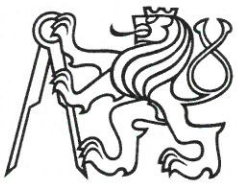
Bachelor Project



Czech
Technical
University
in Prague

F4

Faculty of Nuclear Sciences and Physical Engineering
Department of Cybernetics

My Favourite Thesis; Just the Title is Soooooooo Looooong

Journey to the who-knows-what wondeland

Ronald Krist

Supervisor: Prof. Vojtěch Vonásek
Field of study: Cybernetics and Robotics
Subfield: Control
January 2018

Vzhodní Seznam obrázků

Chapter 1

Introduction

Proteins are macromolecular molecules composed by atomic chains of amino acids. They are essential components of living organisms, for example, over eighty percent of human tissue consists of them. Proteins have numerous functionalities in living organisms. They are basic building unit of many tissues, they function as transport medium for other molecules, they catalyze metabolic chemical processes, they are important for immune system and many more. [strnad, chembook]

The interaction of atoms and molecules during protein folding gives rise to the protein structure, which can be represented in different ways:

Primary structure is given by the order of amino acids in the protein's polypeptide chain.

Secondary structure is determined by local folded structures in the polypeptide chain. (doplnit priklady)

Tertiary structure is the overall geometric shape of the polypeptide which can be described by atomic coordinates.

And finally, quaternary structure is defined for proteins consisting of more than one polypeptide chains. It defines spatial arrangement of their polypeptide chains.

For scope of this thesis, the tertiary, ~~or, if defined, quaternary~~ structure representation of a protein is used. It allows us to represent the molecule as a set of spheres defined by their Van Der Waals radii ~~and their euclidean coordinates~~. This representation can be used as input for the tunnel detection tool, and allows usage of Protein Data Bank repository(PDB).Individual Atom coordinates can be retrieved for most of the structures stored in PDB and at the end of 2014, PDB stored 105 400 structures, which made it the home of majority of experimentally determined structures by the time being. [caver2016]

https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/tutorials/pdbintro.html,

5.11. A visual example of these protein structure representations is given in picture 1.

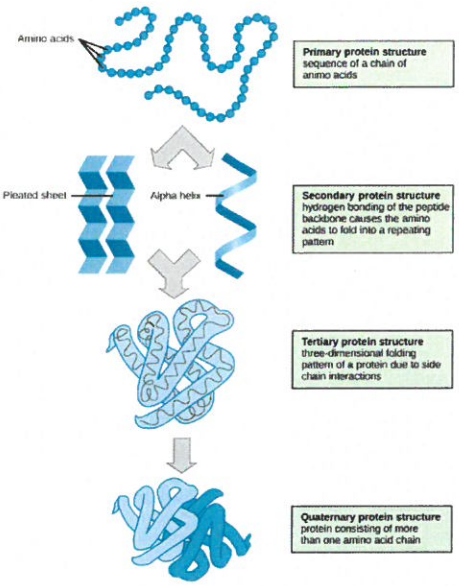


Figure 1.1: Protein structures illustration [https://ka-perseus-images.s3.amazonaws.com/71225d815cafcc09102504abdf4e10927283be98.png]

Proteins are able to interact with surrounding molecules or they can let other molecules such ~~are~~ water, ions or ligands into their structure. These molecules than can reach ~~to~~ so-called active sites or binding sites, where chemical reactions between the small molecule and protein take place. These reactions can be used to change the properties of the small molecule or the protein. Active sites can be positioned on the protein surface, but they are often located in cavities inside the protein structure, therefore studying paths leading to them can give us insights about properties of these proteins and how to modify them to acquire the desired functionality.

Several types of protein ~~cavities~~ can be identified. A tunnel is represented by empty space between protein atoms ~~Van der Waals radii,~~ which forms a pathway between ~~a~~ active site located deep inside the protein and protein's outer environment, called solvent. A small molecule, called ligand, can use this space to get to the active site.

