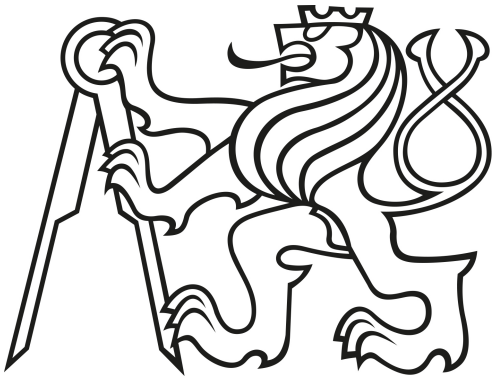
Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Cybernetics

Study programme: Biomedical Engineering and Informatics



Jindřich Durčák

Sampling–Based Motion Planning for Tunnel Detection in Protein Structures

Diploma thesis

Supervisor: Ing. Vojtěch Vonásek, Ph.D.

Prague, May 2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Jindřich D u r č á k

Study programme: Biomedical Engineering and Informatics

Specialisation: Biomedical Informatics

Title of Diploma Thesis: Sampling-Based Motion Planning for Tunnel Detection in Protein Structures

Guidelines:

1. Study the problem of motion planning for rigid 3D objects [3], study sampling-based motion planning methods like Rapidly Exploring Random Tree (RRT) [4].
2. Implement a RRT-based method for motion planning of 3D objects.
3. Get familiar with tunnel detection and motion planning in protein structures [1,2,6].
4. Adapt method from task 2 for the purpose of tunnel detection in static protein structures. Assume tunnels for spherical probes. Utilize existing tools for visualization, like Pymol.
5. Adapt method from 4) for tunnel detection in dynamic protein structures.
6. Design and implement method to analyze traversability of ligands in static tunnels.
7. Models of proteins and ligands will be provided by the supervisor.

Bibliography/Sources:

[1] Cortés, Juan, et al. - A path planning approach for computing large-amplitude motions of flexible molecules - Bioinformatics 21.suppl 1 (2005): i116-i125.
[2] Chovancova, Eva, et al. - CAVER 3.0: a tool for the analysis of transport pathways in dynamic protein structures - (2012): e1002708.
[3] LaValle, Steven M - Planning algorithms - Cambridge university press, 2006.
[4] LaValle, Steven M., and James J. Kuffner Jr. - Rapidly-exploring random trees: Progress and prospects - (2000).
[5] Kavraki, Lydia E., et al. - Probabilistic roadmaps for path planning in high-dimensional configuration spaces - Robotics and Automation, IEEE Transactions on 12.4 (1996): 566-580.
[6] Carl Branden, John Tooze - Introduction to Protein Structure - Garland Science; 2 edition (January 3, 1999).

Diploma Thesis Supervisor: Ing. Vojtěch Vonásek, Ph.D.

Valid until: the end of the summer semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 11, 2017

Author statement for undergraduate thesis:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 24.5.2017 .....

Jindřich Durčák

Acknowledgements

First, I would like to thank my supervisor Vojtěch Vonásek for his guidance of this diploma thesis. In addition to the excellent explanation of motion planning problematics, I really appreciated his enthusiasm for molecular biology.

I would also like to express great thanks to my family and girlfriend for never–ending support throughout my long study years.

Abstract

Study of proteins is a major area of molecular biology research. Since the structure and function of proteins are tightly bound, obtaining information about their spatial configuration is one of the main goals of their research. In the past, a large number of different protein structures has been obtained and free access to this information offers a great potential for calculations in structural bioinformatics. One of the most interesting areas is the detection of tunnels in protein structures.

Development of reliable algorithms, which detect tunnels in protein structures, and simulation of the passage of chemicals through these tunnels can greatly accelerate research in the field of molecular biology and so, many research groups are devoted to these methods all around the world.

Motion planning is a well–known area of cybernetics and robotics, where it is used to design robot paths in state space. If the state space is formed by the structure of protein and instead of the robot we use a probe of a defined size, we can transfer the motion planning algorithms from the robotic to the biological domain.

Within this diploma thesis, algorithms known from the motion planning of robots were modified for the task of tunnel detection in both static and dynamic protein structures. Furthermore, a method for geometrical analysis of the passage of molecules through tunnels was proposed and implemented.

Key words: detection of protein tunnels; motion planning; dynamic protein structures; geometrical analysis of the passage of molecules

Abstrakt

Studium proteinů je hlavní oblastí výzkumu molekulární biologie. Jelikož je struktura a funkce proteinů těsně svázaná, získání informací o jejich prostorové konfiguraci patří mezi hlavní cíle jejich zkoumání. V minulosti se již podařilo získat velké množství struktur různých proteinů a volný přístup k těmto informacím nabízí velký potenciál pro teoretické výpočty strukturní bioinformatiky. Jednou z velmi zajímavých oblastí je i hledání tunelů ve strukturách proteinů.

Vytvoření spolehlivých algoritmů detekce tunelů ve strukturách proteinů a simulace průchodu chemických látek těmito tunely může značně urychlit a zefektivnit výzkum v oblasti molekulární biologie, a proto se těmto metodám věnuje mnoho výzkumných skupin po celém světě.

Plánování cest patří mezi dobře prozkoumané oblasti kybernetiky a robotiky, kde se používá k návrhu možných cest robotů ve stavovém prostoru. Pokud stavový prostor tvoří struktura proteinu a místo robota použijeme sondu definované velikosti, můžeme algoritmy plánování cest převést z robotické do biologické domény.

V rámci této diplomové práce byly implementovány algoritmy známé z plánování cest robotů tak, aby nalezly tunely ve statických i dynamických proteinových strukturách. Dále pak byla navržena a implementována metoda pro simulaci průchodu molekul nalezenými tunely.

Klíčová slova: detekce proteinových tunelů; plánování cest; dynamické proteinové struktury; geometrická analýza průchodu molekul

Table of Contents

List of abbreviations

DNA	deoxyribonucleic acid
MPNN	motion planning nearest neighbor
NMR	nuclear magnetic resonance
OBB	Oriented bounding boxes
RNA	ribonucleic acid
RRT	Rapidly–exploring Random Tree

Chapter 1

Introduction

Proteins are biomolecules that perform many essential functions in every living organism. They consist of twenty α–amino acids that create long sequences. Enzymes are proteins that catalyze biochemical reactions, which occur at the enzyme's active site that can be deeply burried in the protein structure. Individual protein sequences are composed from tens up to several thousands amino acids [1]. Given so many possibilities how different amino acids can be connected, the 3D structures of distinct proteins significantly vary. For the illustration of protein structure and comparison with DNA, the structure of nucleosome complex is shown in Figure 1.1.

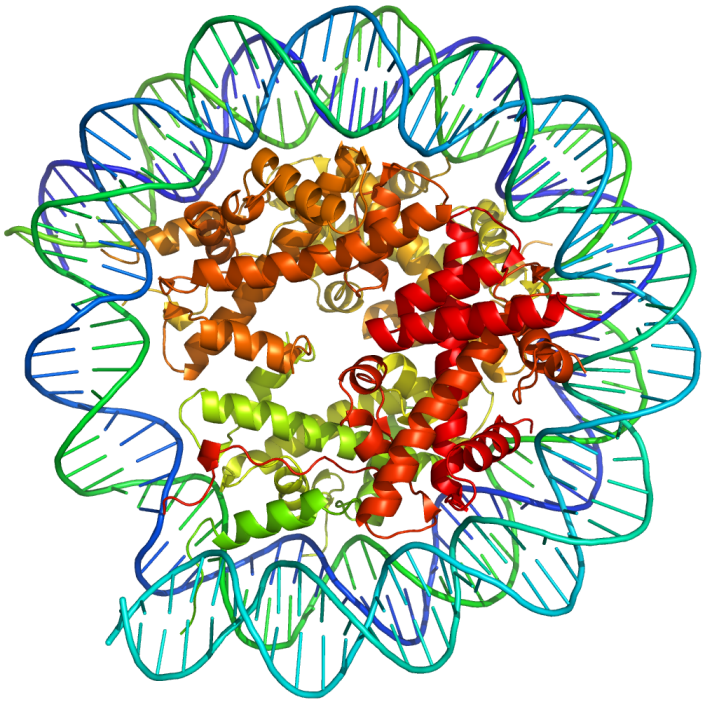


Figure 1.1: The structure of nucleosome complex of a protein (in center) and a DNA fragment (encircling protein), created by software tool PyMOL (pdb file 1AOI) [2].

1.1 Structure determining methods

During the research of proteins in 20th century, scientists discovered that protein function is directly determined by its structure. Not only because of this discovery, huge effort was made in development of protein structure determining methods. Nowadays, there are two main methods: X–ray crystallography and Nuclear magnetic resonance spectroscopy (NMR). Thanks to great improvements in the quality of resolution, also cryo–electron microscopy is used not only for determination of highly complex structures (e.g., ribosome), but also for individual proteins [3, 4].

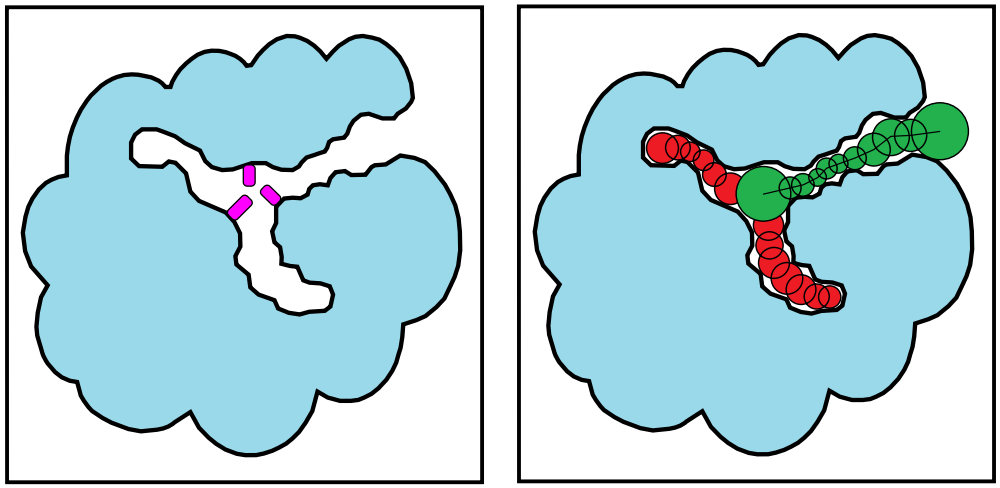
Although there are methods based on electron microscopy, the majority of structures are still determined by X–ray crystallography, which produces static XYZ coordinates of each atom in the studied protein molecule. All protein structures that were published in scientific journals are freely available in Protein Data Bank [5] and protein structures from this database were used in this thesis.

1.2 Protein tunnels

Since a lot of structures have been already determined and a substantial part of them are easily accessible, there are many possible research areas in structural bioinformatics that can be studied. One of the interesting topics is detection of tunnels in the protein structure that lead to enzyme's active site. The active site is a location usually formed by several amino acid residues, where incoming substrates are chemically converted to products and this site can be deeply burried within the core of enzyme [6].

Generally, the task of tunnel detection is to find a collision–free path that leads from the functional site of protein to its surface. The tunnel detection methods utilize the three–dimensional structure usually from Protein Data Bank as the input and the detected tunnels are the output. After the tunnels are detected, a computer–based simulation with various chemical compounds can be performed in order to predict, if the compounds are able to get to the active site. This approach can find promising substances for further examination and experiments in real world. Demonstration of protein structure with tunnels and tunnel finding process is shown in Figure 1.2 on the next page. The tunnel finding process is thoroughly described in the upcoming chapters.

Development of methods that would reliably find tunnels in protein structures is therefore essential. Provided with such techniques, scientists in both basic and applied



a) b)

Figure 1.2: A tunnel in protein structure. a) Simplified illustration of the protein (in light blue) with various paths within its structure and highlighted active site, formed by three amino acid residues (in magenta). b) The result of tunnel finding process, where paths in the structures were scanned with a circular probe. A successfully found tunnel is depicted in green. Paths that were unable to reach the surface of protein are shown in red.

research can significantly increase the efficiency of their work. For example, these methods can be very useful in pharmacology, where it would be possible to simulate passage of various chemicals through the found tunnels well before any experiment in a laboratory takes place. This way it would be possible to identify potential candidates for further testing, while the compounds that were not able to pass in the simulation would not be used for the laboratory experiments.

Widely used tools for tunnel detection in protein structures are usually based on Voronoi diagrams, which allow the development of fast performing implementations. A disadvantage is that three–dimensional non–spherical probes cannot be employed and other methods need to be used in such case, for example motion planning algorithms.

1.3 Motion planning

Motion planning is a widely studied area in cybernetics and robotics. Sampling–based methods offer an efficient solution to what is otherwise a very challenging problem in path planning by sampling the configuration space [7]. Apart from its standard application in the robotic domain, motion planning can be also employed in structural bioinformatics, since the spatial coordinates of a protein from Protein Data Bank can

be used analogously to a robot path planning in buildings.

One of the sampling–based algorithms is Rapidly–exploring Random Tree (RRT). This probabilistic algorithm efficiently searches high–dimensional configuration spaces by randomly building a space filling tree by addition of nodes to collision–free locations. It intentionally grows towards areas of the configuration space that were not visited before and this method can easily handle obstacles. This is a great advantage when compared with the complete geometric methods, which focus on exact representation of geometry or topology of the configuration space, thereby ensuring completeness, but these solutions also have great computational demands in high–dimensional assignments [8].

Figure 1.3 shows an exemplary result of RRT in two dimensions. In this figure, RRT was run for a total of 5000 iterations. It can be seen that the created tree has successfully explored the configuration space, while avoiding circular obstacles (shown in blue color). Similar solutions can be implemented not only in two dimensions, but also high–dimensional configuration spaces, for example a 6D space with three spatial dimensions and three rotation dimensions, which is suitable for molecule motion planning.

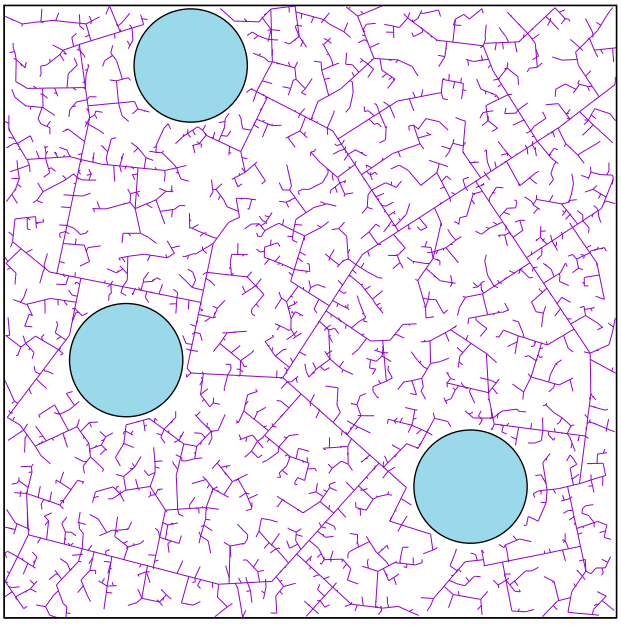


Figure 1.3: Rapidly–exploring Random Tree (RRT) in two dimensions. Blue circles represent obstacles.

