Iterative solution for the narrow passage problem in motion planning

Jakub Szkandera[1] and Ivana Kolingerová[1]

Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Univerzitni 8, CZ 30614 Plzen, Czech Republic.
szkander@kiv.zcu.cz, kolinger@kiv.zcu.cz

Keywords: Motion planning, Sample-based algorithms, Rapidly exploring random tree, Narrow passage, Bottleneck, Binary search

Abstract. Finding a path in a narrow passage is a bottleneck for randomised sampling-based motion planning methods. This paper introduces a technique that solves this problem. The main inspiration was the method of exit areas for cavities in protein models, but the proposed solution can also be used in another context. For data with narrow passages, the proposed method finds passageways for which sampling-based methods are not sufficient, or provides information that a collision-free path does not exist. With such information, it is possible to quit the motion planning computation if no solution exists and its further search would be a loss of time. Otherwise, the method continues to sample the space with sampling-based method (a RRT algorithm) until a solution is found or the maximum number of iterations is reached. The method was tested on real biomolecular data - *dcp* protein - and on artificial data (to show the superiority of the proposed solution on better-imagined data) with positive results.

1 Introduction

Motion planning (finding a collision-free path for a moving object between at least two locations in an obstacle-filled environment) has always been one of the essential research areas. Now it is coming to the forefront even more due to the rapid development of applications, such as the control of autonomous vehicles and sophisticated robots. Fast and reliable solutions in real or near-real time are necessary to cope with such applications. Geometric methods such as Voronoi diagrams for computing centerlines can be used for moving objects (generally referred to as agents) of very simple shapes. However, these methods are inappropriate if the object is complex or even flexible or if the environment is complicated or even dynamically changing. In this case, more sophisticated and general methods should be used.

Motion planning is generally interpreted using the concept of configuration space, which is the set of all existing configurations, where each configuration represents a unique position and rotation of the navigated object. The combination of position and rotation is called the degrees of freedom. The dimension of

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16

the problem and its associated complexity increases with the number of degrees of freedom. For example, the configuration of a navigated object may be up to a 6-dimensional vector describing its position and rotation (both properties have 3 vector components) in 3D space. The total number of configurations in the configuration space is huge. It is impossible to process them all in a reasonable time, so randomised sampling-based methods are used to select and process specific configurations.

Methods based on random sampling [14] [17] randomly generate or select configurations to be tested whether they are collision-free. If the generated configuration is in collision with the environment, it is rejected, and the method generates a new random configuration. The non-colliding configuration is added to the path-finding structure - the so-called roadmap that approximates the configuration space's free regions. Graph-based path planning methods [10] [15] can then be applied to the roadmap. Using roadmaps is an efficient way to find a collision-free passage through the environment, usually in a reasonable (often near-real) time. As already mentioned, the biggest pitfall of the random sampling-based algorithms is the narrow passage problem, as, by random sampling, it is difficult to hit such a place properly.

The proposed solution to the narrow passage problem is an improvement of our previous solution [22], based on a combination of Voronoi diagrams, binary search, randomisation, and the idea of so-called exit areas [19], an instrument for motion planning in protein molecular models. Our previous solution still had the pitfall caused by the randomisation, so it was impossible to decide whether a collision-free path existed or not. The solution proposed in this paper removes this pitfall by a divide and conquer approach. The decision has almost 100% certainty, with the only limitation being the accuracy of the computer representation of real numbers.

The structure of the article is as follows. A description of ideas behind the existing motion planning methods that are widely used for agent navigation in configuration space, and their various modifications to address specific problems, is given in Section 2. A detailed description of the idea behind the proposed solution and the solution itself for motion planning, including a description of the algorithm that unambiguously determines whether or not a solution exists in the narrow passage, is in Section 3. Section 4 contains experiments and results on artificially generated and real bio-molecular data. Section 5 concludes the paper.

2 Related Work

The motion planning algorithms based on the randomized sampling of the environment can be divided into two groups. The former group, Rapidly Exploring Random Tree (RRT) [17], produces a tree structure representing the collision-free part of the environment. The latter, Probabilistic Roadmaps (PRM) [14], creates a graph structure representing the collision-free part of the environment. The idea of PRM algorithms [9] is to generate a set of random samples, which are

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16

simultaneously tested for collisions (sample in a collision is removed, a collision-free sample is preserved). Next, if the edge does not pass through an environmental obstacle, the closest collision-free samples are connected by an edge. Finally, we use one of the graph path planning algorithms such as A~ [10] or the more complex D~ Lite [15] to find the passage through the environment. The set of obstacles can be sufficient input for a PRM-based algorithm since the algorithm itself does not require knowledge of the initial and final configurations. Still, their knowledge can be used in some heuristics to refine the sampling.