Intramolecular tunnels allow transport of reaction intermediates between two distinct active sites inside the protein.

Pores or channels are pathways leading through the protein molecule without interruption by inner cavity. They transport molecules across biological membranes.

Cavities are empty places inside the protein which are not directly accessible from the protein's surface.

On the other hand, pockets are spots on the protein surface which are easily

accessible by ligand. Both can also contain active sites. Examples of all mentioned structures are in figure 2.

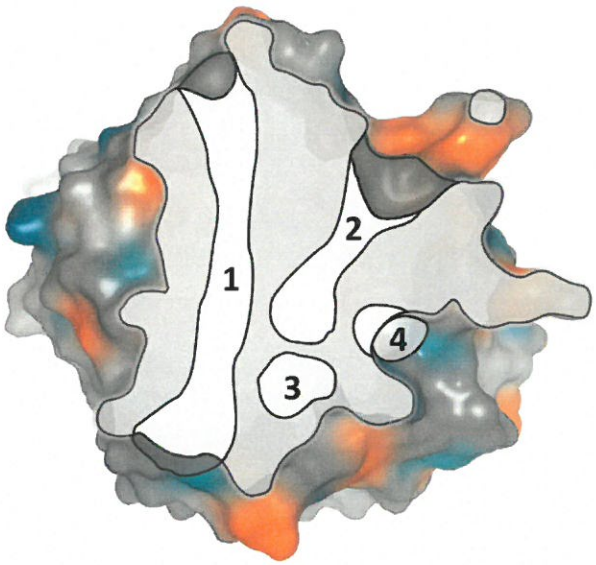


Figure 1.2: Protein cavities illustration [Strnad]

Until now, it was assumed that proteins cavities are time independent, therefore are static properties of the molecule. In real environment, it isn't that simple. Due to intramolecular atomic forces and influence of the molecule's surroundings, protein molecules are constantly changing their inner structure, which leads to constant rearrangement of their inner ~~empty~~ spaces. Study of temporal development of cavities offers substantial advantage in understanding the molecule's behavior, providing biochemists with important insights about specific tunnel's stability in time and therefore it is one of the tasks of this thesis. [Strnad; Stank a spol.]

[e-chembook]

Chapter 2

Problem definition

The task is to find a non-colliding trajectory inside the protein structure for a ligand molecule, which traces the ligand's movement from an active site inside the protein to some place on the protein surface. Using tertiary or quaternary protein structure makes approximation of both ligand molecule and protein structure by a sequence of spheres possible[3]. Formally, tunnel was defined in [zcu thing]. Let the set of spheres representing the protein be denoted by S. Sphere $s \in S$ is defined as $s(c_s, r_s)$, where c_s is a coordinate vector representing the sphere's center and r_s represents sphere's radius. Then we can use distance function which returns negative value for every point in euclidean space, if that particular point lies inside a specific sphere. Such function can be defined as:

$$D(x, a) = \|x - c_i\| - r_i. \tag{2.1}$$

Where x is a point in euclidean space and a is an atom. We can now proceed to define the shortest distance towards nearest atom in the protein structure. Function for that can be defined as follows:

$$r(x) = min\{D(x, a) | a \in S\}. \tag{2.2}$$

Now we can move to formulate the intuitive concept of the tunnel. In [4], tunnel is defined by it's centerline and volume. Centerline a_t is a continuous curve leading from point x inside the protein to point y on the protein surface. Volume is the union of spheres belonging on every point of the centerline with radius equal to the distance towards the closest sphere. Formally, the tunnel T can be defined as:

