×10^2
10	14.7077; 88.4548	3.2233×10^2
11	16.3667; 93.1047	3.1385×10^2
12	20.2798; 96.2173	3.0150×10^2
13	23.4237; 100.1052	2.9661×10^2
14	26.0323; 101.2151	2.9352×10^2
15	30.2048; 103.5182	3.0943×10^2
16	33.2769; 105.4767	3.0715×10^2
17	35.6381; 107.7601	3.1249×10^2
18	36.3526; 112.5053	2.9686×10^2
19	38.9356; 113.8518	3.0986×10^2
20	40.3642; 116.4880	3.1791×10^2
21	43.2363; 120.3796	3.1406×10^2

22	44.9945, 124.2445	3.2480×10^2
23	45.5624; 127.9158	3.1192×10^2
24	47.7306; 131.3137	3.0623×10^2
25	48.0657; 133.2615	3.0714×10^2
26	52.3123; 134.6370	2.9822×10^2
27	57.2180; 135.6034	3.1970×10^2
28	61.5047; 138.1771	3.1142×10^2
29	66.0285; 136.0475	2.9505×10^2
30	70.9539; 135.1870	2.8736×10^2
31	75.2232; 132.7904	2.9115×10^2
32	76.0864; 131.8334	2.9259×10^2
33	80.0550; 131.3032	2.8534×10^2
34	82.1958; 128.1887	2.8505×10^2
35	80.7200; 125.1586	2.9302×10^2
36	81.0854; 119.1186	2.9588×10^2
37	79.8650; 121.3335	2.9661×10^2
38	81.1112 ; 114.5045	2.9653×10^2
39	79.6094; 109.9495	3.1149×10^2
40	80.5754; 107.9077	3.0558×10^2
41	79.8043; 103.6655	3.5052×10^2
42	81.0789; 102.1170	3.4826×10^2
43	78.7475; 98.8809	3.3437×10^2