A simplified version of the Probabilistic Roadmap algorithm called sPRM [14] is used mainly for the analysis of follow-up algorithms. Its additional advantage is that it finds the asymptotically optimal path. The sPRM is also the basis of the PRM* [13] algorithm, which uses a heuristic function to minimise the length of roadmaps, where potential samples to connect are selected from neighbourhoods with radii $r > 0$, which can be defined, e.g., as a function of sample dispersion[17]. Other PRM-based methods attempt to relax the collision constraints [11] [2], leading to a simplification of the overall complexity of the problem, by reducing the agent volume, e.g., by making the agent thinner [11] or by scaling down the size of agent [2]. The resulting found roadmap is only an approximate solution, and, therefore, this solution is iteratively corrected [2] to achieve the original solution. The next PRM method [11] goes one step further and also thins the obstacles themselves, where the level of thinning is determined by a binary search at each step of the algorithm.

The second group of random sampling algorithms was designed for use in models with a number of complex physical constraints. In contrast to PRM, it generates a tree structure instead of a graph, after which this group of algorithms is called the rapidly exploring random tree (RRT) [17]. The input to RRT is a set of constraints (similar to PRM) plus an initial configuration that is the root of the initialised tree structure t_{main}. Creating a tree structure instead of a graph has several undeniable advantages. First, the generated tree incrementally expands towards unexplored regions of the configuration space. Second, it simplifies the path planning part of the process since backtracking is sufficient to find the resulting path. The idea of the whole RRT algorithm contains three basic steps that are cyclically repeated. Randomly generating a new sample in the configuration space is the first step. The next step is to steer the new sample near the nearest list tree t_{main}. The last step is to check if the sample is collision-free. If it is, the sample is added to the t_{main} tree; otherwise, it is rejected.

There are many modifications for the RRT family of algorithms, primarily targeted at special problems. RRT* [13] is an optimised RRT that finds the optimal solution using a heuristic function. It can provide the shortest possible path to the goal in the best case. The problem is that the shortest possible path is guaranteed when the number of samples approaches infinity, which is unrealistic in practice, but the found path is optimal. The dynamic environment can be solved using RRT^X [21] which is the the first proposed asymptotically optimal sampling-based replanning algorithm.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16

Both approaches share the well-known problem of finding a collision-free passage through narrow passages. For PRM algorithms, many different modifications have been introduced to address the narrow passage problem [7]. For example, for low-dimensional configuration spaces, it is possible to exploit this by generating a large number of random samples near obstacles or around the medial axis of the environment [16]. On the other hand, the RRT family of algorithms can work with a medial axis, towards which MARRT (Medial Axis RRT) [6] tries to shift the newly generated samples. The resulting tree structure then does not cover the entire collision-free configuration space, but follows the central axis. Geometry-based methods [23] that sample along a precomputed path can also be used. Another option to solve the narrow passage problem is more detailed sampling around obstacles, such as NP-RRT* (Narrow passage RRT*) [3], or combination of different abstraction levels [20].

Most motion planning methods are primarily developed for the navigation of mechanical objects (robots, autonomous vehicles, etc.), but these methods have applications in other important research areas. For example, the problem of ligand navigation in protein is motion planning in molecular simulations, which inspired our proposed solution. It is possible to use Probabilistic Roadmaps, which help to speed up molecular dynamics simulations, to sample the configuration space of protein [1]. The atomic boundaries of a ligand often lead to sampling in a high-dimensional configuration space, which is an inappropriate problem for the group of PRM algorithms.

A more suitable choice of a planner is an RRT-based algorithm that can even handle motion planning for a flexible ligand [5]. The performance of RRT is greatly affected by the ability to generate new configurations for the high-dimensional space problem, which can be very time-consuming, but ML-RRT (Manhattan-like RRT) copes with this problem [8]. This method has also been modified for flexible ligands [5]. A solution to this problem can also be achieved by projecting the high-dimensional space of the roadmap back into 3D space [4]. Another challenge is to find a passage through a dynamic protein [24] in which individual paths cannot only change their shape but also arise and disappear.