$$T = \bigcup_{x \in a_t} s(x, r(x)) \tag{2.3}$$

Illustration of the definition is given in Figure 1

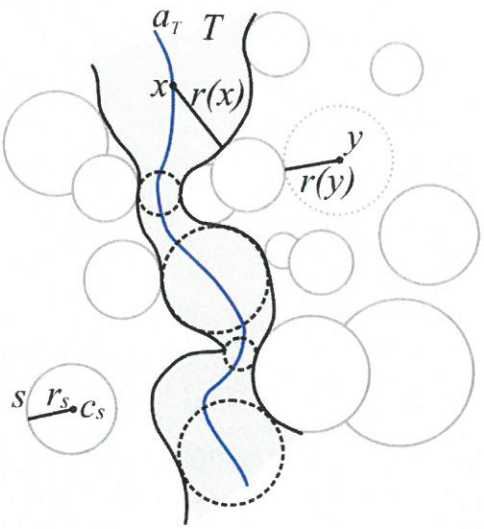


Figure 2.1: Illustration of tunnel definition, taken from [zcu thing]

Protein molecule usually contains more than one tunnel, therefore the output of the algorithm should be set S of unique tunnels. It is assumed that algorithm for tunnel detection should try to find most of them. Tunnels found in one static snapshot of the molecule are referred as static tunnels, therefore, for static molecule, the output of the algorithm should be set S of unique tunnels.
As stated before, the algorithm should detect tunnels in ensemble of molecular dynamics. In the scope of this thesis, molecular dynamics are represented by a series of static snapshots ((taken in very short time intervals, in the orders of ?)) of the molecular structure. Set of tunnels found in one snapshot doesn't give away any information about how the tunnels behave, once the molecule leave the snapshot configuration.

Chapter 3

state-of-the-art

Next step after main thesis' objective definition is review of the state-of-the-art tools used. In recent years, numerous new tools and methods for tunnel detection and classification have been developed[6]. Typical approaches forming core of those tools include grid based methods and using the voronoi diagram.

3.1 grid methods

Methods based on grid approach work by bounding the entire protein by a boundary box and rasterizing the bounded space using regularly distributed grid in three dimensional space. Modeling ~~the protein's molecules with~~ spheres allows checking grid nodes for collision. Grid composed of non-colliding nodes than forms a graph, in which paths can be found using any standard graph-traversing algorithm, such as the Dijkstra's algorithm or A*.[Jankovec, Byska] Tools representing this approach are HOLLOW 1.2 [5], 3V 1.0 [27] or CAVER 1.0 [23] [Byska]

Major disadvantages coming with grid based approaches are dependency of accuracy on the resolution of the grid and The CPU and memory requirements can limit grid methods only to smaller systems [Brezovsky].

3.2 voronoi diagram based methods

To overcome the major disadvantages of grid based methods, approach based on voronoi diagrams was developed [Strnad]. Voronoi diagram is a specific geometric structure created by splitting the n-plane into convex shapes so that every point inside the shape S is closer to the point generating shape S than to any other point in the plane. Formally: ((add))[Jankovec, ??] In practice, Voronoi diagram construction exploits it's duality with Delaunay triangulation, which is constructed from the set of points denoting centers of molecule's atoms. A weighted graph is constructed from the Voronoi diagram by setting the graph's nodes to Voronoi vertices. Edge of the graph is created if two Voronoi vertices are connected. Weight of the edge is set by an evaluation function, which can take into account the distance to nearest atom and length of the edge. The resulting graph is then again traversed by graph search algorithm. Complications arise from difference between Van der Waals radii of individual atoms in the structure, causing the distances in the resulting diagram to be distorted when constructed by above described way. This error is either considered insignificant enough and ignored (Mole)[Brezovsky], or approximation using a number of smaller spheres is implemented (Caver, Molaxis).