1.4 Overview of the thesis

In this diploma thesis, the objective was to modify and apply RRT algorithms for tunnel detection in protein structures and geometrical analysis of passage of chemical compounds through these tunnels. Chapters 2 and 3 provide theoretical background of proteins, tunnel detection and state of the art in this research area. Chapter 4 introduces solution outline that was proposed for all assigned tasks.

In this thesis, we have designed and developed novel modifications of RRT for the purpose of tunnel detection in both static and dynamic protein structures. The ideas of static tunnel detection is based on previous work [9]. This thesis also introduces a new method for tunnel detection in dynamic protein structures. Contrary to already developed method for dynamic tunnel detection [10], which used a continuous pruning of a single configuration tree, we designed a more straightforward method that computes the tunnels separately in each frame.

We also propose a novel method for geometrical analysis of tunnel traversability of non–spherical rigid probes in static tunnels. Chapters 5 and 6 present results and discussion of results. Chapter 7 provides the overall summary of this diploma thesis.

Chapter 2

Foundations of protein biochemistry

During the long history of the Earth, life has evolved into incredibly complex forms. From single–celled microorganisms to multicellular animals and plants, different living creatures have adapted both to extreme arctic and tropic climates, being able to live on the ground, underwater and in the air. What amazes maybe even more is the fact that on the molecular level, individual organisms are quite similar, because they utilize same types of biomolecules and often use identical metabolic pathways.

2.1 Biomolecules

There are four main types of biomolecules — carbohydrates, lipids, proteins and nucleic acids. All of these large macromolecules have fundamental roles in every living organism and this section will shortly introduce them. Carbohydrates mainly serve as structural components and also storage of energy. The complex carbohydrates are usually composed only of carbon, oxygen and hydrogen and they typically form hexagonal monomers, which are connected into large polymers [11].

Lipids are a chemically diverse group of biomolecules, whose common and defining feature is their insolubility in water. They are the major structural elements in biological membranes and also the principal longer term storage of energy. Nucleic acids (DNA and RNA) are the molecular repositories of genetic information. The structure of all proteins, and basically of every biomolecule or other cellular component, is a product of information programmed into the DNA sequence [12]. Proteins mediate virtually every process that takes place in a cell and as the main interest area of this thesis, the following sections describe proteins in detail.

2.2 Proteins

Proteins are large biomolecules that are composed of α–amino acid residues, which form one or more chains. The general structure of an α–amino acid is shown in Figure 2.1. Each α–amino acid contains amine ($-NH_3^+$) and carboxyl ($-COO^-$) functional groups and also a side chain that is specific for every residue. Given the physicochemical properties of amine and carboxyl functional groups, two α–amino acids can undergo condensation reaction, during which both residues are joined. This process can be repeated many times and so, large macromolecules can be formed. There are twenty standard α–amino acids that compose proteins and with this great variability, individual proteins differ in sequence length, function and structure [13].

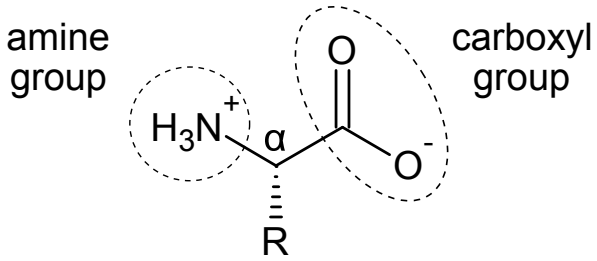


Figure 2.1: The general structure of an α–amino acid in ionized form. The alpha carbon is labeled. R represents the side chain.

2.2.1 Protein functions

Although all four main types of biomolecules play miscellaneous roles in living organisms and none of them has only one purpose, proteins have the most diverse functions and control the majority of all biochemical processes. They are encoded by genes, which are nucleic acid sequences that serve as a blueprint for each protein [14]. Based on the biological function, proteins can be classified into several groups — enzymes (e.g., pyruvate dehydrogenase), transport proteins (hemoglobin), storage proteins (ferritin), mechanical support proteins (collagen), receptors (insulin receptor), antibodies (immunoglobulin G) and hormones (thyroid–stimulating hormone).

The most prominent class are enzymes, which are proteins (or in few rare cases RNA molecules — ribozymes) that catalyze biochemical reactions, during which substrates are converted to products in the active site of enzymes. Almost all metabolic processes in living organisms require these biocatalyzators in order to make reaction rates fast enough to sustain life [15]. Enzymes are uniquely specific to particular chemical substances, which is a feature arising from their three–dimensional structure.

In order to explain the specificity of enzymes, scientists have proposed already in late 19th century that both enzyme and substrate possess precise complementary geometric shapes that fit exactly into each other [16]. This "lock and key" model successfully explains the specificity factor, but fails to interpret the great increase of the reaction rate. Therefore in 1958, a modification to the "lock and key" model was suggested. Since enzymes are flexible protein structures, the active site is continuously reshaped by interactions with the substrate as it approaches the enzyme's active site and this mechanism is often called the "induced fit" model [17]. By this adjustment to the original theory, the binding of the substrate is not seen as a simple attachment to the rigid active site, but as a precise reconfiguration that enables the enzyme to perform its catalytic function. The "induced fit" model is schematically shown in Figure 2.2.

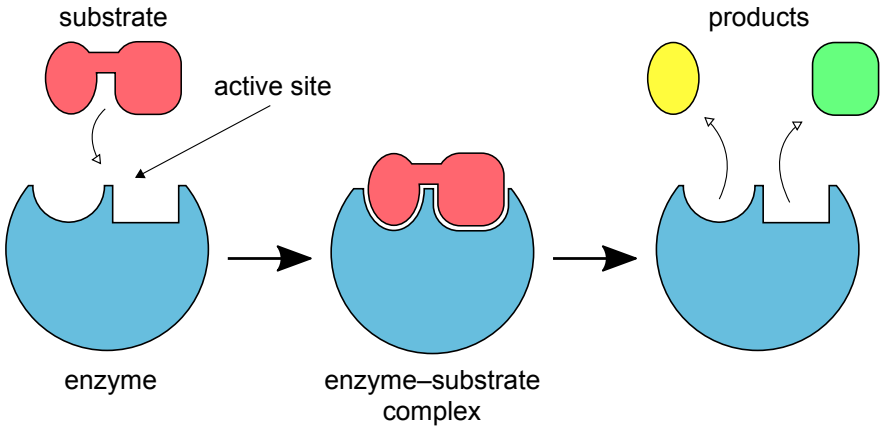


Figure 2.2: Enzyme reaction scheme (induced fit model). Note that the active site in the enzyme–substrate complex slightly changes.

2.2.2 Protein structure

Individual amino acid residues are connected via peptide bonds into long chains. Peptide bonds are formed by the amine and carboxyl groups that are bound to the alpha carbon. For this reason, the whole protein sequence is linked by a repeating motif (amine group — alpha carbon — carboxyl group), which is called a protein backbone. The only difference in the monomers is mediated by the side chains. An exemplary polypeptide chain is shown in Figure 2.3 on the next page.

To make this research area a little clearer, protein structure is categorized into four levels. The primary structure is simply the linear sequence of individual amino acids in the protein chain. The secondary structure refers to a short segment of a polypeptide chain and describes the local spatial arrangement of its main–chain atoms

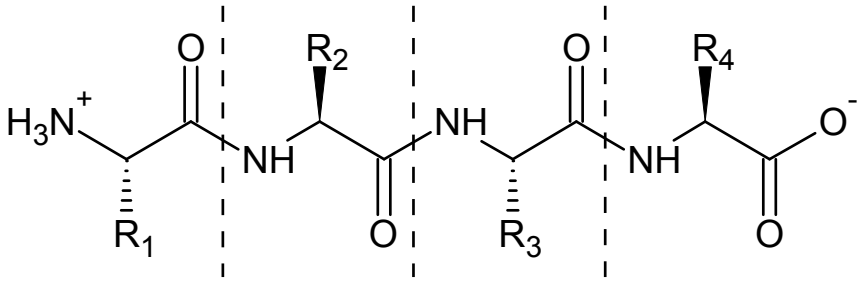
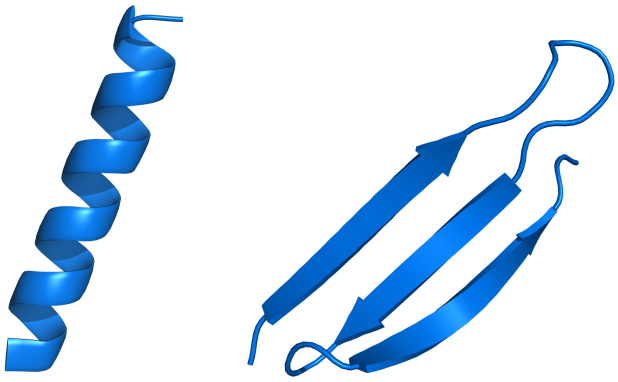


Figure 2.3: A short polypeptide chain, which is formed by four amino acids (separated by dashed lines). Peptide bonds connect amino acids and only side chains of individual residues are different.

(backbone), without regard to the conformation of its side chains or its relationship to other segments [12]. The most common secondary structures are α–helix and β–sheet. Other types of the secondary structures are 3_{10} helix, π–helix, polyproline helix or α–sheet, but these types are rare in native protein structures. An illustration of α–helix and β–sheet is shown in Figure 2.4.



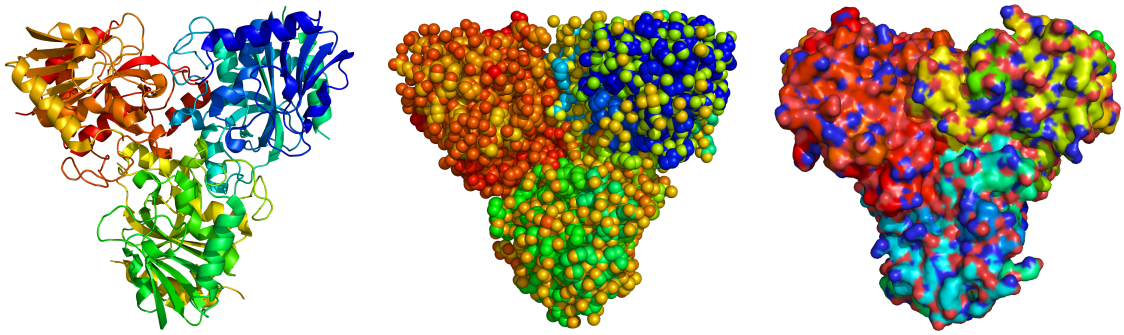
a) b)

Figure 2.4: Protein secondary structure: a) α–helix and b) β–sheet. While α–helix is an independent structure motif, β–sheet is composed of few β–strands (three in this particular example). The images were created from pdb files 5GHR (α–helix, 18 residues) and 1INY (β–sheet, 34 residues), by computer program PyMOL.

The tertiary structure is the complete three–dimensional structure of a polypeptide chain. There are two main classes of proteins based on their tertiary structure: globular and fibrous. Some proteins are composed by two (or more) separate polypeptide chains, which may be identical or different. The quaternary structure is arrangement of these protein subunits into three–dimensional complexes [12]. Figure 2.5 on the next page shows the three basic types of tertiary structure representation that are usually used.

Even small organic molecules like glucose can have different spatial conformations,

therefore its structure is dynamic and individual atoms from the glucose molecule change positions relative to each other. For that reason, it is no surprise that large macromolecules like proteins also have a dynamic structure. X–ray crystallography can only provide static snapshots of conformational substates, but NMR methods can reveal the rates of interconversion between the substates, giving a little insight into the dynamic nature of the protein structure [18].



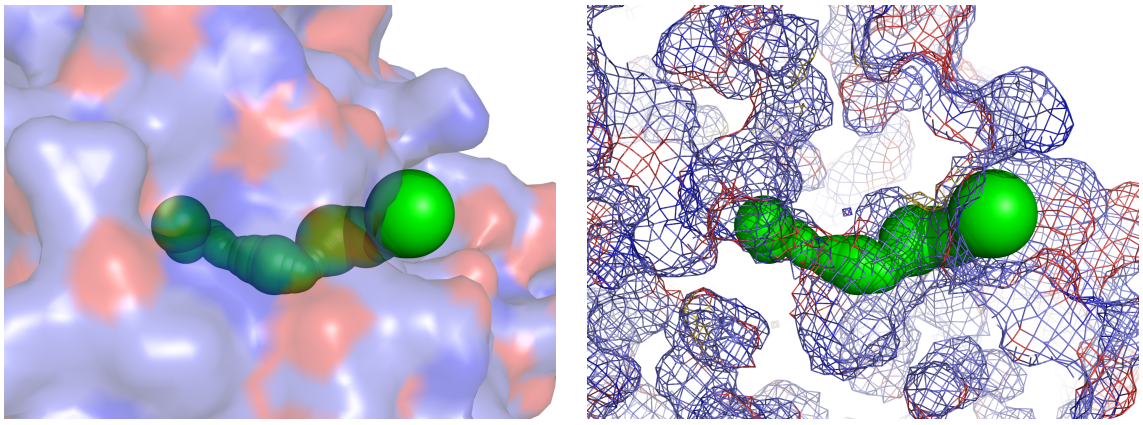
a) b) c)

Figure 2.5: Common representations of protein tertiary structure. a) The cartoon representation shows the secondary structure of protein and provides general protein appearance. b) The sphere representation depicts all atoms as spheres. This illustration is useful for the tunnel detection task, because spheres are the basis of state space used in three–dimensional motion planning. c) The surface representation shows protein surface that would be traced out by the molecules of water in contact with the protein. Images were created from pdb file 1LV8 by computer program PyMOL.

2.2.3 Protein tunnels

Active sites in enzymes, or other functional sites in different proteins, are often located in the core of these macromolecules. In order to be functional, the active sites must be accessible to substrate and other molecules that undergo the biochemical reaction. Accessibility is enabled by paths that connect the active site with the protein surface and also its surroundings. These paths are ordinarily called tunnels, pores or channels, although channels are more frequently perceived as paths that go completely through the protein, thus having two entrances and no active site is involved [19].

Protein tunnels have certain physicochemical properties that decide which molecules can access the functional site and make a physiological effect. Obtaining the properties is fundamental for analysis of structure–function relationship and also for the design of new molecules in various biotechnological applications. Figure 2.6 on the following page shows a tunnel which leads from the active site to the protein's exterior.



a) b)

Figure 2.6: A tunnel in protein structure. a) Illustration of the tunnel with the surface representation. In this case, surface is transparent, which enables us to see both ends of the tunnel. b) The same tunnel, but depicted with so–called mesh representation. This representation shows the web–like surface projection, which gives better insight into the protein structure near the found tunnel. The images were created by computer program PyMOL (from pdb file 1CQW). The tunnel was detected by RRT algorithm that is described in chapter 4.1.