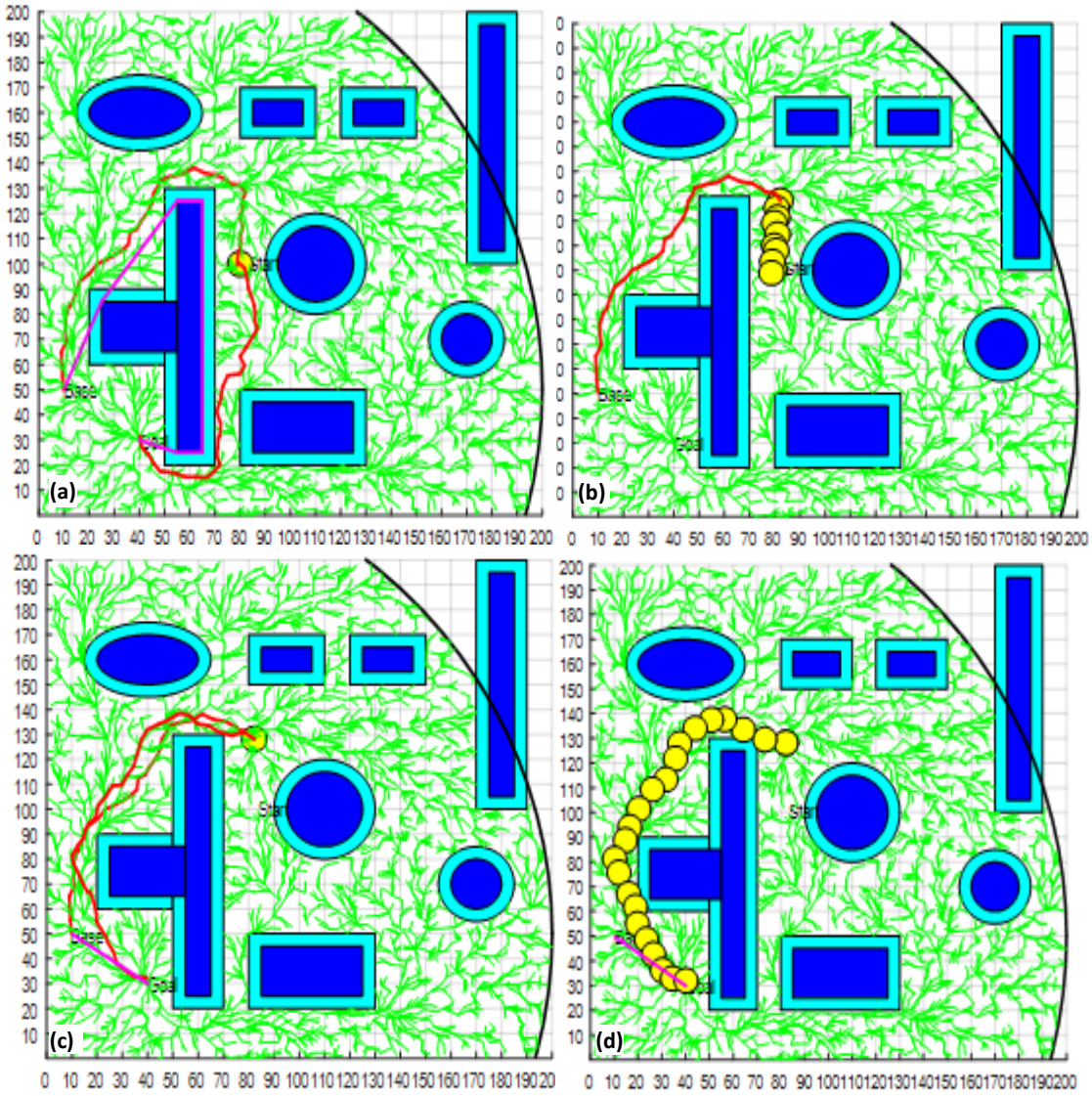


Figure 4.5: Motion planning for a tethered robot. a)-The initial cable configuration concatenated with the path solution and the estimated cable configuration. b)-Retraction of the robot by 9 way points. c)- The new initial cable configuration concatenated with the path solution and the estimated cable configuration. d)-Motion of the tethered circular robot toward the goal.

The robot has retracted back by nine way points to find a new path solution. The length of the shortest path from goal to the new starting position is 158.7603 .The estimated length of the tether is L'= 32.2831.This result costs 269.19 seconds.

Since L' <L, the robot can reach the goal as depicted in Figure 4.5-d.

4.2.2. Anytime motion planning approach

Anytime algorithms are algorithms that quickly find some feasible but not necessarily optimal motion plan, and then incrementally refine it over time toward optimality until time runs out.

Let's consider a point robot having an initial cable configuration of length 71.0349 as depicted in Figure 4.6-a and assume that the cable is of length **L=100.**

Applying the anytime approach to our algorithms results the following:

- The length of the path candidate found after launching the RRT* for 15 s from the goal is 133.7115. The estimated length of the tether is 22.3607as illustrated in Figure 4.6(b).

- The length of the path candidate found after launching the RRT* for 40 s from the goal is 127.6424. The estimated length of the tether is 97.1639 as shown in Figure 4.6(c).
- The length of the path candidate found after launching the RRT* for 150 s from the goal is 106.4016.The estimated length of the tether is 22.3606 as depicted in Figure 4.6(d).