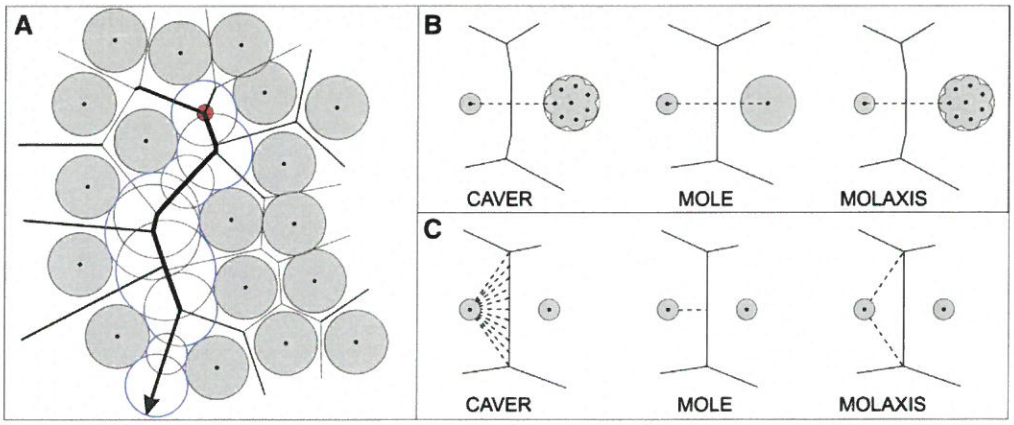


Figure 3.1: Visual comparison of different tools based on voronoi diagrams
Figure A shows tunnel found in two dimensional environment found using the voronoi diagram.
Figure B presents how compared tools deal with discussed difficulty from variable atom sizes.
Figure C illustrates principles according to which the edge price is computed.
taken from [Brezovsky]

Molecular dynamics are can be addressed (Caver, Mole) by clustering algorithms. Tunnels from different snapshots are assembled into clusters by some function evaluating their similarity. Mole takes into account atoms lining the

tunnel, Molaxis uses split distance parameter limiting the distance from the last common node. [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2447770/] Unlike grid methods, voronoi diagram based methods don't depend on resolution of the grid, the diagram is computed straightly from placement of atom Van der Waals spheres.

Chapter 4

Sampling methods

4.1 Introduction

Motion planning is a term most commonly used in robotics. In robotics motion planning involves a task of getting a robot to automatically determine how to move while avoiding collisions with obstacles.
The basic path planning problem is summarized as: Given an initial placement of the robot, compute how to gradually move it into a desired goal placement so that it never touches the obstacle region. Consider the task in terms of algorithm inputs and output.
The inputs consist of an initial placement of the robot, a desired goal placement, and a geometric description of the robot and obstacle region.
The output is a precise description of how to move the robot gradually from its initial placement to the goal placement while never touching the obstacle region.
The output description will be a path through the set of all intermediate transformations of the robot, from start to finish. [2]
It is possible to describe the problem of tunnel detection from this perspective. The obstacle region is defined as the set of balls representing the protein molecule, the ligand molecule is represented as the robot for which the motion is planned and the goal is defined as a location lying outside of the molecule. With this definition, after some modifications, one of the well-established algorithms can be used to find trajectory representing the tunnel.
In general case the ligand molecule has six degrees of freedom. In the scope of this thesis only spherical ligands are assumed, therefore only three degrees of freedom are needed, leaving out the rotations of the ligand molecule. Let

the configuration space C be space of all available spatial positions in three dimensional space inside the boundary enclosing the protein molecule. Single configuration is then defined as $q \in \mathbb{R}^3$. Let $P \in \mathbb{R}^3$ be the set of atoms representing the protein, therefore the C_{obs}, then the free space can be defined:

$$C_{free} = \{q \in C | C \bigcap S(q) \bigcap P \varnothing\} \tag{4.1}$$

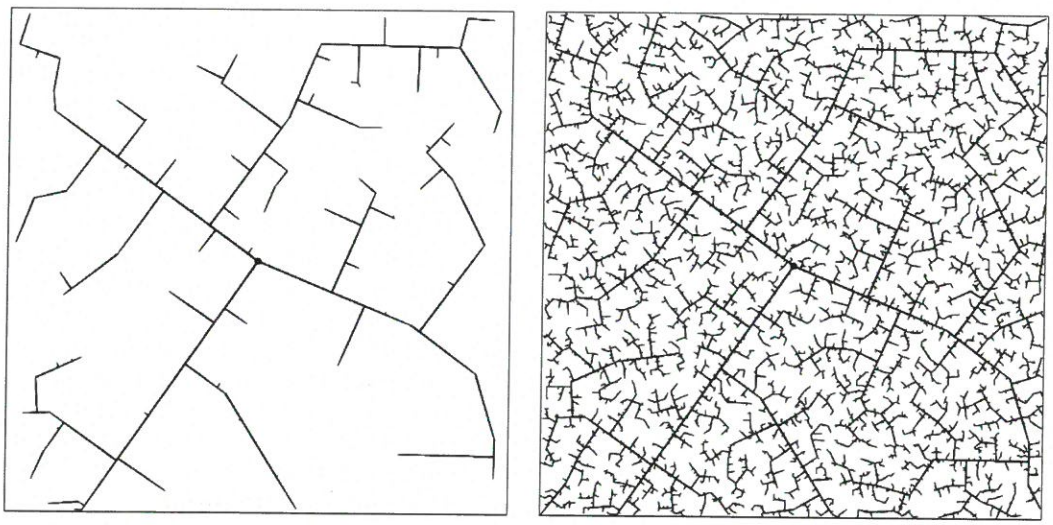
where S(q) represents sphere of some radius q in location defined by coordinates of q. Let C_{out} represent the part of configuration space which is considered to be outside of protein molecule. Then the goal region is equal to C_{out}. The concrete definition of this region is left to the needs of the specific implementation of the algorithm.
Main approaches used for solving motion planning problems are based on combinatorics and sampling. Main idea of sampling based methods is to avoid explicit construction of obstacle configuration space. This idea offers important advantage in problems with thousands or even millions of geometric primitives, such problems would be practically impossible to solve with methods involving explicit construction of obstacle configuration space. [Lavalle article - 1]
One of the algorithms based on sampling is the RRT (rapidly exploring random tree) algorithm. The It's main idea is taking random samples from configuration space and connecting them to a tree structure until a path to place in some arbitrary distance from goal is found.
Let's assume that C is a metric space. RRT is a topological graph, $G(E, V)$. The exploration algorithm begins by initiating the first vertex at start position, q_{start}. Then, for K iterations, C is randomly sampled according to some sampling strategy. When new sample q_{rand} is produced, the algorithm tries to connect it to it's closest place in the tree, q_{near}. That means the closest place, as defined by metric used by the specific implementation, to tree's swath S defined as:

$$S = \bigcup_{e \in E} e([0, 1]) \tag{4.2}$$

,where $e([0, 1])$ is the image of the edge E. The path to closest point is checked for collision by vertex interpolating function extend q_{near} towards q_{rand} using some collision detection function and connected to the closest point from the first point, after which the path to the closest point is considered as collision free. The closest point can be found analytically or an approximation by creating intermediate vertices along tree's line segments and searching for the nearest vertice instead can be used. The pseudocode is given in algorithm 1 and illustration of trees expansion in figure (()).

Algorithm 1 Original RRT

```
1: T.add(q_start)
2: while iteration < K do
3:     q_rand = random configuration
4:     q_near = nearest neighbor in tree T to q_rand
5:     q_new = extend qnear toward q_rand
6:     if q_new can connect to q_near then
7:         T.addVertex(q_new)
8:         T.addEdge(q_near, q_new)
9:     end if
10:    if ϱ(q_new, q_goal) < distanceToGoalTh then
11:        break
12:    end if
13: end while
```



45 iterations 2345 iterations

Figure 4.1: Illustration of the rrt expansion, taken from [Lavalle book]

RRT

4.2 Differences

! Lepší a vyšší žánr jsou malzel !