2.3 Motivation for the thesis

The function of proteins is given by their three–dimensional structure. Proteins realize their functions by interacting with other molecules and these interactions take place at the active site. Active sites are accessible through so–called tunnels. Since it is not easy to determine if a specific active site can be reached by a molecule of interest, there is high demand for computational methods that can detect tunnels in 3D models of protein structures. Acquiring knowledge about tunnels and their traversability is valuable due to the potential to reduce the costs of laboratory experiments.

Tunnel detection has proved to be useful in many applications, because it can provide an unique perspective into the roles of particular tunnels for reaction mechanism of diverse enzymes. Apart from the clarification of biochemical processes, information derived by different software tools was also used in drug development or design of proteins with novel functional properties [20]. General approaches to the tunnel detection and also description of selected software tools are summarized in the following chapter.

After the protein structure is determined, numerous computations in the field of structural bioinformatics can be performed. Major research topics include protein folding and dynamics, molecular recognition, drug discovery, protein engineering or tunnel detection [21]. With these theoretical approaches, valuable information about

molecular functions of different proteins can be obtained.

Although the behavior of molecules is determined mainly by electrostatic interactions, many researchers believe that suitable trajectories of the ligand (a molecule that forms a complex with a biomolecule) in protein structures can be detected solely on the geometrical properties of the protein. For this reason, many approaches for tunnel detection have been developed in last two decades. The next chapter describes selected state of the art methods for tunnel detection.

Chapter 3

Computations of protein tunnels

There are many different areas of interest in structural bioinformatics. With the increasing amount of data, which is generated about structures of various biomolecules every year, computational approaches can provide scientists with powerful tools that could improve the research process in many study fields of molecular biology. In this thesis, the main focus is directed towards tunnel detection in protein structures and passage simulation of small rigid molecules through the found tunnels.

3.1 Tunnel detection in proteins

In the last twenty years, several distinct geometry–based tunnel detection tools have been developed. These methods can be categorized into several groups, based on different solving principles — grids, Voronoi diagrams, sampling methods and surface–based algorithms. Individual techniques have different advantages and disadvantages that will be discussed in this chapter.

3.1.1 Grid–based methods

This group of tunnel detection algorithms is based on rasterization of 3D space using a uniformly distributed grid. Protein atoms, represented by hard spheres with appropriate van der Waals radii, are modeled on the discrete grid and all grid nodes are subsequently clustered into two classes, either located in the protein atoms or outside the atoms. The van der Waals radius of an atom is the radius of an imaginary hard sphere that represents the smallest distance, in which another atom can approach. The nodes that are not in protein atoms can be in cavities, tunnels or any external

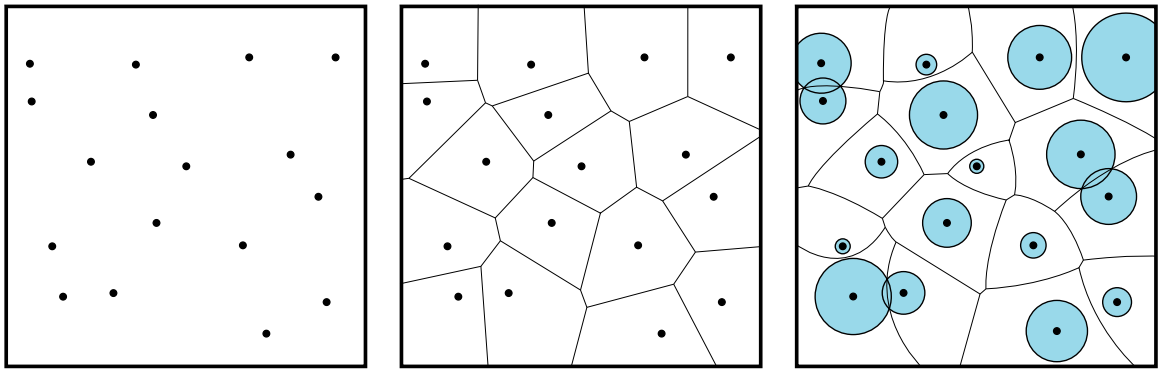
environment of protein. Tunnels are then detected via graph–traversing algorithms.

One of the grid–based tools is CAVER 1.0 [22]. After protein atoms are modeled on the grid, the attention is paid to nodes that are on the boundary of the protein hull, because these nodes are at possible tunnel entrance positions. A weighted graph is constructed and then, modified Dijkstra's algorithm is used to find shortest low–cost path. Other software tools are POCKET [23], HOLLOW [24] or 3V [25].

Grid–based methods are relatively easy to implement and they can compute optimal solutions. But for a specific resolution, the size of memory that is used to store the grid, and also time to search the grid, grows exponentially with the number of dimensions of the space. These properties limit grid–based tools only to low–dimensional problems, usually to 3D. Another problem rises when a different resolution of grid is used, because with varying grid settings, computed tunnels can have diverse shape or volume [8].

3.1.2 Methods based on Voronoi diagrams

The next group of methods employs Voronoi diagrams to solve the tunnel detection task. The main conception of this approach is the division of state space to a diagram in a such fashion that each edge of the diagram has the same distance to its nearest points. Furthermore, the points from the state space can have different significance and the result of division of such point set is a weighted Voronoi diagram. This modification is useful for atoms, because they have different radii that can be used as weights. Both elementary and weighted Voronoi diagrams are presented in Figure 3.1.



a) b) c)

Figure 3.1: Voronoi diagrams. a) Two–dimensional state space with a set of points. b) Elementary Voronoi diagram of given points, where all points are equivalent. c) Weighted Voronoi diagram of the same point set, but with weight assigned to each point. Larger area of circles means higher weight, resulting in curvature of some edges. Images were reproduced from [26].

There are many employed applications of Voronoi diagrams in chemistry and molecular biology. Several tools have been developed also for tunnel detection, including CAVER 3.0 [27], MOLE 1.2 [28] and MolAxis 1.4 [29]. These software tools approximate the weighted Voronoi diagrams by substitution of protein atoms with a collection of smaller spheres that have the same radius. More precise and also more numerically stable solutions are achieved with this approximation, but it also requires more computer memory. There are also implementations that directly use weighted Voronoi diagrams [30]. Even though the approximation of weighted Voronoi diagrams requires more memory than the regular Voronoi diagram, it is still less than the memory requirements of grid–based tools.

A great feature of CAVER 3.0 is also the possibility of dynamic protein structure analysis. To detect tunnels in a sequence of snapshots of molecular dynamics, the ordinary Voronoi diagram is computed in each frame. In molecular dynamics, the positions of the atoms change in each frame and their Voronoi diagrams as well, so the correspondences of Voronoi diagrams from the consecutive frames have to be found in order to detect a path through the sequence [27]. Among disadvantages are inaccuracy because of the approximation of weighted diagram and inability to use other than spherical probes.

3.1.3 Sampling–based methods

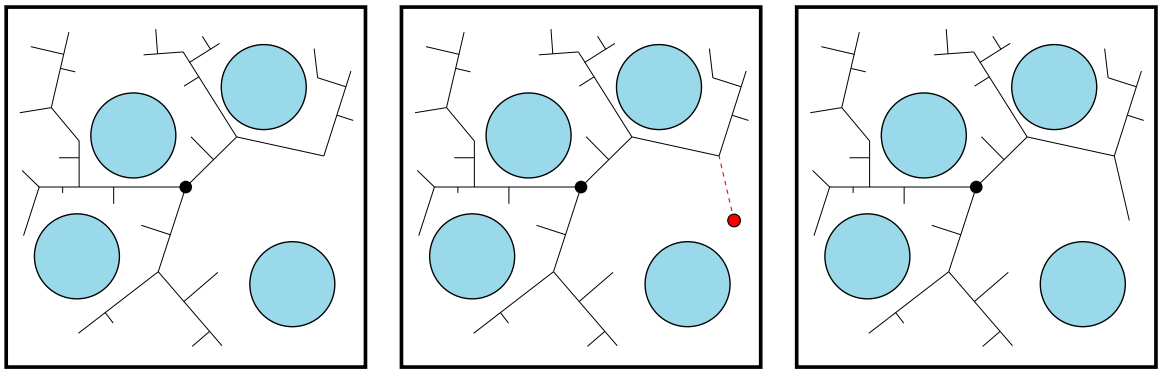
The computation of tunnels in the grid–based and Voronoi–based methods requires an explicit description of the boundary of void space. This is represented as the surface of spheres representing the individual atoms. Such an explicit representation allows us to analytically compute the distance to the boundary, which is necessary for the construction of Voronoi diagrams. However, this explicit representation can be used only in the case of spherical probes.

If a tunnel has to be found for a non–spherical probe, e.g., a small molecule, the above mentioned methods cannot be used. To detect tunnels for a non–spherical molecule, it is necessary to consider its shape, and consequently, it is also necessary to consider additional degrees of freedom describing rotation of the molecule. Motion planning methods known from robotics can be used to solve this problem.

The complex problem of motion planning is one of the most studied fields in robotics [8]. There are several approaches, how scientists and engineers tackle this problematics that can be generally divided into two groups. First, the complete meth-

ods that focus on exact representation of the configuration space. This solution ensure completeness, but unfortunately these methods are mathematically and computationally prohibitive to derive for high–dimensional problems. Since there is a high demand for algorithms that can handle high–dimensional real world problems, there is a great interest in second group, the sampling–based methods.

The main idea of sampling–based methods is the separation of the motion planning task into several steps. Firstly, a random or deterministic function chooses a sample from the configuration space. Then, another function determines the nearest previous free–space sample. After that, it is checked if the sample is in a free configuration (i.e., with no collision with obstacles) and finally both of these points are connected. The whole process repeats and a graph, or tree, is gradually being built. Rapidly–exploring Random Tree (RRT) is an exemplary sampling–based method [31]. The RRT algorithm is schematically shown in Figure 3.2.



a) b) c)

Figure 3.2: Individual steps of the RRT algorithm in two dimensions. a) Rapidly–exploring Random Tree after several iterations. Blue circles represent obstacles, black dot is the root of the tree. b) A random sample is generated (red dot), the nearest node from the tree is determined and also free configuration is checked. c) Since there is no collision, new node is added to the tree.

In the field of tunnel detection in proteins, several tools were implemented with sampling–based methods [32, 33]. Solutions that employ the sampling–based principle can plan the path with an arbitrary probe shape, while other methods are limited only to spherical probes (for example Voronoi diagrams–based methods). These methods also require less memory when compared to grid–based methods. On the other hand, if a solution to a given task does not exist, the sampling–based algorithms can run forever and the user needs to define time–dependent constraints. Another downside of sampling–based methods is the variability of results due to the random nature of

the approaches. If the task of these algorithms is the tunnel detection in the protein structure, the number and properties of found tunnels may differ in every single computation.

3.1.4 Surface–based methods

Another group of tunnel detection methods defines the protein surface by a specific solvent or ligand. The most common surface representation is generated with water as the solvent that determines the surface features. A frequently used surface representations by researchers are solvent excluded surface, solvent accessible surface and ligand excluded surface [34]. All of these representations are based on a similar principle, which is the detection of protein parts that can be accessed by given solvents or ligands.

One of the developed surface–based tools is Travel depth [35]. The idea of tunnel detection of this method is the calculation of physical distance (the travel depth) that a solvent molecule would travel from the protein surface to a predefined reference surface (e.g., the convex hull). This approach is especially suitable for detection of surface pockets or tunnel entrances.

3.2 Ligand molecule path planning

Another group of computational approaches tries to predict how ligands bind or dock to macromolecular targets. These methods also want to screen thousands of distinct compounds to find possible novel binding partners. The computational ligand docking and screening is an already established method in the pharmaceutical industry to accelerate the drug discovery process and choose promising drug candidates from large collections of virtual compounds that could bind target proteins [36].

Molecule planning techniques aim to utilize algorithms from motion planning to the biological domain. These methods try to find a collision–free path between the binding site of protein and the exterior of the given protein.

The main challenge of molecule planning and docking experiments is the dynamic nature of both protein and ligand, because when these two substances interact, a number of structural changes within the ligand binding site might occur. This flexibility of binding site is also the reason, why many diverse molecules can sometimes bind to the same place.

3.2.1 Protein–ligand docking

Computational docking experiments can be used to predict bound conformations of small ligands to proteins or other macromolecules [37]. This research technique is often used for the study of molecular interaction and reaction mechanisms, and it is the basis of structure–based drug design. Several tools have utilized docking computations, including AutoDock Vina [38], rDock [39] or EADock [40].

AutoDock Vina is a software tool based on a scoring function and fast gradient–optimization conformational search [37]. In other words, the scoring function evaluates how well a given ligand conformation binds to the protein. The goal of the optimization task is to find a global minimum of the scoring function. It should be mentioned that the docking generally assumes much or all of the protein rigid, because dynamic calculations are yet too complex to accomplish. Ligands are treated as flexible compounds with 0–32 active rotatable chemical bonds. Results of AutoDock Vina docking and comparison with structures of protein–ligand complexes obtained via X–ray crystallography are presented in Figure 3.3.

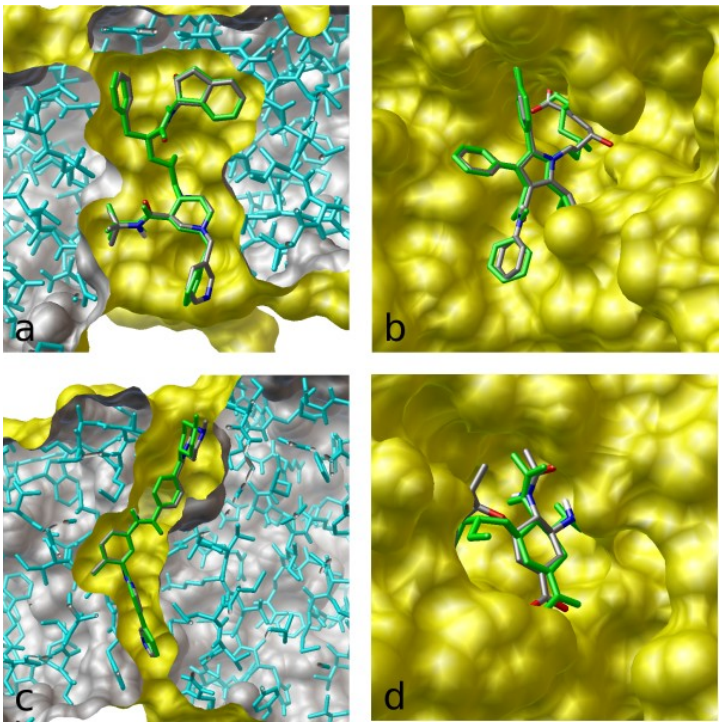


Figure 3.3: Results from AutoDock Vina docking software tool. Flexible docking is illustrated with green color of ligands, while grey color of ligands are conformations obtained by X–ray crystallography. Ligands are a) indinavir, b) atorvastatin, c) imatinib and d) oseltamivir bound to their respective targets (HIV–1 protease, HMG–CoA reductase, Bcr–Abl tyrosine–kinase and influenza neuraminidase). The results show great agreement of both computational and experimental results. Reproduced from [41].