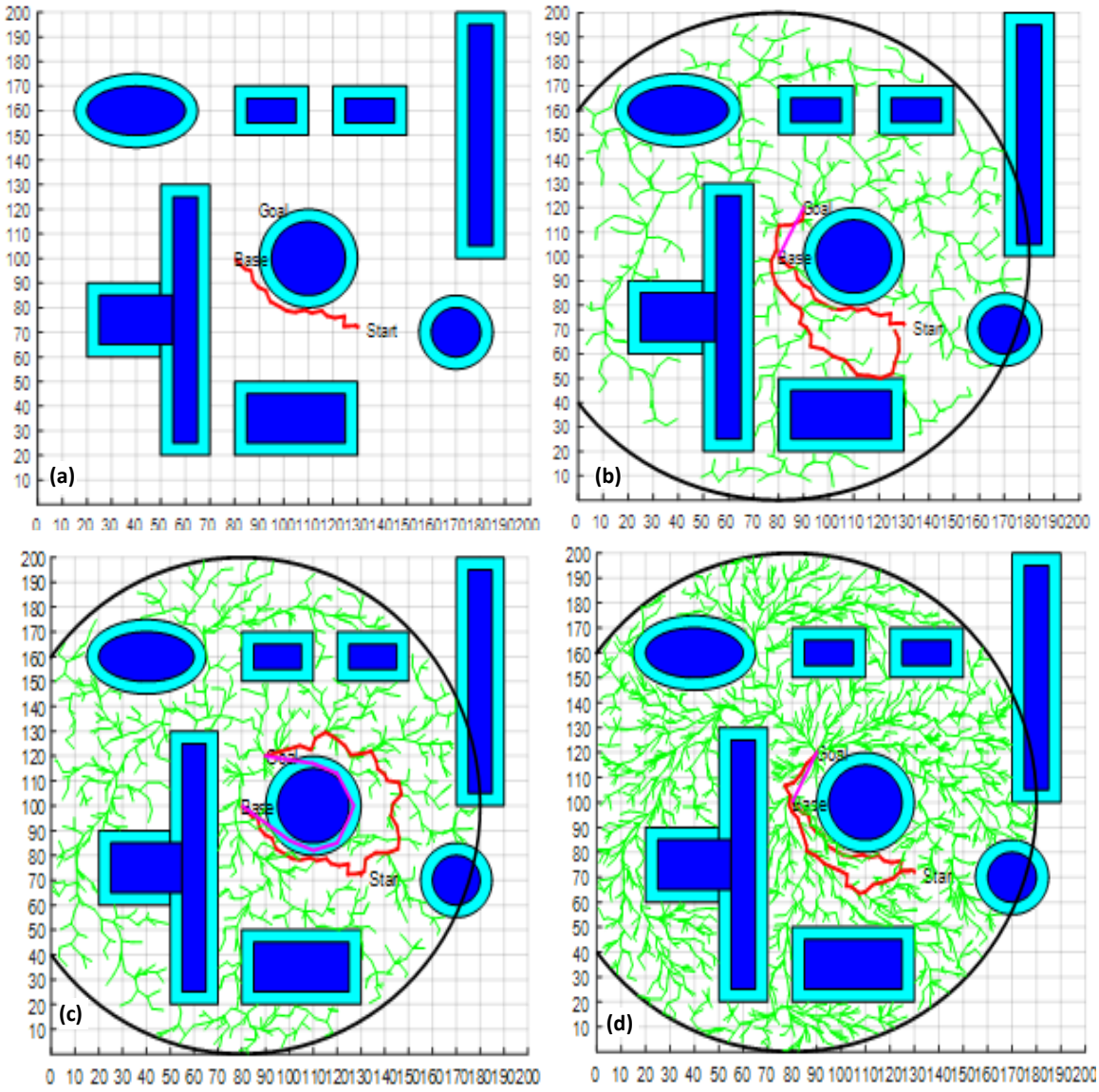


Figure 4.6: Anytime motion planning. a)-The initial cable configuration. b)-The solution path and the estimated tether configuration (15s).c)-The solution path and the estimated tether configuration (40s).d)- The solution path and the estimated tether configuration (150s).

4.3. Discussion

Many important remarks have been drawn from the simulation results:

- Increasing the number of nodes in launching the RRT* may give smoother and more optimized paths. However, it increases the time complexity and slows down the algorithm.

- Launching the RRT* in the reachable region of the robot instead of the whole space enhances the algorithm's performance by pending out the expansion of the tree in regions that exceeds the length of the cable and saving the computational time for these portions.
- Selecting the way point that has the minimum additive cost from the goal position and from the start position among all the way points that constitute the initial cable configuration in the retract back algorithm ameliorate the quality of the solution path and the computation time by avoiding doing successive iterations to retract from the start points which is computationally expensive.
- The shortcutting function has a major role in the validity of our algorithm. Using a shortcutting function with a high shortcut range (resolution) yields to estimations that can violate homotopy classes; thus, giving wrong results. Additionally, a shortcutting function with a small number of trials may provide a path belonging to the same homotopy class with reduced computational time. However, the smoothed path may not be the shortest, resulting in bad conclusions about the reachability of the goal position.
- In anytime probabilistic approach, a feasible solution path is found within the allotted time. However this solution may not be the shortest path. An improved one that tends to optimality can be obtained as the running time is extended.

4.4. Summary

The number of nodes of the RRT* and the inputs to the shortcut function (number of trials, shortcut resolution) affect highly the efficiency of the algorithm. A tradeoff has to be made between accuracy of the conclusion about the solution path and the running time of the algorithm.

General conclusion

1. Concluding remarks

In this work we approached the motion planning problem for a tethered mobile robot with a new perspective. The proposed algorithm is valid for any shape of the robot and not restricted to circular mobile robot. The assumptions that have been made throughout this report are:

- ✓ The cable connected to the base is flexible and retractable of maximum length L.
- ✓ The obstacles are of different shapes.

The objective set is to find the shortest path from the initial robot-cable configuration to a final robot position. Our proposed algorithm solves the problem by determining the shortest path from the starting position to the goal using RRT*.

Our contribution consists mainly of using a good heuristic to represent the cable configuration. This heuristic consists of concatenating the initial configuration of the cable with the path candidate, and then performing the smoothing operation over the concatenated path using sufficiently small random shortcuts to preserve its homotopy class. In case the target is not reachable from the initial position, the robot is allowed to retract back along the way points of the initial cable configuration .A heuristic is also used to test the reachability of the goal through these points without going all the way back to the base or performing many iterations.