Unlike methods based on geometry used in mentioned standard approaches, sampling-based motion planning algorithms are incomplete. That means, it is not assured that the algorithm will decide in finite time whether a solution exists. They instead rely on notion of probabilistic completeness, meaning that the probability of finding a solution converges to one with enough samples, given that the solution exists, but they can run forever if

solution doesn't exist. [Lavalle, book]
Usage of random sampling makes random sampling based methods nondeterministic, therefore different outcome is expected from individual runs of a tool based on them.
From functional perspective, the nature of motion planning paradigm offers two important distinctions:
Firstly, they were developed for planning motion of robots with many degrees of freedom. That means they are naturally extensible for planning with non-spherical ligand composed from more atoms.
Secondly, by using four dimensional configuration space, $\mathbb{R}^3 \times Time$ they allow direct search in molecular dynamics without need for clustering between tunnels from individual snapshots.

4.2.1 branch collision effect

All of the mentioned state-of-the-art methods form a graph, which is then traversed by a graph search algorithm. However, motion planning methods can use a tree structure instead. Tree structure prevent detection of some valid tunnels due to what we call the branch collision effect. Illustration is the best way to explain it, so see figure

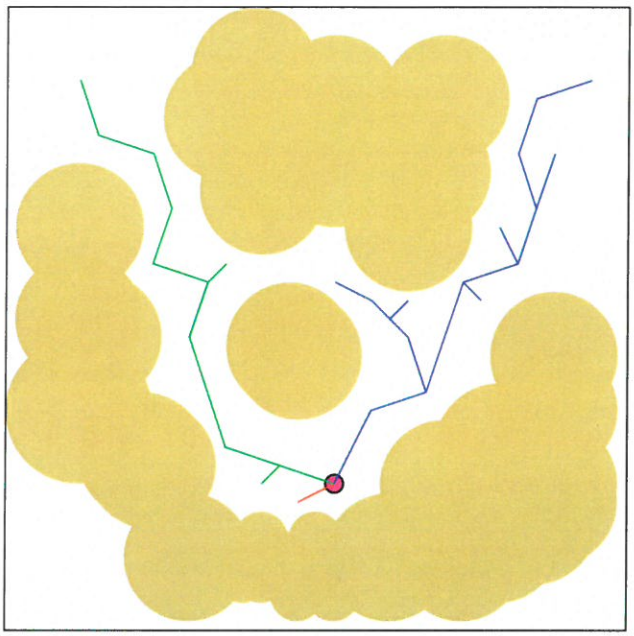


Figure 4.2: Illustration of the branch collision effect
We see a tunnel that a tree based method with already developed tree at this state will probably not detect. Joining the green and the blue branch is not possible, it would invalidate the tree structure. Due to properties of nearest neighbor search, the possibility of one branch developing into tunnel already occupied by another branch is very low.

Chapter 5

Algorithm description

5.1 Overview

The goal of this thesis is providing algorithmic solution for our task based on the original RRT algorithm. The original RRT needs extensions such as search for multiple tunnels, processing important information about individual tunnels and support for search in molecular dynamics. Algorithm presented is inspired by the TOM-RRT algorithm described in [Jankovec]. TOM-RRT offers required functionality for detecting protein tunnels in static environment, therefore our main task is to extend it's functionality with the ability to detect protein tunnels in dynamic environment. This algorithm was chosen because it already offers RRT based solution for some needed functionality.
Our tool was implemented from scratch, testing various different approaches. The reason is to test some modifications of the original version, which could perform better for our purpose.