3.2.2 Ligand path planning

Although the transportation of ligand into the functional site of protein is usually a complex process, it is still driven by geometric properties [42]. For this reason, it is possible to simulate the ligand path from surface of protein to its binding site (or the other way around). Only geometrical computation is a great simplification model, but it can serve as a starting point for further development of more complex simulations. This basic approach can help researchers to find the most interesting parts of ligand trajectory and understand its behavior.

One of the already published tools for ligand path planning is MoMA–LigPath [43]. It is a sampling–based method that applies a robotic–inspired algorithm to explore a given configuration space and it also considers partial flexibility of the ligand and side chains of protein. As a purely geometric approach, it enables to simulate ligand unbinding from the protein within short computing time, which is one of this method's advantages. Results from this model can serve as a first approximation that can be further refined by standard molecular modelling techniques. The MoMA–LigPath tool also has more sophisticated algorithmic variants that take protein backbone flexibility and energy models into account [44, 45]. Whilst these modified methods can provide more trustworthy simulations, it is at the expense of additional computational costs. Figure 3.4 shows result of phenol unbinding from insulin hexamer computed by MoMA–LigPath.

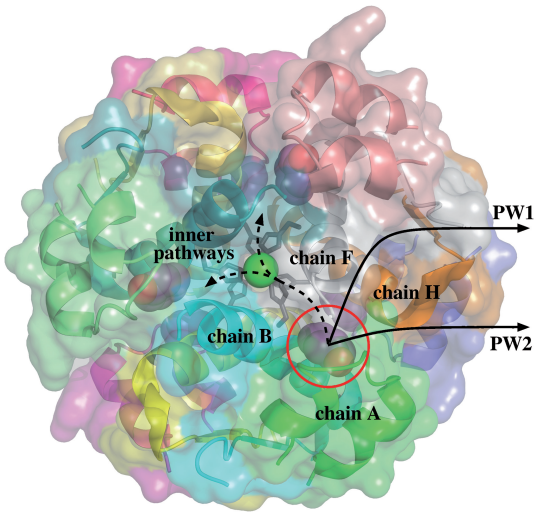


Figure 3.4: Results of MoMA–LigPath software tool in phenol unbinding from insulin hexamer. In most simulations, the phenol molecule (in red circle) exited via pathway 1 or pathway 2 (PW1, PW2). In few simulations, ligand diffuses inside the hexamer before finding an exit pathway (dashed line). Reproduced from [43].

3.3 Conclusion

There are already several works, which discuss the employment of the sampling–based RRT method in the analysis of protein tunnel traversability by non–spherical ligands. Among them is a article written by Vonásek and Kozlíková [10], which introduces a novel method of tunnel detection in dynamic protein structures and also suggests non–spherical ligand utilization for more realistic results. This approach would increases the computational cost, because sampling with non–spherical ligand needs to be performed in the 6D configuration space in each frame of the dynamic protein structure. The authors propose a novel sampling method, which would generate the random samples around the previously detected spherical tunnels to decrease the computational demands.

Another work that describes utilization of the RRT method for ligand path planning, is [9]. The author designs and implements a RRT–inspired method for the detection of paths inside proteins for spherical ligands and also presents the results from motion planning of such ligand from the inside of a large protein.

Although many different research approaches were used to create various software tools, the problematics of tunnel detection in proteins and computations related to ligand binding are still unresolved. During the research in this field, it turned out that the intuitive geometrical approach is not sufficient to model the great complexity of macromolecules, but it can be a good starting point to development of more sophisticated solutions. This thesis focuses on implementation of sampling–based method that can be used to detect tunnels and also simulate passage of small rigid molecules through the found tunnels.

Chapter 4

Solution outline

For the purpose of this diploma thesis, several guidelines were set. Three main tasks have been stated for the practical part: implementation of a method for tunnel detection in static protein structures, adaptation of this method for tunnel detection in dynamic protein structures, design and implementation of a novel method for analysis of ligand traversability through static tunnels. All the solutions are based on Rapidly–exploring Random Trees, a sampling–based motion planning algorithm (the basic principle of this algorithm is presented in section 3.1.3).

RRT was chosen due to its ability to cope with objects of arbitrary shape and the ability to find paths in high–dimensional systems. Both features are necessary for tunnel detection and molecular path planning. The inputs for this method are the protein pdb file, parameters of ligand and properties of the tree. Output of this method is a constructed tree and detected tunnels.

Since the task of tunnel detection has some specificities, the Rapidly–exploring Random Tree algorithm cannot be applied directly, but certain modifications need to be made. The modified RRT algorithm was implemented in C++ programming language, while Python programming language was used for scripting and supporting calculation purposes. The modifications and implementation details of all solutions are presented in the upcoming sections.

4.1 Tunnel detection in static protein structures

The original RRT algorithm is not designed to detect multiple tunnels (i.e., generally paths) in protein structure. It was rather developed to find a single path between two precisely defined places. For the purpose of the tunnel detection, the task needs to be

redefined, because finding a path to an explicitly defined goal could noticeably limit the performance speed, which is undesirable. Given the quite simple nature of the RRT algorithm, it can be modified for our needs.

The basic idea of the solution is to built a tree from the functional site of protein, reach the surface of this protein with a spherical probe and detect the tunnels that lead from the functional site. Results from the proposed RRT algorithm for tunnel detection in static protein structures were compared to the results of CAVER 3.0 on the same protein structures. The original RRT algorithm leaves the programmer with many choices — how to sample the state space, how to define the "nearest" node (especially in higher dimensions) and how to plan the motion from the nearest node to newly generated nodes. Even a small change of the sampling method can yield a dramatic change in the running time of the planner [8]. Furthermore, the protein domain brings another challenges and so, there are several subproblems that need to be resolved:

- Selection of a sufficient sampling region that is necessary for successful tunnel detection
- Employment of a technique for protein surface detection
- Preventing the tree to build far outside the protein structure
- Detection of tunnel endpoints
- Designing a method for tunnel comparison, which is used to determine if protein tunnels are identical

The above mentioned issues are discussed in the following sections and the solution for each of them is explained. In this diploma thesis, we have proposed solutions for all of these challenges and developed a method, which successfully detects tunnels in static protein structures. Figure 4.1 on the next page shows the pipeline that was proposed for the static tunnel detection.

4.1.1 Input parameters and sampling region

There are several input parameters that are necessary for the task of tunnel detection. First of all, it is obviously the information about protein atoms. For this purpose, data from Protein Data Bank were used for XYZ coordinates of atoms and each atom was assigned with its van der Waals radius. The next two parameters describe the initial coordinates for the RRT algorithm and the radius of spherical probe that is used for tunnel detection. Another parameter is used for description of the protein surface (see section 4.1.3) and the last input parameter is a list of RRT parameters

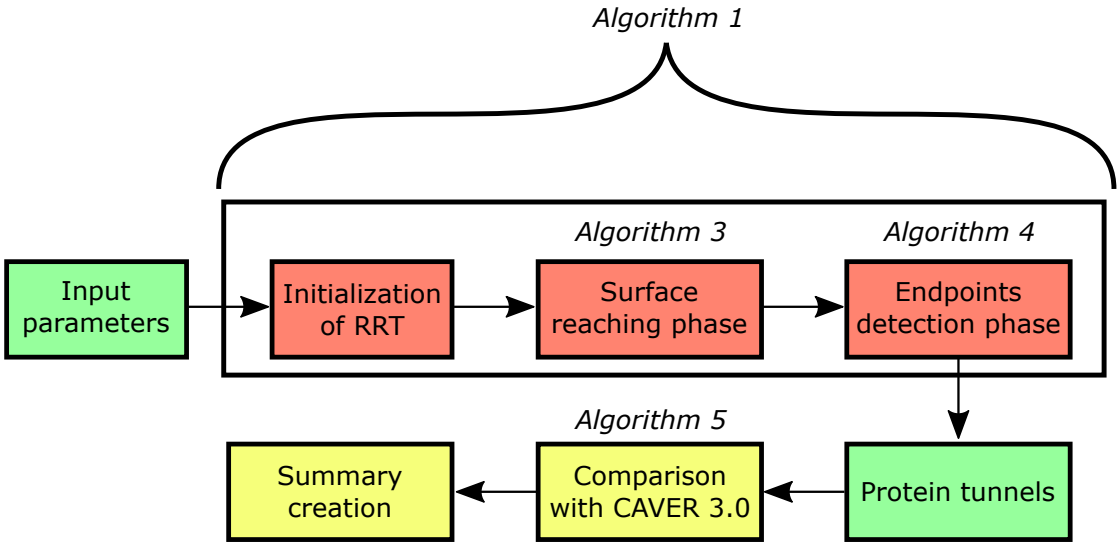


Figure 4.1: Proposed pipeline for the tunnel detection in static protein structures. Algorithms 1, 3, 4 and 5 correspond to individual elements of the pipeline and the pseudocode of each algorithm is presented in this chapter.

(e.g., number of nodes in the tree, the distance between individual nodes).

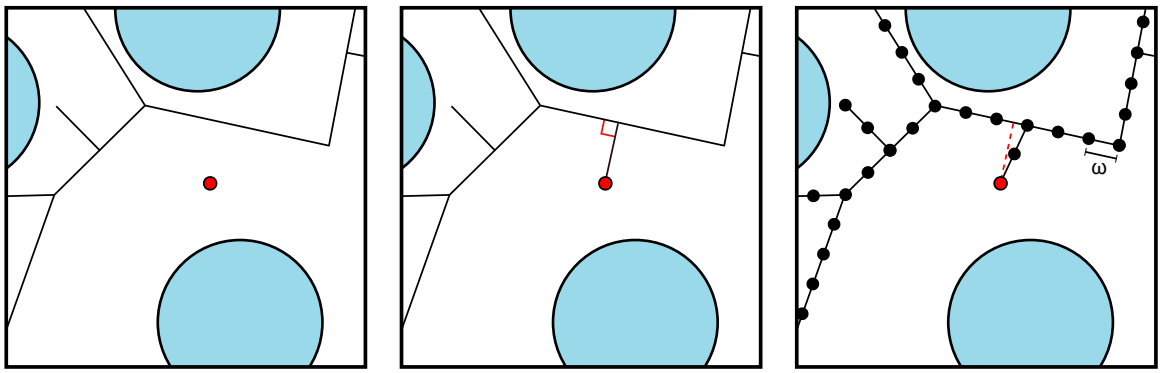
In order to detect tunnels, choice of the sampling region is very important. The sampling region must be big enough to contain the whole protein structure, but not too big, because it is useless to generate samples that are far away from the protein. If the sampling region is smaller than the protein itself, it is obviously unfeasible to reach its surface and detect the tunnels. At the beginning of the RRT algorithm, it is also desirable to generate most of the random samples within the protein structure, because it would be usually impossible to connect node from outside of the protein structure with a node that is inside.

As the protein atoms are being loaded to the algorithm, minima and maxima of XYZ coordinates are saved. In order to let RRT reach the protein surroundings, minima and maxima are extended by 8 Å (1 Å = 10^{-10} m; proteins can be approximated by a sphere with diameter usually in tens or hundreds of Å), which creates the sampling region. After the sampling region is defined, both surface reaching phase and endpoints detection phase use it to get random samples.

4.1.2 RRT implementation for tunnel detection

The original RRT algorithm is in theory designed to directly add the newly generated random sample and the searching of the nearest point is intended as finding the closest point of the tree. For practical purposes, a line connecting the new random

sample and its closest node in the tree is segmented to several small parts, which serves as an approximation, so the closest point of the RRT can almost be detected. Both theoretical and practical approaches are described in Figure 4.2.



a) b) c)

Figure 4.2: Addition of new point to RRT. a) A part of RRT tree, the red point is the newly generated random sample from the sampling region, blue circles represent obstacles. b) Theoretically, the new point is connected to the already created tree with a perpendicular line that leads to the closest part of the tree. c) The practical approach approximates theoretical behavior by not directly adding the new random point, but by incremental addition of new points on the line with a given distance (shown as ω in the image). The deviation from theoretical approach is shown as red dashed line.

The main advantage of the practical approximation is a noticeable decrease in computational cost, because the nearest neighbor can be found with libraries that are directly developed for this purpose. In this thesis, *MPNN* library was used for the nearest neighbor search [46]. This library utilizes a k–d tree. With this representation, the time complexity of finding the nearest neighbor in a tree with n nodes takes $\mathcal{O}(\log\ n)$ time on average case.

One of the most computationally demanding part of the sampling–based planners is the collision detection. As it is performed for each newly created node, the speed of the collision detection strongly influences the overall time performance of the planners. The classical $\mathcal{O}(n)$ checking of collisions between a given sphere (probe) and all protein atoms, would be too time consuming.

A fast collision detection requires more sophisticated data structures such as the Oriented bounding boxes (OBB) trees. The principle of OBB trees is to create a hierarchical, binary–tree representation of the objects, in this case the protein atoms. Each node in the tree contains either a bounding box or a list of atoms. The root of the tree contains the bounding box of the all proteins, and its two children contain smaller bounding boxes that are contained in the first one. The leafs of the tree contain either

single atoms or a list of few atoms.

The collision detection between this structure and a given sphere (probe) then starts by checking collision with the bounding box of the root node. If there is no overlap between the sphere being tested and the bounding box, there is definitively no collision and the algorithm terminates. Otherwise, there might be a possible collision, which needs a further examination. Therefore, it is determined in which part of the current bounding box the query sphere lies. The search then continues in the corresponding subtree. This repeats until the search reaches a leave, where collisions between the protein spheres and the query sphere are accurately computed.

The big advantage of this approach is fast computation, as the binary–tree hierarchy can be traversed in $\mathcal{O}(\log\ n)$ time, where n is the number of protein atoms. There are many methods for building the hierarchical collision detection trees. In this work, we employed the *OZCollide* [47] library, which has been shown to provide very fast collision detection between spherical objects [9].

Pseudocode of the RRT algorithm that was used for the static tunnel detection is presented in Algorithm 1 on the next page. In the first two steps, the k–d tree (τ) and collision detection structure are initialized based on user input parameters (initial coordinates q_{init} and protein coordinates). The algorithm then executes two very similar sections.

In the first section, new nodes are added until the user–specified number of nodes limit (S) for the surface reaching phase is achieved. The algorithm also check if the tree is growing in function *isTreeGrowing()*. If the newly generated nodes from the sampling region have many collisions in a row and therefore, no nodes are added to the tree, it indicates that the tree already explored the paths in protein structure and the loop is terminated. The idea of the algorithm is to generate a new random node q_{rand} from the sampling region, then finding the nearest node q_{near} from the tree and successive addition of new nodes q_{new} to the tree. Function *newNodePosition()* provides coordinates for the new node q_{new}, which is on the line between q_{rand} and q_{near} and in distance ω from node q_{near}. The new nodes are then also added to the tree at the distance ω from previous parent node in function *surfaceReachingPhase()*. The second section is almost identical, only the iteration limit (T) and the function that adds new nodes differ. Both of these functions, which add new nodes to the tree, are discussed in detail in the following paragraphs.