The Voronoi diagram is used not only for better motion planning [23] but also for calculating cavities and their exit areas in protein models [19]. The Voronoi vertex and edge graph capture all possible trajectories of spherical probes avoiding collisions between spherical obstacles (the atoms of the protein model). A probe located in the cavity of a protein model cannot reach the outer space without collision unless the probe radius is reduced to allow the probe to escape from the cavity. By analysing a graph of Voronoi vertices and edges, the exact location where the probe will be located (the primary exit location), and the exact value of the probe radius to which the probe must shrink to escape the cavity can be calculated. The edge of the Voronoi diagram along which the probe could escape from the cavity is then removed, and the shrinking of the probe continues until all remaining exits from the cavity are discovered. The groups constructed from the intersecting probes at the cavity exit locations are referred to as exit areas.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16

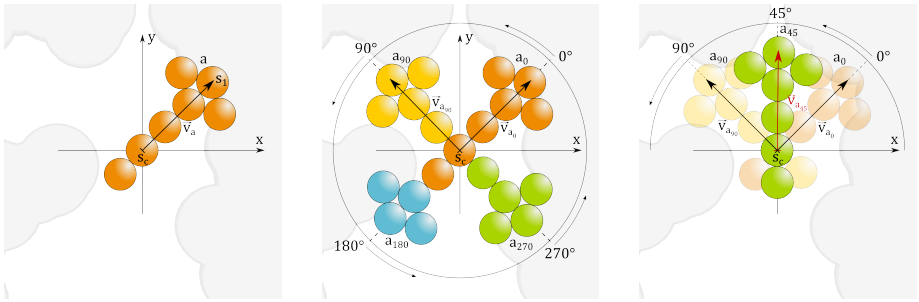
3 Proposed Solution

The proposed solution combines three different approaches - exit areas [18], RRT [17] and the idea of binary search - into one sophisticated motion planning method. Our previous proposed solution [22] combined only exit areas and the RRT method, which was very helpful for detecting narrow passages so that they can be processed in more detail. The biggest problem of the previous solution is that it was not possible to unambiguously decide whether a collision-free agent location exists. Detecting the collision-free location of an agent in a narrow passage was still guided by random pattern generation, so it could happen that an existing collision-free solution was not found. The newly proposed solution removes this pitfall and essentially determines whether an agent's collision-free placement in a narrow passage is possible.

A few modifications to the RRT algorithm are required in the proposed solution. Before the main RRT cycle starts, we need to calculate the exit areas that tell us the exact location of the most problematic places (narrow passage) in the data. By incorporating and modifying the binary search, we can now unambiguously determine if a collision-free path exists through the narrow passage. If this is not possible, we can temporarily close the narrow passage, e.g., by using temporary barriers to avoid the RRT sampling the narrow passage unnecessarily. Otherwise, we have essentially found a collision-free path through one of the worst possible places in the data. Once the narrow passages are located and processed, the main RRT cycle can be run to find a collision-free path through the data.

Let us illustrate in 2D the main idea of the proposed solution to unambiguously determine whether there is a collision-free position in the narrow passage or not. First, we define the principal direction - the orientation of the agent (Fig. 1a). The principal direction can be defined in the same way, which must be uniform for each flexible agent configuration. In our case, the agent $\boldsymbol{A}$ is a ligand consisting of several spheres, which are stored in a list of spheres. Its principal direction $\overrightarrow{\boldsymbol{v_a}}$ can be defined, for example, as the principal direction vector between the center of the central sphere $\boldsymbol{s_c}$ taken as the origin of the local coordinate system of the agent and the center of the first defined sphere $\boldsymbol{s_1}$ inside the structure representing the agent, if the first sphere is not also the central sphere. In case $\boldsymbol{s_1}$=$\boldsymbol{s_c}$, another sphere from the structure needs to be used (e.g., the second or the last one). In the next step, we place the agent in the center of the narrow passage whose exact location is known by computing the exit areas $\boldsymbol{v^i_{exit}}, i = 1, ..., n$, whose center is the center of the narrow passage. The agent is in its initial position (without rotation), and now we need to find the correct collision-free rotation of the agent. We divide the imaginary space of rotations into m sectors, e.g., $m = 4$, where the boundary between the sectors corresponds to the initial angular rotation of the agent. By rotating the agent we get 4 rotated agents $\boldsymbol{a_0}, \boldsymbol{a_{90}}, \boldsymbol{a_{180}}$ and $\boldsymbol{a_{270}}$ whose principal direction is rotated by 0°, 90°, 180° and 270° in the narrow passage (Fig. 1b). For each of these rotated agents, we compute a collision function $\boldsymbol{w}$, which can be arbitrarily defined (e.g. number of collision spheres, size of volume in collision, sum of