The basic outline is described in

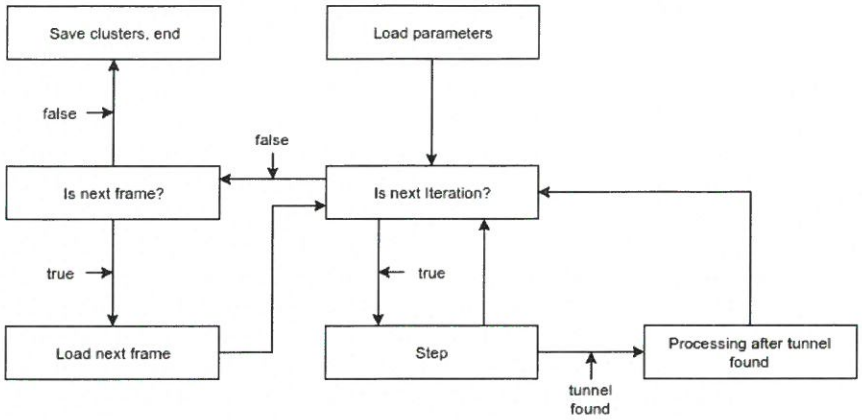


Figure 5.1: Algorithm flow chart

The step function tries to connect new sample and checks whether the goal region was reached. If the check was successful, it triggers optimization procedures ((todo - rozsirit))

5.2 sampling region

Sampling region is the area from which the random samples are taken. It is necessary for it to stretch over the whole protein molecule and some space around it. If not, we can end in situation where the tunnel endpoint is missing because none of it's potential locations lie in sampling region.

The sampling region in our solution is found automatically by function analyzing atom coordinates in pdb files and locating atoms lying on the edges of the molecule. They are selected by extreme values of their X, Y and Z euclidean coordinates.

Two approaches for sampling probability distribution were tested. One is simple uniform distribution. The second one is based on biasing the sampling area to use more samples from inside the protein molecule than the ones from outside.

The main idea of this approach is slowing down the tree's expansion towards outer environment and instead allowing it to grow more densely inside the molecule. That could allow more time to be spent on developing the tree in more difficultly reachable regions rather than reaching to new tunnels in already explored parts, which could easily be duplicates of the ones already found there.

The sampling function tries to put the testing sphere inside the generated sample and if it doesn't collide, the sample is marked as lying in outer environment. Parameter inside_sampling_bias is used for quantification of this bias. It describes the percentage of samples required to lie inside the protein. For every sample, a random number in interval(0, 1) is generated. If it lies under the threshold defined by the parameter, the sample is required to lie inside the protein. If it doesn't, another sample is created. If the random number crosses the threshold, the sample can lie anywhere in the configuration space.

Testing revealed that increase in parameter value leads to larger count of found tunnels and longer computing times. Experimental data are presented in chapter six. Generally, while the increase in computing times remains the same, the increase in number of found tunnels is less significant for larger parameter value. After experiments, the default value was set to 0.75. User can use whichever value he/she wants to by editing the config file of the application.

5.3 Connecting samples

For joining to the closest point in tree, approximation with nearest neighbor search is used. Tree edges are interpolated to the resolution specified by parameter interpolation_step.
After a sample is created, it is moved in a straight line so it's distance from it's nearest neighbor is equal to parameter max_step_distance. If the distance is shorter, the sample is left in it's place. Then we position the probe incrementally towards the nearest neighbor. We move the probe on the straight line by interpolation_step distance and check for collision until the distance to the nearest neighbor is shorter than interpolation_step. The tree edge begins on the first place after which the probe doesn't collide. Tree node is added at the edge's beginning and at every spot after it, which was checked for collision. For visualization of this procedure check figure(()). Pseudocode for connecting samples is given in algorithm 3 and for interpolation of sample in algorithm 4. Algorithm 4 practically only checks for collision interpolated coordinates from the sample to the nearest neighbor, adds them to temporary container and saves container index for the first valid node, than goes from the nearest neighbor to first valid node and adds nodes to tree.