Algorithm 1: RRT – static protein tunnel detection pseudocode

Input: protein coordinates, surface spheres, initial configuration, RRT parameters

Output: protein tunnels

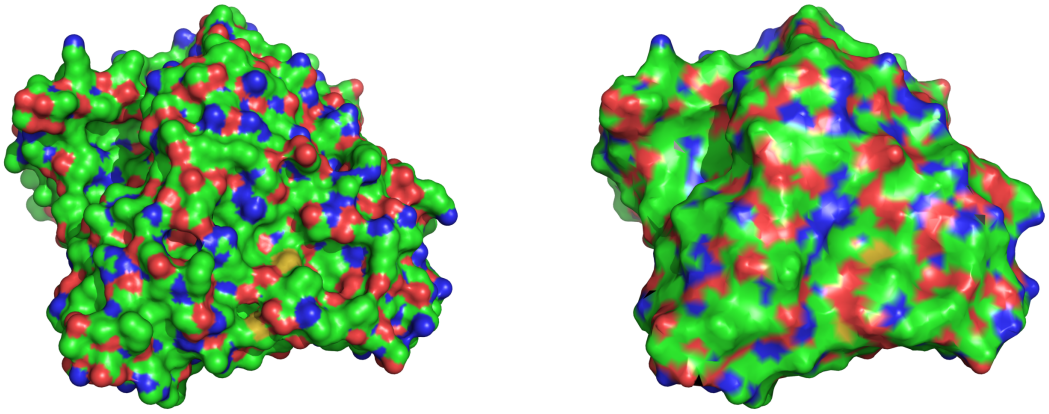
```
1  τ.init(q_init);
2  proteinAtoms ⟵ setFromCoordinates(proteinCoordinates);
3  while i++ < S and isTreeGrowing() do
4      q_rand ⟵ randomCoordinates(samplingRegion);
5      q_near ⟵ nearestNeighbor(q_rand, τ);
6      q_new ⟵ newNodePosition(q_rand, q_near, ω);
7      surfaceReachingPhase(τ, proteinAtoms, surfaceSpheres, q_rand, q_new);
8  while j++ < T and isTreeGrowing() do
9      q_rand ⟵ randomCoordinates(samplingRegion);
10     q_near ⟵ nearestNeighbor(q_rand, τ);
11     q_new ⟵ newNodePosition(q_rand, q_near, ω);
12     endpointsDetectionPhase(τ, proteinAtoms, q_rand, q_new);
```

4.1.3 Surface reaching phase

There are several ways, how the surface of proteins can be represented in structural bioinformatics computations. One of the common approaches is the concept of so–called α–shape, which describes the surface as a set of points, which can be achieved by a sphere of a given radius that rolls on the protein atoms.

For the purpose of this thesis, computer program PyMOL was used to create the surface representation of a given protein. PyMOL can save the surface as a set of points, which are generally a specification of a 3D polygon. It is also possible to change the radius of the sphere used for the surface creation. This radius was usually set to 4–5 Å, which gives a lower surface resolution, but it prevents the sphere from entering the protein, which would complicate tunnel endpoints detection. Comparison of surface representation by solvent with radius 1.4 Å and 4 Å is presented in Figure 4.3 on the facing page.

A function (Algorithm 2 on the next page) was developed within this thesis that takes the set of points that describe the surface (∼20 000 points) and returns a set of spheres with a selected radius (∼600 spheres for radius of 3 Å). The function gradually



a) b)

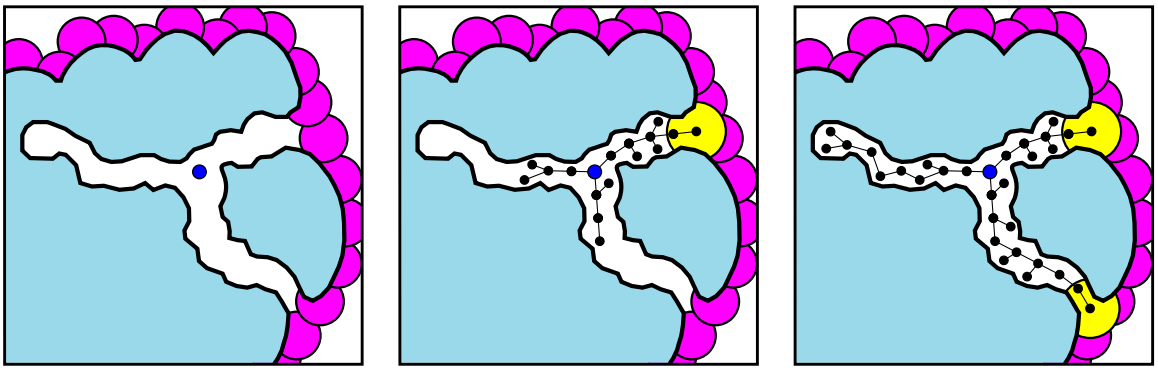
Figure 4.3: Comparison of PyMOL surface representation by solvent with radius a) 1.4 Å (represented by ∼145 000 points) and b) 4 Å (represented by ∼23 000 points).

adds spheres to *surfaceSphereSet*, if the selected point (a possible center of new sphere) lies more than 3 Å away from the nearest already added sphere. The lower the radius, the more spheres are returned. As a rule of thumb, the radius was most often set to 3 Å, which served as a good surface representation, while the number of spheres was not too high. The returned sphere set is then used as one of the input parameters of RRT.

Algorithm 2: Function that returns spheres for the surface representation

```
1 Function surfacePoints2surfaceSpheres(pointSet, thresholdRadius = 3 Å)
2     surfaceSphereSet.init(pointSet[0]);
3     foreach point p in pointSet do
4         q_near ⟵ nearestNeighbor(p, surfaceSphereSet);
5         if distance(q_near, p) > thresholdRadius then
6             surfaceSphereSet.add(p);
7     return surfaceSphereSet;
```

In the surface reaching phase (schema is shown in Figure 4.4 on the following page), RRT starts to build from the initial configuration q_{init}. As the tree reaches the surface, which is checked as a collision with spheres from *surfaceSphereSet*, it needs to be prevented from extending out of protein structure, since it is useless to build the tree in the protein's surroundings. Furthermore, it is desirable to let the RRT algorithm to further explore paths within the protein. This behavior is achieved by addition of artificial spheres that serve as plugs for possible tunnels.



a) b) c)

Figure 4.4: The surface reaching phase. a) The initial configuration. Protein is represented in light blue, magenta circles represent the surface, blue dot is the root of tree. b) After several iterations, the surface was reached and an artificial yellow circle is added to prevent RRT from exploring its surroundings and also to let RRT further explore the paths inside the protein. c) The end of the surface reaching phase. Another plug was added and the limit of total nodes was reached.

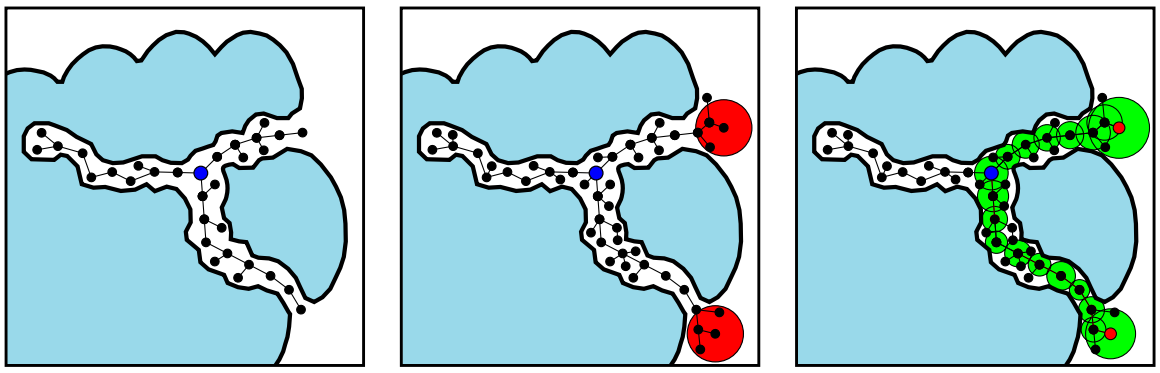
Algorithm 3 describes the function *surfaceReachingPhase()*. While the distance between nodes q_{rand} and q_{new} are greater than ω (a user–defined constant), two conditions are checked. The first is the collision with protein atoms. If the node q_{new} with probe radius collides with protein atoms, the loop execution is stopped, otherwise q_{new} is added to the RRT. Then, collision with surface spheres is checked and if q_{new} collides with the surface spheres, a plug is added by function *addPlug()* at the position of q_{new} and the loop is terminated.

Algorithm 3: Surface reaching phase

```
1  Function surfaceReachingPhase(τ, proteinAtoms, surfaceSpheres, q_rand, q_new)
2      while distance(q_new, q_rand) > ω do
3          if proteinAtoms.collide(q_new, probeRadius) then
4              break;
5          else
6              τ.add(q_new);
7          if surfaceSpheres.collide(q_new, probeRadius) then
8              addPlug(q_new);
9              break;
10         q_new ⟵ newNodePosition(q_rand, q_new, ω);
```

4.1.4 Endpoints detection phase

The next step is the endpoints detection phase, where the plugs from the previous phase are removed and tunnel endpoints are being searched. As the RRT expands out of the protein structure, the tunnel endpoint is detected, if a sphere with radius of 2 Å does not have collision with the protein atoms. The whole process repeats until the user–defined limit of total node count (T) is reached. Tunnels are then traced back from tunnel endpoints to the root of the tree. Figure 4.5 schematically shows the endpoints detection phase.



a) b) c)

Figure 4.5: Endpoints detection phase. a) RRT configuration from surface reaching phase without the plugs. The root is depicted as the blue dot. b) The algorithm continues to explore protein structure and it also expands out of the protein. If a red circle of a given radius (2 Å was used in the implementation) can be inserted without collision with protein atoms, tunnel endpoint is detected. c) Tunnels are then produced by tracing from tunnel endpoints to the root of the tree. Found tunnels are showed in green, tunnel endpoints are the red dots.

The function that detects tunnel endpoints (Algorithm 4 on the next page) is very similar to to the surface reaching function. New nodes are still added to the tree with the same technique and the same collision with protein atoms is performed. The difference is that the surface is not detected, but it is rather constantly checked, if a sphere with radius of 2 Å (*limitRadius*) can be inserted without any collision with the protein atoms. If this condition is fulfilled, a tunnel endpoint is detected and the loop execution is stopped.

Algorithm 4: Endpoints detection phase

```
1  Function endpointsDetectionPhase(τ, proteinAtoms, q_rand, q_new)
2      while distance(q_new, q_rand) > ω do
3          if proteinAtoms.collide(q_new, probeRadius) then
4              break;
5          else
6              τ.add(q_new);
7          if ¬proteinAtoms.collide(q_new, limitRadius) then
8              tunnelEndpoints.add(q_new);
9              break;
10         q_new ⟵ newNodePosition(q_rand, q_new, ω);
```

4.1.5 Comparison metric

In order to determine if two tunnels that were detected by different methods are identical, some sort of comparison metric needs to be employed. The tunnels are usually represented as a set of spheres that connect a place inside the protein structure (e.g., the root of tree in the RRT method) and the protein surface. The length of individual tunnels can differ and so, the number of spheres representing different tunnels can vary as well.

In this thesis, the following method for tunnel comparison was utilized. Let us have two tunnels that are represented as a sequence of spheres. For each sphere from *tunnel 1*, the closest sphere from *tunnel 2* is found, distance of these two spheres is calculated and the average from all distances is determined. The whole process repeats, but now for each sphere from *tunnel 2*, the closest sphere from *tunnel 1* is found. Both approaches can give different results. The maximum result from both computations is selected and this number is used for tunnel comparison. If the result is smaller than 3.5 Å, the tunnels are labeled as identical. This threshold was chosen empirically. The pseudocode of function that computes the comparison metric is shown in Algorithm 5 on the facing page.

Algorithm 5: Comparison metric

```
1 Function compareTunnels(tunnel_1, tunnel_2)
2     firstResult = evaluateMetric(tunnel_1.spheres, tunnel_2.spheres);
3     secondResult = evaluateMetric(tunnel_2.spheres, tunnel_1.spheres);
4     return maximum(firstResult, secondResult);
5 Function evaluateMetric(spheres_1, spheres_2)
6     foreach sphere s in spheres_1 do
7         nearestSphere = s.findNearestSphere(spheres_2);
8         allDistances.add(distance(s, nearestSphere));
9     return average(allDistances);
```

4.2 Tunnel detection in dynamic protein structures

As noted in the previous chapters, proteins are dynamic structures and the atoms constantly change their positions. It is therefore important to develop algorithms that can detect tunnels not only in static protein structures, but also in dynamic protein structures. As the protein molecule is moving in time, its tunnels are changing as well [48]. A tunnel that seems to be wide in a random instance of time might in reality be open just for a short period of time, which could imply that the tunnel is biochemically unimportant.

In practice, the computation of static tunnels is performed on a single pdb file, while the tunnel detection in dynamic structures has to be performed on multiple pdb files that can be considered as snapshots from a given time interval. Individual snapshots are usually generated by computer simulation from a pdb file that was obtained by X–ray crystallography.

4.2.1 Pipeline for tunnel detection in dynamic structures

In this thesis, we have modified the proposed RRT–based technique, which was used for the tunnel detection in static protein structures, to detect tunnels in dynamic protein structures. The basic principle is to search several times for static tunnels in each provided pdb file (each snapshot). Ten runs per snapshot were used in this case. After the static tunnels are detected in one frame, all found tunnels from these

runs are successively compared to each other with the metric described in chapter 4.1.5. If a tunnel is identified in more than 50 % of the runs, it is marked as an actually detected tunnel for the frame. The same process is then performed on the next snapshot. The pipeline for the tunnel detection in dynamic protein structures is presented in Figure 4.6.