The proposed algorithm was executed using simulations in a cluttered environment, and its applicability was illustrated using Matlab. The obtained results were compliant with the objectives set along the project. The simulation demonstrated the validity of our propositions and assumptions related to the conformity of the cable's configuration with the smoothed path obtained from base to goal position. The efficiency of our algorithm in terms of accuracy was shown to be dependent on the quality of the shortcutting function, and the time complexity depends on both the shortcutting function and the number of nodes used in launching the RRT*.

2. Future work

New subjects of concern have arisen during the accomplishment of this thesis, and further improvements will have to be done. The following points emphasize potential objectives to be pursued:

A. Improve the shortcut algorithm

This can be achieved by determining:

- A heuristic for the maximum number of attempts to "short-cut" the path.
- A heuristic for the maximum length of the shortcut to be used.

B. Extension to planning problems with dynamic constraints

Earlier work can be generalized using the Kinodynamic-RRT* for motion planning for robots with linear dynamics, and incorporate nonholomonic constraints to the RRT*.

C. Extending to the 3D-Space:

Extension of the heuristic to represent the cable configuration in a 3D planar environment is another essential goal that requires deeper study to investigate the validity of the cable’s layout in 3D space.

Bibliography

[1] Soonkyum Kim, S. B.," *Path Planning for a Tethered Mobile Robot*". Robotics and Automation(ICRA). IEEE International Conference, May31 2014-June 7 2014.

[2] Syed Ali Ajwad, J. I.*,"Recent Advances and Applications of Tethered Robotic Systems"*. 2014.

[3] Melissa M. Tanner, J. W. "*Online Motion Planning for Tethered Robots in Extreme Terrain*". IEEE International Conference on Robotics and Automation (ICRA).Karlsruhe, Germany, May 6-10, 2013.

[4] Stephen Cass. DARPA Unveils Atlas DRC Robot, Jul 2013.

[5] Li Yibo, X. Q., "*Modeling and PID Control of Tethered Unmanned Quadrator Helicopter*". International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC) . Shenyang, China, Dec 20-22, 2013.

[6] Takeo Igarashi and Mike Stilman. "*Homotopic path planning on manifolds for cabled mobile robots"*. In David Hsu, Volkan Isler, Jean- Claude Latombe, and MingC. Lin, editors, Algorithmic Foundations of Robotics IX, volume 68 of Springer Tracts in Advanced Robotics, pages 1–18. Springer Berlin Heidelberg, 2011.

[7] MILOŠ ŠEDA, "*Roadmap Methods vs. Cell Decomposition in Robot Motion Planning*", Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation, Corfu Island, Greece, February 16-19, 2007.

[8] Oren Salzman and Dan Halperin, "*Optimal motion planning for a tethered robot: Efficient preprocessing for fast shortest paths queries"*, IEEE International Conference on Robotics and Automation (ICRA), Washington State Convention Center, Seattle, Washington, May 26-30, 2015.

[9] P.G. Xavier. "*Shortest path planning for a tethered robot or an anchored cable*." .In Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, volume 2, pages 1011–1017, 1999.

Bibliography

[10] Soonkyum Kim1 and Maxim Likhachev, "*Path Planning for a Tethered Robot using Multi Heuristic A* with Topology-based Heuristics",* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015.

[11] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun, " *Principles of Robot Motion: Theory, Algorithms, and Implementations"*, MIT Press, June 2005.

[12] Felix Burget. "*Whole-Body Motion Planning for Robotic Manipulation of Articulated Objects".* Master Thesis in Artificial Intelligence and Robotics, 2012/13.

[13] Sertac Karaman, Emilio Frazzoli." Sampling-based Algorithms for Optimal Motion Planning". International Journal of Robotics Research, May 2011.

[14] Steven M. LaValle. "*Planning Algorithms*". Published by Cambridge University Press 2006.

[15] Jauwairia Nasir, Fahad Islam, Usman Malik, Yasar Ayaz, Osman Hasan, Mushtaq Khan and Mannan Saeed Muhammad. "*RRT*-SMART: A Rapid Convergence Implementation of RRT* ".International Journal of Advanced Robotic Systems, June 2013.

[16] Noreen, A. Khan, and Z. Habib, "*A comparison of RRT, RRT* and RRT*-smart path planning algorithms*", IJCSNS, vol. 16, pp. 20-27, 2016.