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16



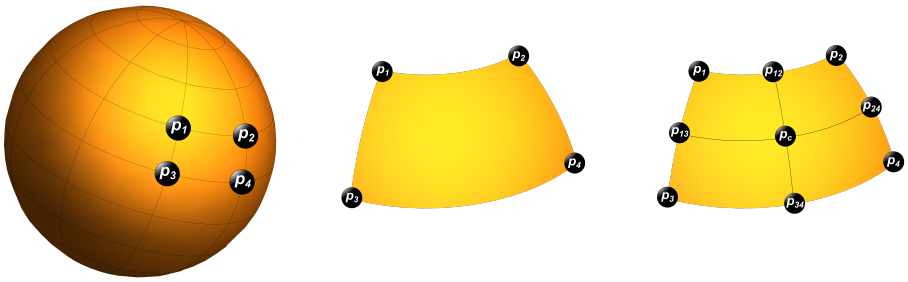
(a) (b) (c)

Fig. 1: The main idea of proposed solution where transparent grey represents protein and white collision free tunnel. (a) Example of agent illustrated as set of orange spheres in the narrow passage of protein, (b) Initial agent rotation to calculate collision-free rotation where each initial agent rotation has a different color for better predictability, (c) First iteration in 2D where the initial agent rotations are illustrated with transparent colors.

collision distances, etc.), and then we sum every two adjacent collision functions ($\boldsymbol{w(a_0)+w(a_{90})}$, $\boldsymbol{w(a_{90})+w(a_{180})}$, etc.). We rank the summed collision functions from the smallest to the largest and process the sector with the smallest summed collision function value, e.g., agents $\boldsymbol{a_0}$ and $\boldsymbol{a_{90}}$. We calculate the new rotation of the agent $\boldsymbol{w(a_{45})}$ that is in the middle of the processed sector, i.e. 45°, (Fig. 1c) and calculate the collision function $\boldsymbol{w(a_{45})}$. We select the half of the sector that has the smaller value of the sum of the collision functions ($\boldsymbol{w(a_0)+w(a_{45})}$ or $\boldsymbol{w(a_{45})+w(a_{90})}$). We repeat this halving until we obtain a collision-free result or reach the chosen maximum number of iterations. If a collision-free result is not found, we process the second initial sector in the same way. In case the result is not found in the second sector, continue with the third, etc.

The proposed solution can also be generalised for 3D. The whole idea behind the algorithm is exactly the same. The only difference is that in 2D only one angular variable is needed to rotate the agent and create sectors, which is insufficient information in 3D. To achieve similar sectors in 3D, it is convenient to use parametric equations of a sphere to achieve sectors which look like plates, see Fig. 2a. The corners of the plate are the principal direction of the initial agent rotation for the sector (Fig. 2b). Dividing the plate gives 4 sub-plates (Fig. 2c) for which 5 new agent rotations ($\boldsymbol{\vec{v}_{p_{12}}}$, $\boldsymbol{\vec{v}_{p_{13}}}$, $\boldsymbol{\vec{v}_{p_c}}$, $\boldsymbol{\vec{v}_{p_{24}}}$ and $\boldsymbol{\vec{v}_{p_{34}}}$) need to be computed in order to compute the summed collision functions ($\boldsymbol{w(p_1)+w(p_{12})+w(p_{13})+w(p_c)}$, $\boldsymbol{w(p_{12})+w(p_2)+w(p_c)+w(p_{24})}$, etc.). The sub-plate with the smallest value of the summed collision functions is further divided into smaller parts. For each selected sub-plate, it is then necessary to compute next 5 new angular rotations and agent collision functions, which can be computationally intensive especially when using poor data structures.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16



(a) (b) (c)

Fig. 2: (a) Space of rotation in 3D, (b) Initial spherical plate, (c) First iteration in 3D

Experimentally, we have found that it is sufficient to compute only the agent rotations to the center of the $\boldsymbol{p_c}$ plate, since the sum of the two collision functions (i.e., $\boldsymbol{w(p_1) + w(p_c)}$, $\boldsymbol{w(p_2) + w(p_c)}$, etc.) is sufficient to determine the appropriate sub-plate for further processing. and each weighting function is then just the sum of the collision function at one corner of the plate and at its center.

We incorporated this solution into our previously proposed solution [22] to find and sample narrow passages in more detail. The structure of the whole algorithm and the proposed solution for unambiguous narrow passages resolution is then as follows. First, we find all exit areas in the data representing narrow passages. We try to place an agent in each found exit area so that it does not collide with the environment (Alg. 1). To do this, we divide the imaginary sphere of possible agent rotations into plates. We compute the collision function $\boldsymbol{w(p)}$ for each plate $\boldsymbol{p}$ (Alg. 1, lines 2-3) and sort the plates according to the value of the collision function $\boldsymbol{w(p)}$ from smallest to largest (Alg. 1, line 4), and then process all plates. Cycling through all the plates (Alg. 1, lines 5-8) is only used if, in some artificial case, the same value of the collision function existed in almost all angular rotations.