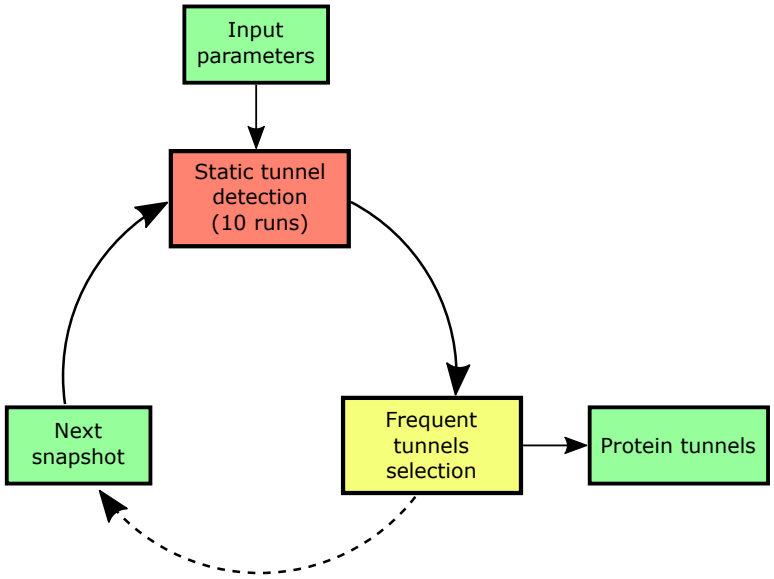


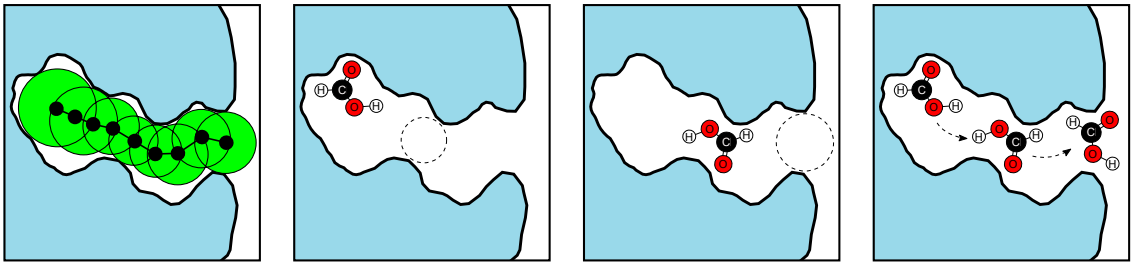
Figure 4.6: Pipeline for tunnel detection in dynamic protein structures.

This approach to the tunnel detection in dynamic structures assumes that tunnels are present in multiple snapshots. The number of runs per snapshot is a parameter that can be easily altered. For each snapshot, the surface of protein was represented by the set of spheres, which were generated with the help of computer program PyMOL (see section 4.1.3 for details).

4.3 Passage simulation of molecules through tunnels

The simulation task of molecule passage through protein tunnels was also approached as a motion planning problem. The molecule is represented as a set of hard spheres and a collision–free path through the protein tunnel is searched. Three dimensions, the spatial XYZ coordinates, were used for the detection of tunnels in both static and dynamic protein structures. For the task of passage simulation, another three dimensions need to be introduced — rotation angles α, β, γ. These three angles can describe rotation of a molecule in the state space and so, simulation of molecule passage becomes a six–dimensional problem.

Solution, which was implemented for this task, can be separated into two parts. Firstly, for a given protein structure, the tunnels are detected (chapter 4.1 describes the process). One of the resulting tunnels is selected for the second part — the passage simulation itself. In the second part, a ligand molecule is guided through the tunnel from the core of protein to its surface. An illustration of passage simulation is shown in Figure 4.7.



a) b) c) d)

Figure 4.7: Simulation of passage of a small molecule through a protein tunnel. a) Result of tunnel detection in static protein structure — the first part of the task. b) In this illustration, formic acid is at the protein's binding site, where the simulation starts. The dashed sphere represents the current sampling region. c) A later phase of the algorithm – as the molecule changes position, another sphere is used as the sampling region. d) The result of motion planning. The molecule gradually shifts through the tunnel and it also rotates on the way. In this example, it successfully passes the tunnel.

Pseudocode for the function that performs the passage simulation is presented in Algorithm 6 on the next page. The principle is similar to the already proposed RRT implementations and utilizes a guided RRT–path approach, which was developed by Vonásek et al. [49]. This function employs the tree, protein atoms, sequence of spheres that represent tunnel and the molecule as parameters. As the first step of the algorithm, only protein atoms that are close to the tested tunnel are used for collision detected, since distant atoms cannot collide with the analyzed molecule in the case of guided approach. During the run of this algorithm, each sphere is subsequently used as the sampling region and at the same time, each sphere progressively serves as the goal region that needs to be reached.

Until the last sphere of the tunnel is reached, random sample is generated from currently first sphere in the sequence. This sample has six dimensions (X, Y, Z, α, β, γ), the spatial coordinates XYZ are randomly generated from the sampling region. The rotation angles α, β and γ are based on rotation angles of the parent node, the equation for angle α is: $\alpha = \alpha_{parent} + U(-0.2, 0.2)$, where U denotes the uniform distribution. Angles β and γ are calculated analogously. The nearest point is then

found and new node is created in the distance ω. The inner loop checks if protein atoms collide with the molecule, whose coordinates are determined by q_{new} (it represents the center of molecule). After the inner loop adds the last point, it is checked, whether the currently first sphere in the tunnel sequence is reached and if this condition is fulfilled, this sphere is poped from the list. Since there is a possibility that the ligand cannot reach the last sphere from the sequence, a time–dependent constraint needs to be employed to prevent infinite looping.

Algorithm 6: Simulation of passage of molecule through a tunnel

```
1  Function simulatePassage(τ, proteinAtoms, tunnelSpheres, molecule)
2      proteinAtoms ⟵ selectAtomsNearTunnel(proteinAtoms);
3      while ¬tunnelSpheres.empty() do
4          currentSphere ⟵ tunnelSpheres.getFirstSphereInSequence();
5          q_rand ⟵ randomCoordinatesInSphere(currentSphere);
6          q_near ⟵ nearestNeighbor(q_rand, τ);
7          q_new ⟵ newNodePosition(q_rand, q_near, ω);
8          while distance(q_new, q_rand) > ω do
9              if proteinAtoms.collideWith(molecule, q_new) then
10                 break;
11             else
12                 τ.add(q_new);
13             q_new ⟵ newNodePosition(q_rand, q_new, ω);
14         if isSphereReached(currentSphere) then
15             tunnelSpheres.pop();
```

In practice, the small molecule is described in a pdb file by local coordinates. During the geometrical analysis of passage, the node q_{new} contains the global coordinates that define position of the molecule within the protein structure. Every time a collision check of protein atoms with the molecule is performed (line 9 in pseudocode), the molecule is first rotated by angles α, β, γ and then translated by X, Y, Z coordinates of node q_{new} before each sphere from the molecule is tested for collision with protein atoms. The three–dimensional rotation that is calculated by the rotation matrix is shown in the following equation:

$$\begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_R \\ y_R \\ z_R \end{pmatrix}$$

The rotated coordinates (x_R, y_R, z_R) are obtained by multiplication of the rotation matrix by local spatial coordinates (x, y, z). Angles α, β, γ are expressed in radians. Each sphere needs to be first rotated and only then translated, because the rotation is performed with respect to the coordinates of origin.

Chapter 5

Results

Algorithms and methods, which were designed in the previous chapter, were implemented and successfully utilized for tunnel detection in both dynamic and static protein structures and also for passage simulation of different molecules through found tunnels. Results of the tunnel detection in static protein structures are presented in section 5.1. Comparison with CAVER 3.0, an already established software tool for tunnel detection, is also provided. The CAVER tool was used in its default configuration. Sections 5.2 and 5.3 respectively present results of tunnel detection in dynamic protein structures and simulation of molecule passage through tunnels.

5.1 Tunnel detection in static protein structures

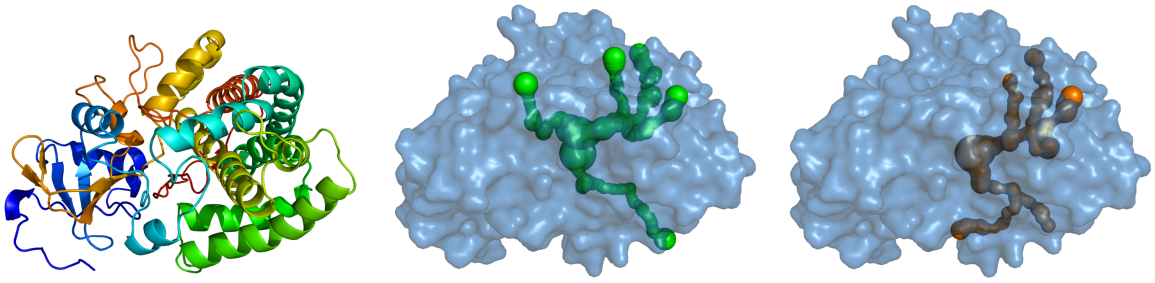
Several protein structures were used for the computation of static tunnels. The RRT implementation that was developed for this task successfully found tunnels in all provided protein structures. Results from tunnel detection are shown in following sections, which are named according to pdb filenames from Protein Data Bank website.

In order to compare results from the RRT implementation with CAVER 3.0 (in text also referred to simply as CAVER), the RRT algorithms were run 100 times on each protein structure. Given the stochastic nature of RRT, the results vary with each run, while CAVER 3.0 is a deterministic method. All computations were executed with probe radius of 0.9 Å. Every image presented in this section was created by computer program PyMOL.

5.1.1 1AKD

Figure 5.1 shows results from tunnel detection in protein 1AKD, its structure and also tunnels found by CAVER 3.0. The RRT implementation found five protein tunnels in this particular run and CAVER 3.0 detected also five tunnels. Based on the comparison metric (designed in section 4.1.5), four out of five tunnels were identical. In the RRT algorithm, the limit for surface reaching phase (S) was 10 000 nodes and for endpoints detection phase (T) 5 000 nodes. The distance between individual nodes (ω) was set to 0.2 Å.

During the 100 runs that were executed, the mode number of tunnels was 5, while the extremes were minimal 2 and maximal 9 found tunnels. The overall comparison of results from the RRT implementation with results of CAVER 3.0 is shown in Table 5.1.



a) b) c)

Figure 5.1: Results from the tunnel detection in protein 1AKD. a) Cartoon representation of protein 1AKD. b) Transparent surface representation of protein 1AKD. Green tunnels were found by the proposed RRT algorithm. c) Orange tunnels, which were found by CAVER 3.0.

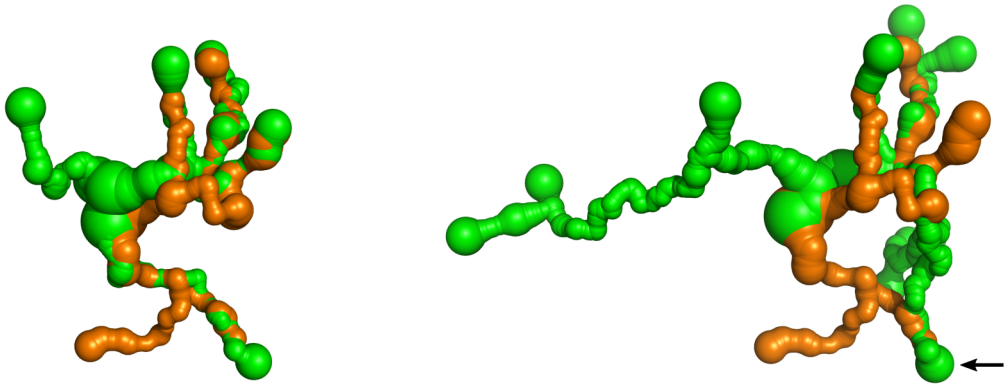
CAVER ID	Length [Å]	Success rate of RRT
1	25.7	37 %
2	25.3	55 %
3	38.2	66 %
4	32.3	20 %
5	50.9	16 %

Table 5.1: Comparison of results from the RRT implementation with results of CAVER 3.0 for protein 1AKD. The RRT algorithm was executed 100 times. Individual columns contain the tunnel index from CAVER 3.0 results, the length of this tunnel (in Ångströms) and the probability of finding the tunnel by the RRT implementation.

The probabilities in Table 5.1 are not very high, despite the RRT implementation usually finds similar count of tunnels as CAVER. To further inspect this outcome,

other results were examined as well as the structure of protein 1AKD. It turned out that this protein has quite sparse structure, which has many different paths that cross each other. This enables the RRT algorithm to take various paths and reach the surface of protein at different positions. With many possibilities how the path can lead and a random sampling process, the algorithm usually takes different paths in individual runs.

Because of the mentioned reasons, it is hard to find identical tunnels as CAVER in each iteration, especially if the tunnels are narrow and with frequent turns, which is exactly the case with tunnels that were found by CAVER in this protein. Although this RRT implementation can also detect narrow and serpentine tunnels, they are often not identical with CAVER's results. So despite finding similar count of tunnels as CAVER, the tunnels found by RRT are not the same in many cases. Figure 5.2 shows overlaid tunnels found by both methods.



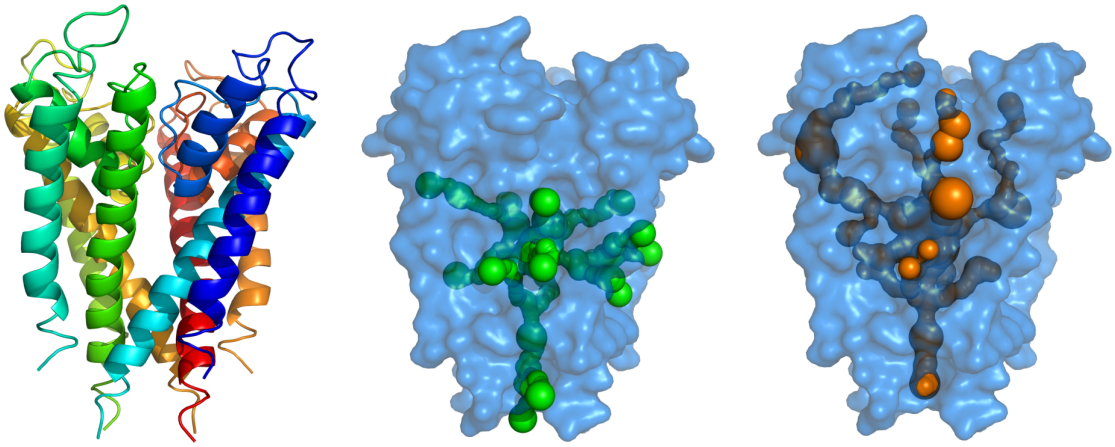
a) b)

Figure 5.2: Comparison of tunnels found by a single run of RRT (in green) and CAVER 3.0 (in orange) in protein 1AKD. a) In this RRT run, both methods found five tunnels and comparison metric marked four out of five as identical. b) In another RRT run, nine tunnels were detected. It can by observed that three tunnels were detected on the left side, where no tunnel was determined by CAVER. Because of the sparse nature of this protein, some of the tunnels (e.g., the one marked by black arrow) take paths, which lead to the right side of the protein, but then end up in the bottom part and therefore, these tunnels can not be identical to tunnels found by CAVER. In this RRT run, no tunnels were marked as identical.

5.1.2 1BL8

The results from tunnel detection in protein 1BL8 are shown in Figure 5.3 together with its structure and also results from CAVER 3.0. In the presented results, RRT found 15 tunnels and CAVER detected 17 tunnels. In this run, 13 tunnels from CAVER were identical to tunnels found by RRT. In the RRT algorithm, the limit parameter S was 10 000 nodes and the limit parameter T was 5 000 nodes. The distance ω was set to 0.2 Å.