Algorithm 2 connectSample

```
1: parameters ⟵ readParameters()
2: counter ⟵ 0
3: while isNextFrame() do
4:     while counter < parameters.iterations do
5:         sample ⟵ createSample()
6:         step(sample)
7:         counter++
8:     end while
9: end while
```

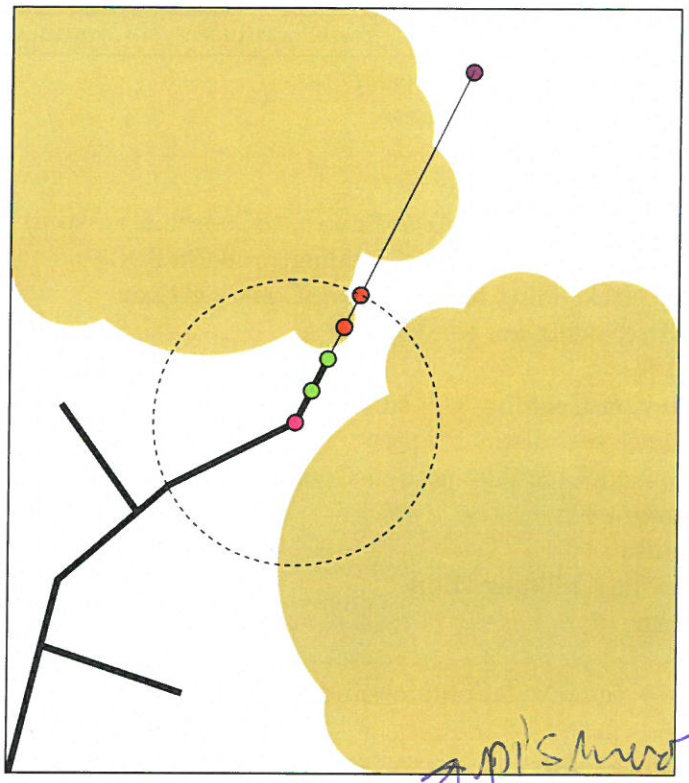


Figure 5.2: Illustration of new sample connection
Purple node represents sample, red nodes denote interpolation steps, which are not added to the tree. Green nodes represent interpolation steps, which are added to the tree. Fuchsia node represents the nearest neighbor. The dotted circle represents max_step_distance. Current node is added to the tree if no node closer to the nearest neighbor collides.

Algorithm 3 connectSample

```
1: distance ⟵ computeDistance(sample, nearestNeighbor)
2: if distance > maxStepDistance then
3:     moveNodeToAnother(sample, nearestNeighbor, distance - maxStepDistance)
4: end if
5: return interpolateSegment(sample, nearestNeighbor)
```

Algorithm 4 interpolateSegment(fromCoordinates, toCoordinates, distance)

```
1: curCoordinates ⟵ fromCoordinates
2: isPreviousColliding ⟵ true
3: counter ⟵ 0
4: while distance > step do
5:     moveCoordinates(curCoordinates, toCoordinates, step)
6:     isCurColliding ⟵ cur.isColliding(probeRadius)
7:     if !isCurColliding && isPreviousColliding then
8:         firstNodeIndex ⟵ counter
9:     end if
10:    isPreviousColliding ⟵ isCurColliding
11:    distance ⟵ distance - step
12:    tempNodeContainer.add(curCoordinates)
13:    counter++
14: end while
15: if isPreviousColliding then
16:    return
17: end if
18: index ⟵ tempNodeContainer.indexAtLastElement
19: break ⟵ false
20: while !break do
21:    curCoordinates ⟵ tempNodeContainer.get(index)
22:    addNodeToTree(curCoordiantes)
23:    if areInGoalRegion(coordinates) then
24:        afterTunnelFoundProcedure()
25:        return
26:    end if
27:    if index == firstNodeIndex then
28:        break ⟵ true
29:    end if
30:    index−−
31: end while
```

5.4 probe

The so-called probe serves the function of the robot in motion planning task. The probe in our tool is limited to a spherical object with user defined radius. This approach allows detection of tunnels with specified minimal bottleneck.