If the method does not find a collision-free rotation of the agent, then there is no collision-free path for the agent through this narrow passage and we can fill the narrow passage with a temporary obstacle. If it is possible to place the agent in the narrow passage without collision with the environment, we run a tiny RRT algorithm to map just the neighborhood of the narrow passage. We try to fix each collision pattern with our proposed solution. We do this for each exit area found, yielding a set of tree structures whose root is at the center of the narrow passage. The main RRT algorithm is then run, which searches for a path through the protein from the protein's active site (initial position). In case the main tree is close to some other tree that has been formed around the narrow passage, we merge these two tree structures into one, which solves the narrow passage problem very easily and quickly. The algorithm terminates when we have found a way out of the protein or after a specified maximum number of iterations.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16

Algorithm 1: The proposed solution for finding collision-free position

```
Data: The flexible agent A, List of initial plates P
Result: The collision-free rotation of agent A
Algorithm ProposedSolution
    foreach Plate p in P do
        Compute summed collision function w(p) of plate p
    Sort P by w(p)
    foreach Plate p in sorted P do
        A ← ProcessSection(A, 1, p)
        if A is collision-free then
            return A
    return ∅
Procedure ProcessPlate(The flexible agent A, Current iteration i, Plate p)
    if i = max_iteration then
        return A
    A_c ← Rotate A so its orientation is pointing to the middle of p
    if A_c is collision-free then
        return A
    Sum up w(A_c) with each corner value separately
    Select the sub-plate s of p with minimum summed collision function w(s)
    A ← ProcessSection(A, i + 1, s)
    return A
```

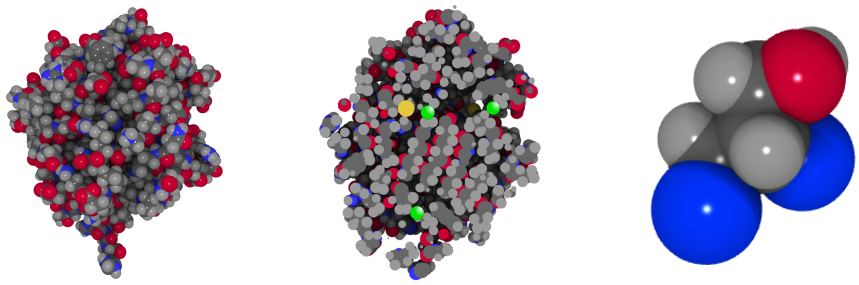
4 Experiments and Results

Both real data and artificial data were used to test the proposed method. Within the real data, we used protein data, which are also freely available in the data bank, and flexible ligands as agents whose meaningfulness was consulted by biomolecular experts. To present the results, we used the *dcp* protein to represent the real data. The artificial data were used to illustrate the strengths and weaknesses of the method. All experiments were performed on a computer with the CPU Intel® Core™ i7-7700K (4.2GHz) and 64GB 2400MHz RAM, the proposed method was implemented in C#. Also the images in this section were created using CAVER Analyst 2.0 [12] unless otherwise noted.

The results of the proposed method for real data are difficult to compare with existing methods since they specialize in different problems, so we compare them with our previous solution (pRRT) to show a significant improvement for finding a collision-free path through the narrow passage.

The experiment is on real protein data, specifically the protein *dcp* (each atom has different color - hydrogen light grey, carbon dark grey, oxygen red, etc.), which can be seen in Fig. 3a. Fig. 3b shows the cut through protein *dcp* with few highlighted narrow passages (green points) which are visible in this particular cut. The example of one configuration of the flexible agent is shown in Fig. 3c. The result of the experiment is a comparison of the collision-free path through narrow passages found using the previous solution and the current solu-

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16



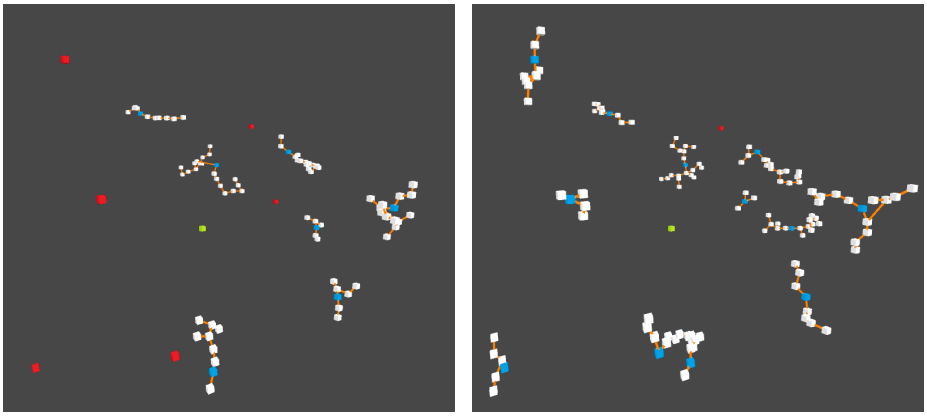
(a) (b) (c)