For the 100 runs of RRT algorithm, the mode number of tunnels was 15, the extremes were minimal 0 and maximal 25 found tunnels. In order to prevent the RRT algorithm to look for additional tunnels when the structure is already searched, the execution is stopped, if specific number of new random samples have collision with protein atoms. Number of this parameter was set empirically to 2 800 in this case and it should normally speed up RRT execution by skipping the unnecessary searching in already examined structure. Although this approach works well in almost all runs and threshold of 2 800 samples was sufficiently set, in a single run it terminated the RRT before any tunnel was found. Therefore the minimal found tunnels was 0. The overall comparison of results from the RRT implementation with results of CAVER 3.0 is shown in Table 5.2 on the facing page.



a) b) c)

Figure 5.3: Results from the tunnel detection in protein 1BL8. a) Cartoon representation of protein 1BL8. b) Transparent surface representation of protein 1BL8. Green tunnels were found by the proposed RRT algorithm. c) Orange tunnels, which were found by CAVER 3.0.

Results of comparison from tunnel detection in protein 1BL8 by RRT and CAVER 3.0 show much higher finding percentages than in protein 1AKD. The RRT implementa-

CAVER ID	Length [Å]	Success rate of RRT
1	8.2	99 %
2	18.4	81 %
3	19.0	82 %
4	19.0	69 %
5	27.4	85 %
6	26.4	81 %
7	26.3	77 %
8	33.9	83 %
9	19.3	79 %
10	30.5	82 %
11	33.1	73 %
12	33.4	73 %
13	37.1	78 %
14	54.8	52 %
15	57.6	5 %
16	57.6	5 %
17	73.7	0 %

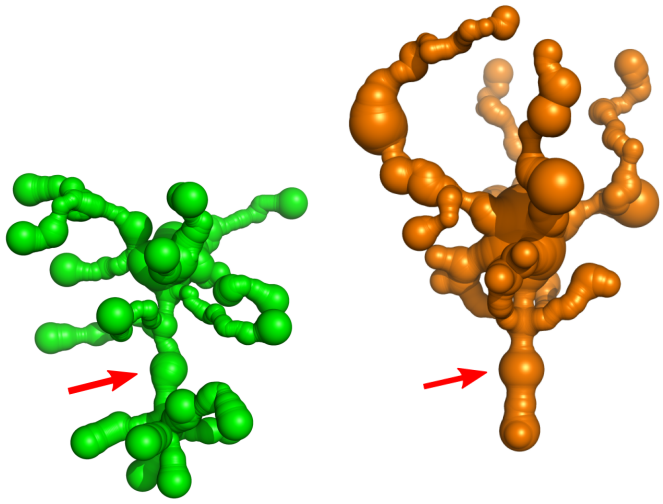
Table 5.2: Comparison of results from the RRT implementation with results of CAVER 3.0 for protein 1BL8. The RRT algorithm was executed 100 times. Individual columns contain the tunnel index from CAVER 3.0 results, the length of this tunnel (in Ångströms) and the probability of finding the tunnel by the RRT implementation.

tion successfully found same shorter tunnels as CAVER, while the longest tunnels were found by RRT only in rare cases. After a further examination of tunnels, it turned out that tunnel with ID 17 from CAVER results is very long and noticeable part of this tunnel is on the surface of protein. The RRT implementation aims to prevent this phenomenon by inserting plugs at tunnel endpoints. Since RRT has not detected this tunnel at all, this approach seems to be suitable.

The position of the initial coordinates was in this case very close to the surface of protein. Since surface is represented as a set of spheres (described in section 4.1.3), even the initial coordinates with probe radius 0.9 Å had collision with the surface, which resulted in addition of plug at initial position and RRT did not add another node. The position of initial coordinates had to be moved several Ångströms in the tunnel. These new coordinates were still quite close to the surface of protein. In several runs, this resulted in expansion of the RRT predominantly at this position and in some

runs and multiple tunnels were found in similar place, while CAVER found only one tunnel there. Figure 5.4 shows results from such situation.

Apart from the fact that RRT can primarily build in close range to initial coordinates, these results also show different techniques in detection of tunnel endpoints. While CAVER determines tunnel endpoints as soon as it reaches tunnel entrances (already mentioned tunnel with ID 17 is just an exception in this case), the RRT implementation detects tunnel endpoints only after the tree expands out of protein structure. This case could be seen in Figure 5.3 on page 58, where tunnels found by RRT reach out further from the protein than tunnels, which were found by CAVER. The different approach to tunnel endpoints detection might also cause RRT to look for tunnels at locations, where other methods already completed searching. The diverse methods in tunnel endpoints detection can be seen in Figure 5.4. In CAVER results, the endpoint position of the single tunnel below initial coordinates serves as the terminus and no other tunnels are detected. On the other hand, the RRT uses this position as a branching site, from which several tunnels are found.



a) b)

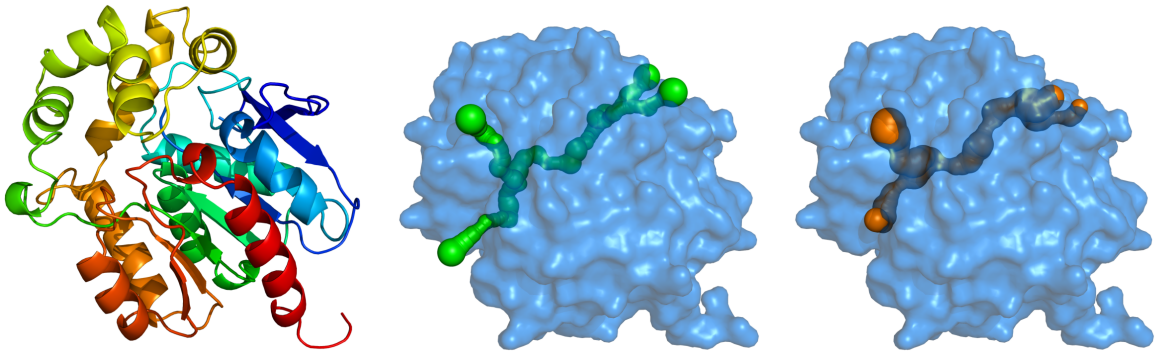
Figure 5.4: Found tunnels from a) RRT and b) CAVER 3.0. The initial coordinates are marked by red arrow. While CAVER found only one tunnel below initial coordinates, the RRT found a total of eight tunnels at the same location in this trial.

5.1.3 1CQW

Figure 5.5 on the facing page shows results from tunnel detection in protein 1CQW, its structure and also tunnels found by CAVER 3.0. In the presented results, the RRT implementation found four protein tunnels and CAVER 3.0 detected also four tunnels.

All the tunnels were identical based on the comparison metric. In the RRT algorithm, the limit parameter S was 10 000 nodes and the limit parameter T was 5 000 nodes just like in the previous two cases. The distance ω was set to 0.2 Å as well.

During the 100 runs of the RRT algorithm that were executed, the mode number of tunnels was 4, the extremes were minimal 0 and maximal 7 found tunnels. No tunnels were detected in only a single case. The overall comparison of results from the RRT implementation with results of CAVER 3.0 is shown in Table 5.3.



a) b) c)

Figure 5.5: Results from the tunnel detection in protein 1CQW. a) Cartoon representation of protein 1CQW. b) Transparent surface representation of protein 1CQW. Green tunnels were found by the proposed RRT algorithm. c) Orange tunnels, which were found by CAVER 3.0.

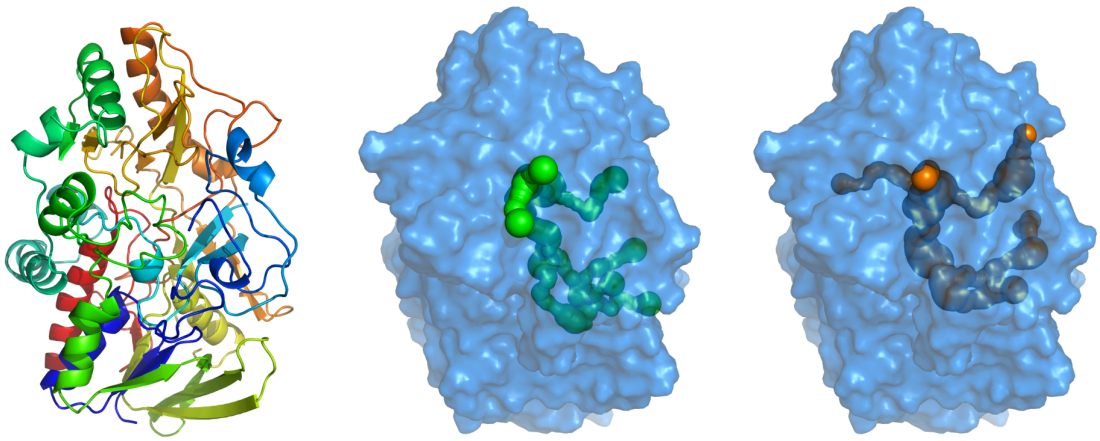
CAVER ID	Length [Å]	Success rate of RRT
1	17.2	96 %
2	22.6	97 %
3	23.1	95 %
4	26.2	95 %

Table 5.3: Comparison of results from the RRT implementation with results of CAVER 3.0 for protein 1CQW. The RRT algorithm was executed 100 times. Individual columns contain the tunnel index from CAVER 3.0 results, the length of this tunnel (in Ångströms) and the probability of finding the tunnel by the RRT implementation.

The finding probabilities of RRT are very high and results are comparable to results of CAVER. The structure of protein 1CQW does not contain many different paths and so, RRT algorithm does not have many possibilities, how to built the tree. Therefore, the RRT implementation in almost all runs detects very similar tunnels in this protein as CAVER 3.0.

5.1.4 1MXT

The results from tunnel detection in protein 1MXT are shown in Figure 5.6 together with its structure and also results from CAVER 3.0. In this particular run, RRT found six tunnels and CAVER detected also six tunnels. Five out of six of these tunnels were identified by comparison metric as identical. The limits for both surface reaching phase and endpoints detection phase were the same as in previous cases, S was 10 000 nodes and T was set to 5 000 nodes. The distance ω was again set to 0.2 Å.



a) b) c)

Figure 5.6: Results from the tunnel detection in protein 1MXT. a) Cartoon representation of protein 1MXT. b) Transparent surface representation of protein 1MXT. Green tunnels were found by the proposed RRT algorithm. c) Orange tunnels, which were found by CAVER 3.0.

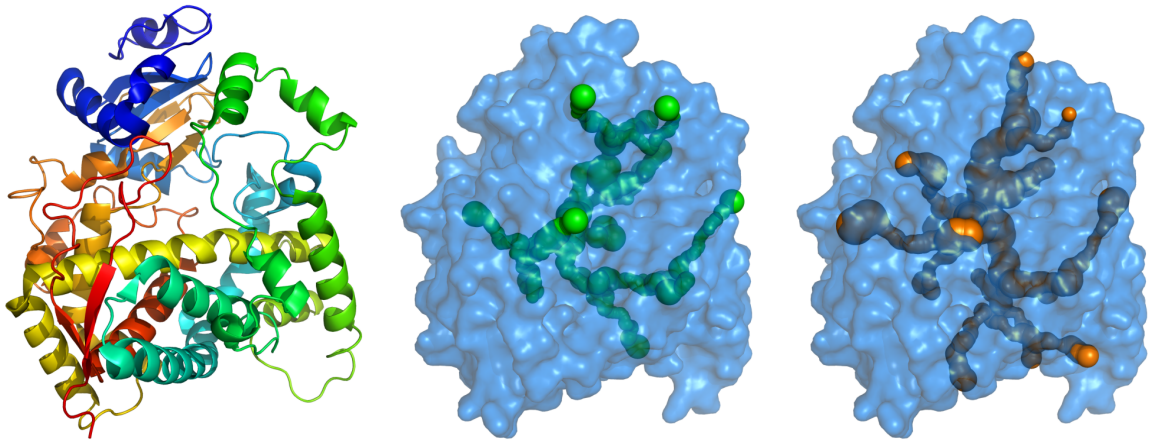
For the 100 runs of RRT algorithm, the mode number of tunnels was 5, the extremes were minimal 3 and maximal 12 found tunnels. The summary of comparison of the RRT algorithm results and results from CAVER is presented in Table 5.4 on the next page. Tunnels with ID 1 and 3 were found in every single run of RRT and also tunnel with ID 2 was detected in almost every run. Tunnel with ID 5 has noticeably low finding probability, since it was detected in only 36 trials. After a more thorough examination of the results, it turned out that this tunnel guides through very narrow place in protein structure and it also turns along its path. Tunnels with such properties are hard to find by the RRT implementation, when compared to wide and straight tunnels, which are find much more easily.

CAVER ID	Length [Å]	Success rate of RRT
1	20.3	100 %
2	21.3	93 %
3	34.6	100 %
4	30.2	79 %
5	21.6	36 %
6	49.6	68 %

Table 5.4: Comparison of results from the RRT implementation with results of CAVER 3.0 for protein 1MXT. The RRT algorithm was executed 100 times. Individual columns contain the tunnel index from CAVER 3.0 results, the length of this tunnel (in Ångströms) and the probability of finding the tunnel by the RRT implementation.

5.1.5 1TQN

Next, tunnels were detected in protein 1TQN. The results are shown in Figure 5.7, with protein structure and also results from CAVER 3.0. In the presented results, the RRT implementation found 15 protein tunnels and CAVER detected 17 tunnels. In this run, 10 out of 17 tunnels were identical based on the comparison metric. In the RRT algorithm, the limit parameter S was 20 000 nodes and the limit parameter T was 5 000 nodes.



a) b) c)

Figure 5.7: Results from the tunnel detection in protein 1TQN. a) Cartoon representation of protein 1TQN. b) Transparent surface representation of protein 1TQN. Green tunnels were found by the proposed RRT algorithm. c) Orange tunnels, which were found by CAVER 3.0.

The number of nodes for the surface reaching phase was increased, since with 10 000 nodes the number of detected tunnels was too low in some runs and protein structure

was not sufficiently explored. The distance ω was set to 0.2 Å.

During the 100 runs of RRT algorithm that were executed, the mode number of tunnels was 18, minimal 8 and maximal 28 tunnels were found. The overall comparison of results from the RRT implementation with results of CAVER 3.0 is shown in Table 5.5. Most of the tunnels have quite high finding probability, the exceptions are tunnels with ID 8 and 16, which are narrow and with several turns. When CAVER tunnels have such properties, they usually have smaller finding probabilities by the RRT algorithm, whereas wide and straight tunnels are ordinarily detected.