Fig. 3: (a) Tested data of protein *dcp*, (b) *dcp* protein cut with highlighted narrow passages, (c) The example of one configuration of flexible agent $\boldsymbol{A}$.

tion (Fig. 4 which is created by our own computation system). In both images, the *dcp* protein is invisible so that all narrow passages and the starting point of the main cycle of the RRT algorithm can be seen. The results do not show the finding of the entire tree structure and the resulting passes through the protein, but only the completed processing of the narrow passages and the processing of their closest surroundings. The starting point of the motion planning is the vertex colored in green. The narrow passages are the vertices colored red and blue. If the vertex is blue, the algorithm has found a collision-free agent location in that narrow passage. If no collision-free location was found, the vertex is red. White vertices represent the found collision-free samples by the RRT algorithm in the surroundings of the narrow passage. The individual vertices are then connected to each other by an orange edge that uniquely identifies where the agent can move from its current location. Fig. 4a shows the processed narrow passages of *dcp* protein with the previous solution (pRRT). The result of the same problem by the currently proposed method (cRRT) is shown in Fig. 4b. As can be seen, the cRRT method succeeded in the collision-free placement of the flexible agent even in the narrow passages where the pRRT method failed. One narrow passage remained without finding a collision-free location, which, unlike the results of the pRRT method, does not mean that we cannot climb the path but that there is no collision-free passage for the agent under test through this narrow passage. It is the most significant advantage of the proposed solution.

Table 1 contains a comparison of three algorithms - standard RRT, pRRT, and cRRT. Testing was performed 1000 times on the real biomolecular data mentioned above; each algorithm was timed to two minutes. Each iteration had a different random generator seed to guarantee a different result. At the same time, the same seed was used for all algorithms (the first run with seed x for all algorithms, the second run with seed y, etc.). Table 1 shows how many times a given algorithm was better than the other two algorithms in the evaluated property, where the properties being assessed were a higher number of paths found and shorter pathfinding time. The current proposed cRRT solution always found more paths, namely five, than the other two algorithms. The reason is that

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16



(a) (b)

Fig. 4: (a) Result narrow passage trees of previous RRT solution, (c) Result narrow passage trees of current RRT solution.

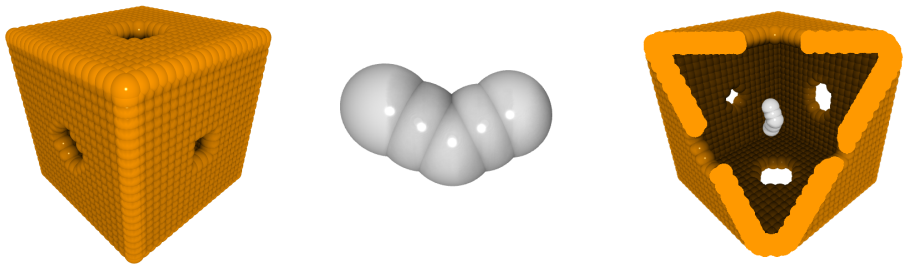
Table 1: Comparison of the RRT algorithm, our previous (pRRT) and current (cRRT) modified versions - tested on the real bio-molecular data

Algorithm	Number of paths	Time of the found path				
		First	Second	Third	Fourth	Fifth
RRT	0	127	78	1	0	0
pRRT	0	243	217	273	48	0
cRRT	1000	630	705	726	952	1000

RRT finds only the first two paths that are wide enough for this algorithm, and pRRT finds a different third path and, in most cases, a fourth path but fails on the fifth path. It is also interesting to note that the RRT algorithm was once able to find the third path while being faster than the other algorithms, a purely random result. The time results of pathfinding between the pRRT and cRRT methods usually differed by units of milliseconds.

The following experiment was to determine the narrow passage threshold for each method. That is, to find how much wider the narrow passage must be for the method to have no problem finding a path through the narrow passage. For this experiment, we made artificial data (Fig. 5a) in the shape of a cube; each wall has a passage with different radius (e.g. first wall has passage with $r = 1$, second passage with $r = 1.4$, etc.). The radius of the smallest possible passage corresponds to the radius of one sphere of the agent (Fig. 5b). The initial position is in the middle of the cube (Fig. 5c); the ideal number of paths found is the same as the number of walls of the cube, i.e., six. The result is in Table 2, which indicates how many times one of the RRT algorithms was able to find a path through the narrow passages in the wall. For each algorithm,

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16



(a) (b) (c)

Fig. 5: (a) The artificial testing data with different radii of narrow passages, (b) The example of flexible agent $\boldsymbol{A}$, (c) Data cut with starting position of the agent $\boldsymbol{A}$

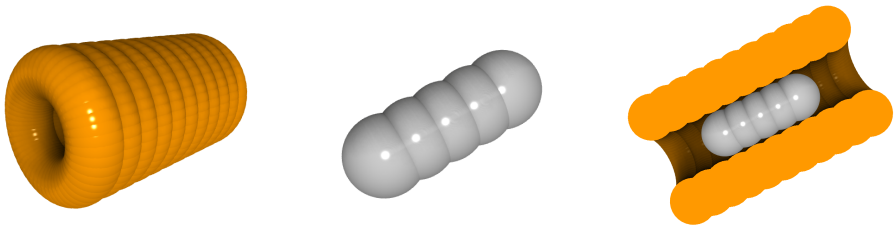
Table 2: Collision-free paths through narrow passages with different radii sizes

Algorithm	Narrow passage number (radius)					
	First (1)	Second (1.3)	Third (1.6)	Fourth (1.9)	Fifth (2.2)	Sixth (2.5)
RRT	0	0	0	0	476	1000
pRRT	0	0	671	1000	1000	1000
cRRT	1000	1000	1000	1000	1000	1000

we repeated the measurements 1000 times with a agent whose sphere radius is $r = 1$. In all cases, the standard RRT algorithm finds only the largest narrow passage ($r = 2.5$), which is the expected result since the method does not use any auxiliary information or structures. With a sufficiently long sampling (up to 2 minutes) of the configuration space, it also finds a path through the second-largest narrow passage ($r = 2.2$) in almost half of the cases. The previous solution pRRT performs much better, always finding a path through the narrow passage in half of the cases. The best solution is the proposed solution, which finds a collision-free path through every narrow passage, even the smallest possible.

The previous results have shown that the proposed method finds a passage through even the tightest possible narrow passage. Therefore, now we focus on the method itself to see what operations and how many iterations are needed to find one collision-free position. For this experiment, we have created artificial data in the form of a tunnel (Fig. 6a) into which we are trying to place a rod-shaped agent (Fig. 6b). Before each algorithm run, the data and the agent itself were randomly rotated (each with a different rotation). The experiment was repeated 500 times; the method searched for collision-free rotations of the agent (Fig. 6c). The results of this experiment are shown in Table 3. We tried this experiment for different numbers of initial spherical plates, which must be sorted further by the collision function and processed. As we can see, as the number of plates increases, the computation time and finding the correct solution

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16



(a) (b) (c)

Fig. 6: (a) The artificial testing data of tunnel, (b) The example of agent $\boldsymbol{A}$, (c) Data cut with starting position of agent $\boldsymbol{A}$.

Table 3: Result of different initial conditions of the proposed solution cRRT

Number of plates	Plate order			Plate iteration			Time [ms]
	First	Second	Higher	First	Second	Higher	
2	500	0	0	251	68	181	27.871
6	413	56	31	370	43	87	34.158
15	387	93	20	401	67	32	65.966
28	341	107	52	384	71	45	139.466
45	326	108	66	398	58	44	204.932

also increase. It makes sense because more initial agent rotations and collision functions must be calculated.

In most cases, the solution was found in the first spherical plate and even in the first possible iteration. However, it is surprising that the method works very well and very fast also for a minimal number of plates, namely two. All solutions were found in the first plate, which makes sense since the data is symmetric. However, the method needs to be adjusted appropriately, as the larger the initial plate, the greater the chance that the method iterates into the wrong part of the plate. In this case, we need to add a stack or priority queue and process the rejected part of the plate as well.

We conclude the section with three essential observations from testing the proposed method. The first is that the method is severely limited by the accuracy of the computer representation of decimal numbers. It is vital when calculating the rotation of an agent from one position to another, so one needs to have a perfect understanding of working with decimals on the computer or use a suitable library (e.g., Unity can handle this). The next observation is that the most appropriate collision function for biomolecular data composed of spheres turned out to be the total volume of the agent that is in collision with its surroundings. The last is that the unambiguous result of whether or not a collision-free path

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16

exists is closely related to the maximum number of iterations. Experiments have shown that for protein data it is longest in the fourth iteration.

5 Conclusion

A modified version of the RRT algorithm, cRRT, based on our previous method, whose improvement for collision-free pathfinding consists mainly in finding collision-free paths through narrow passages, is presented in this paper. In addition to finding the path itself, the method also provides the exact location of the narrow passages, which it can investigate in detail. This information is crucial, and our modified RRT algorithm can decide whether the narrow passage path exists. Since the proposed solution must additionally compute the position of the exit areas and sample their neighborhoods, it is not suitable for data without narrow passages - it will find the correct path, but probably more slowly than the original RRT. Once the data contains narrow passages, the proposed solution excels and provides better results than our original solution and many times better results than the original RRT algorithm. Moreover, the method can find collision-free narrow passages even for data that are only touching (agent with protein).

Acknowledgement

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic, the project SGS-2022-015 New Methods for Medical, Spatial and Communication Data.

References

1. N. M. Amato, K. A. Dill, and G. Song. Using motion planning to map protein folding landscapes and analyze folding kinetics of known native structures. *Journal of Computational Biology*, 10(3-4):239–255, 2003.
2. O. B. Bayazit, D. Xie, and N. M. Amato. Iterative relaxation of constraints: A framework for improving automated motion planning. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3433–3440. IEEE, 2005.
3. A. Belaid, B. Mendil, and A. Djenadi. Narrow passage rrt*: a new variant of rrt. *International Journal of Computational Vision and Robotics*, 12(1):85–100, 2022.
4. J. Cortés, S. Barbe, M. Erard, and T. Siméon. Encoding molecular motions in voxel maps. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(2):557–563, 2011.
5. J. Cortés, D. T. Le, R. Iehl, and T. Siméon. Simulating ligand-induced conformational changes in proteins using a mechanical disassembly method. *Physical Chemistry Chemical Physics*, 12(29):8268–8276, 2010.
6. J. Denny, E. Greco, S. Thomas, and N. M. Amato. Marrt: Medial axis biased rapidly-exploring random trees. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 90–97. IEEE, 2014.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16

7. M. Elbanhawi and M. Simic. Sampling-based robot motion planning: A review. *Ieee access*, 2:56–77, 2014.
8. E. Ferré and J.-P. Laumond. An iterative diffusion algorithm for part disassembly. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, volume 3, pages 3149–3154. IEEE, 2004.
9. R. Geraerts and M. H. Overmars. A comparative study of probabilistic roadmap planners. In *Algorithmic Foundations of Robotics V*, pages 43–57. Springer, 2004.
10. P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
11. D. Hsu, G. Sánchez-Ante, H.-l. Cheng, and J.-C. Latombe. Multi-level free-space dilation for sampling narrow passages in prm planning. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1255–1260. IEEE, 2006.
12. A. Jurcik, D. Bednar, J. Byska, S. M. Marques, K. Furmanova, L. Daniel, P. Kokkonen, J. Brezovsky, O. Strnad, J. Stourac, et al. Caver analyst 2.0: analysis and visualization of channels and tunnels in protein structures and molecular dynamics trajectories. *Bioinformatics*, 34(20):3586–3588, 2018.
13. S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.
14. L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171, 1998.
15. S. Koenig and M. Likhachev. D* lite. In *AAAI/IAAI*, pages 476–483, 2002.
16. H. Kurniawati and D. Hsu. Workspace-based connectivity oracle: An adaptive sampling strategy for PRM planning. In *Algorithmic Foundation of Robotics VII*, pages 35–51. Springer, 2008.
17. S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.
18. M. Manak. Voronoi-based detection of pockets in proteins defined by large and small probes. *Journal of Computational Chemistry*, 40(19):1758–1771, 2019.
19. M. Manak, A. Anikeenko, and I. Kolingerova. Exit regions of cavities in proteins. In *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering*, BIBE, pages 1–6. IEEE Computer Society, 2019.
20. A. Orthey and M. Toussaint. Section patterns: Efficiently solving narrow passage problems in multilevel motion planning. *IEEE Transactions on Robotics*, 37(6):1891–1905, 2021.
21. M. Otte and E. Frazzoli. RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research*, 35(7):797–822, 2016.
22. J. Szkandera, I. Kolingerová, and M. Maňák. Narrow passage problem solution for motion planning. In *International Conference on Computational Science*, pages 459–470. Springer, 2020.
23. V. Vonásek, J. Faigl, T. Krajník, and L. Přeučil. A sampling schema for rapidly exploring random trees using a guiding path. In *Proceedings of the 5th European Conference on Mobile Robots*, volume 1, pages 201–206, 2011.
24. V. Vonásek, A. Jurčík, K. Furmanová, and B. Kozlíková. Sampling-based motion planning for tracking evolution of dynamic tunnels in molecular dynamics simulations. *Journal of Intelligent & Robotic Systems*, 93(3):763–785, 2019.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_16