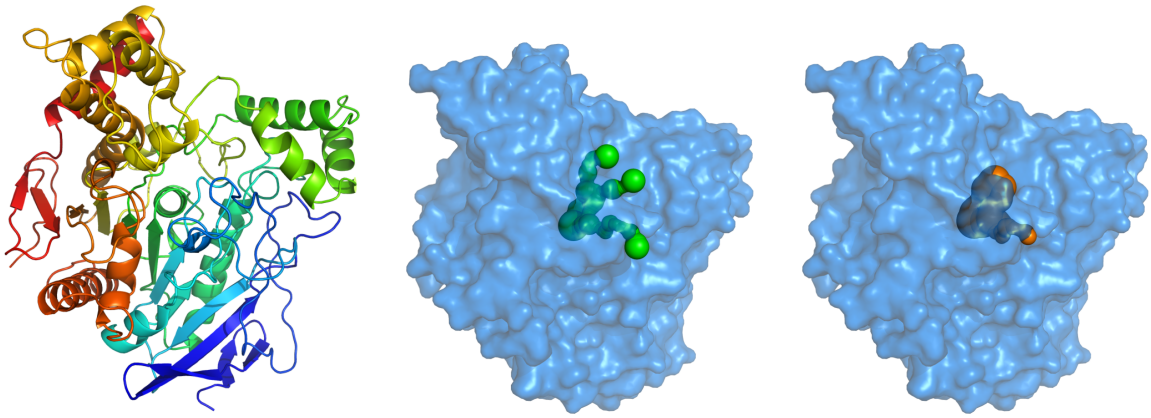
CAVER ID	Length [Å]	Success rate of RRT
1	25.5	98 %
2	20.0	84 %
3	17.6	78 %
4	35.3	78 %
5	26.4	85 %
6	18.0	84 %
7	40.4	41 %
8	30.5	0 %
9	39.3	87 %
10	26.3	94 %
11	30.3	66 %
12	35.7	84 %
13	28.6	70 %
14	45.2	69 %
15	38.7	31 %
16	46.9	7 %
17	46.8	22 %

Table 5.5: Comparison of results from the RRT implementation with results of CAVER 3.0 for protein 1TQN. The RRT algorithm was executed 100 times. Individual columns contain the tunnel index from CAVER 3.0 results, the length of this tunnel (in Ångströms) and the probability of finding the tunnel by the RRT implementation.

5.1.6 2ACE

The last static tunnel detection was performed with protein 2ACE. Figure 5.8 on the facing page shows results from tunnel detection in protein 2ACE, the structure

of this protein and also tunnels found by CAVER 3.0. In this particular run, the RRT implementation detected four protein tunnels and CAVER detected three tunnels. Based on the comparison metric, all three CAVER tunnels were found also by RRT. In the RRT algorithm, the limit parameter S was 8 000 nodes and the limit parameter T was 4 000 nodes. Both limits were decreased, because lower number of nodes were sufficient for the tunnel detection in this case, where just a small number of short tunnels were present. The distance ω was set to 0.2 Å.



a) b) c)

Figure 5.8: Results from the tunnel detection in protein 2ACE. a) Cartoon representation of protein 2ACE. b) Transparent surface representation of protein 2ACE. Green tunnels were found by the proposed RRT algorithm. c) Orange tunnels, which were found by CAVER 3.0.

During 100 runs of the RRT algorithm that were executed, the mode number of tunnels was 4, the extremes were minimal 2 and maximal 7 found tunnels. The overall comparison of results from the RRT implementation with results of CAVER 3.0 is shown in Table 5.6. All three tunnels were found in almost every single run and results of both methods from the tunnel detection in protein 2ACE were comparable.

CAVER ID	Length [Å]	Success rate of RRT
1	11.2	94 %
2	9.3	96 %
3	20.7	100 %

Table 5.6: Comparison of results from the RRT implementation with results of CAVER 3.0 for protein 2ACE. The RRT algorithm was executed 100 times. Individual columns contain the tunnel index from CAVER 3.0 results, the length of this tunnel (in Ångströms) and the probability of finding the tunnel by the RRT implementation.

5.2 Tunnel detection in dynamic protein structures

For the detection of tunnels in dynamic protein structures, we have analyzed the structure of haloalkane dehalogenase. Individual snapshots were acquired in a computer simulation, which takes a single frame with the protein structure and calculates the coordinates of protein atoms in selected number of frames. A hundred snapshots were used for the detection of dynamic tunnels and each of these frames was analyzed by ten runs of the RRT algorithm.

The results from tunnel detection in dynamic protein structures are shown in Figure 5.9 on the facing page. The software tool CAVER 3.0 is also designed to detect dynamic tunnels and the results from CAVER are presented together with the results of the RRT implementation. It can be observed that both methods have found similar tunnels and also similar count of tunnels. The RRT implementation detected the same tunnels as CAVER in 82 % of the runs and in the remaining 18 %, usually only one tunnel was not detected, which is a very good match of both methods.

Two phenomenons can be also noticed in the results. Firstly, the tunnels that were detected by CAVER seem to be wider. This can be caused by the fact that the RRT implementation randomly samples the sampling region and individual spheres, which represent the tunnel, are not exactly in the center of this tunnel. On the other hand CAVER aims to insert spheres, which represent the tunnel, in the same distance from protein atoms, thus being closer to the center of this tunnel. Secondly, the tunnel endpoints are again detected at different positions (same situation arose in the results of static tunnel detection in section 5.1). The tunnels detected by RRT are generally a little longer, whereas tunnels found by CAVER are terminated closer to the tunnel entrances of the protein surface.

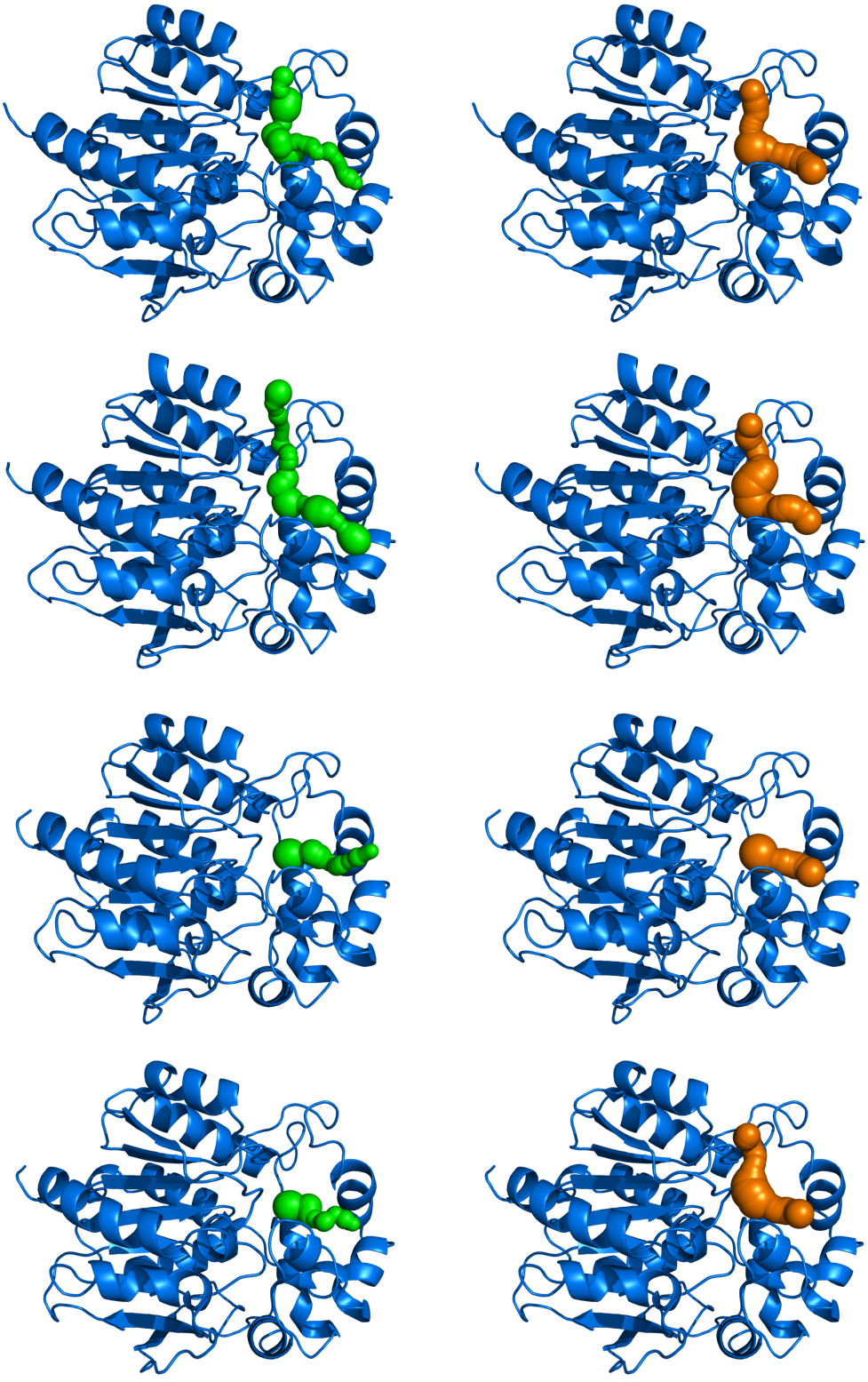


Figure 5.9: Results from the tunnel detection in the dynamics of haloalkane dehalogenase. The left side of this Figure contains four different snapshots of the protein and green tunnels, which were detected by the RRT implementation. The right side shows the same four snapshots of the protein and orange tunnels, which were found by CAVER 3.0. It can be observed that both methods found similar tunnels in separate snapshots. Only the bottom pair differs, where RRT detected only one tunnel, while CAVER found two tunnels.

5.3 Simulation of molecule passage through tunnels

After the RRT–based method for geometrical analysis of small molecule passage through protein tunnels was implemented, several ligands were tested for the traversability of a given tunnel. The experiments were performed with tunnels, which were found for the probe radius of 0.9 Å. It was usually not possible to detect tunnels with higher value of the radius.

At the beginning, all atoms from molecules were assigned with corresponding van der Waals radius, but since the probe radius for tunnel detection was quite low, the radii of atoms had to be decreased just in order to fit the molecule in collision–free initial configuration. Two different tunnels, detected by RRT in the structure of protein 1BL8, were used for computations. In the first instance, water molecule was analyzed for the traversability of given protein tunnel. The results are shown in Figure 5.10.

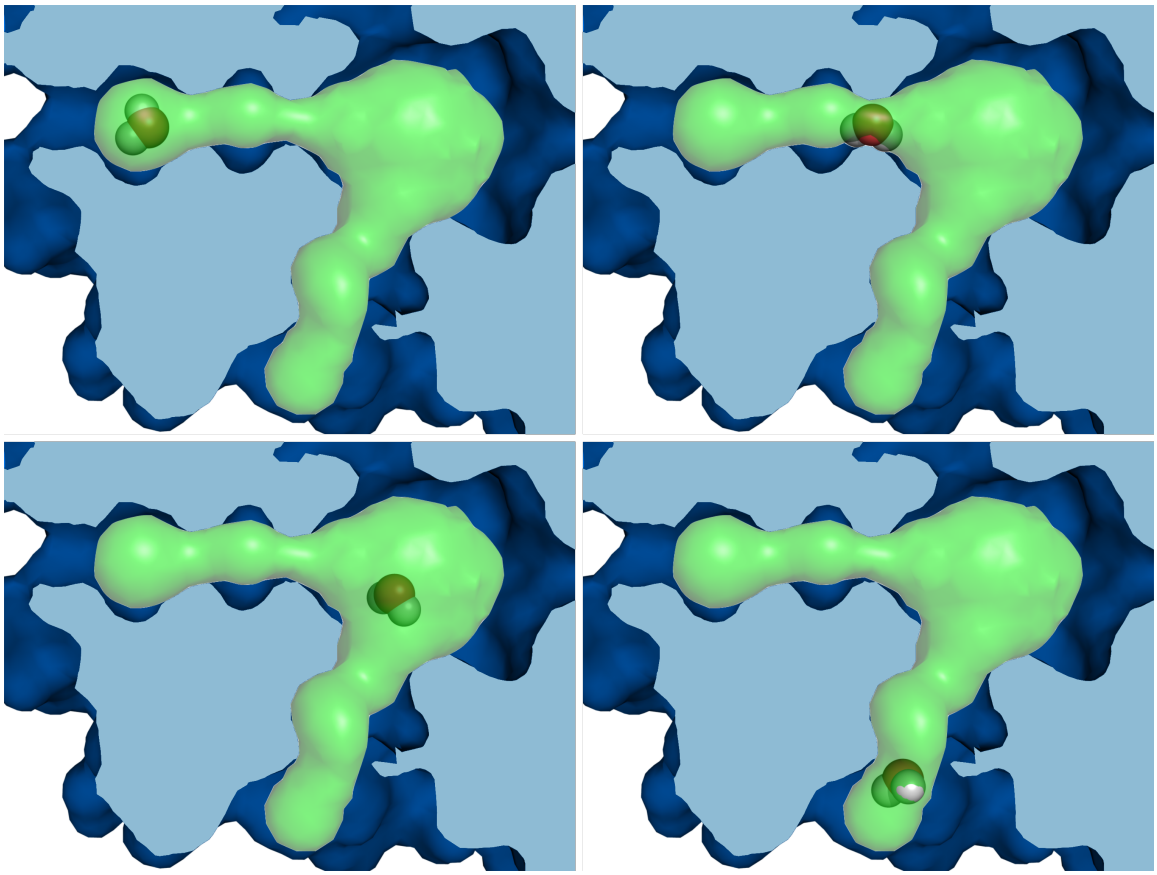


Figure 5.10: Geometrical analysis of water molecule passage through a protein tunnel (represented in green) found in pdb file 1BL8. Sliced protein is shown in blue. The left upper image depicts the initial configuration. The next three images show, how the water molecule progressively moves to the tunnel endpoint in the bottom right image.

Even though the water molecule is small, the van der Waals radii of individual

atoms had to be decreased in order to find a path from start to tunnel end. The van der Waals radii of oxygen and hydrogen are respectively 1.52 Å and 1.2 Å. For the purposes of RRT computations, these values were altered to 1.1 Å for oxygen and 0.9 Å for hydrogens. The distances between atom centers in water molecule were not changed.

The results of water molecule passage simulation show that at some places the tunnel narrows and the molecule needs to rotate into specific configuration to successfully pass to next section. On the other hand, some parts of the tunnel are quite wide, which allows the molecule to freely rotate without any obstacles. With the described alterations of atomic radii, the water molecule reached the tunnel endpoint.

The next geometrical analysis was performed with the ethanol molecule, which contains nine atoms and is considerably bigger than the molecule of water. The results of ethanol passage simulation are shown in Figure 5.11 on the next page. Another tunnel, which was also detected in protein 1BL8, was used for the analysis of the traversability in this case. This tunnel was significantly shorter then the tunnel tested with the water molecule and it did not have many narrow places. Individual atoms of ethanol molecule had to be scaled down even more than in the case of water molecule, otherwise the initial configuration could not be found. The radii of oxygen and hydrogens were respectively decreased to 1.0 Å and 0.7 Å. The carbon atoms were scaled down to 0.9 Å, while its van der Waals radius is 1.7 Å. The distances between atoms centers in the ethanol molecule were not changed.

Since there was not too much free space for rotation in this case, it can be observed that the ethanol molecule does not distinctly rotate. It rather turns just a little in order to successfully progress to the next parts of the tunnel. With the decreased radii of individual atoms, the ethanol molecule reaches the tunnel endpoint.

The results also show that some atoms of the molecule get out of the defined tunnel boundaries in certain parts of the simulation. The spheres, which represent the tunnel, are used to generate random coordinates for the molecule center. After these coordinates are obtained, it is checked, whether individual atoms from molecule have collision with protein atoms. Therefore, some atoms may overlap the boundaries of found tunnel, since it does not completely fill the space in the protein structure.

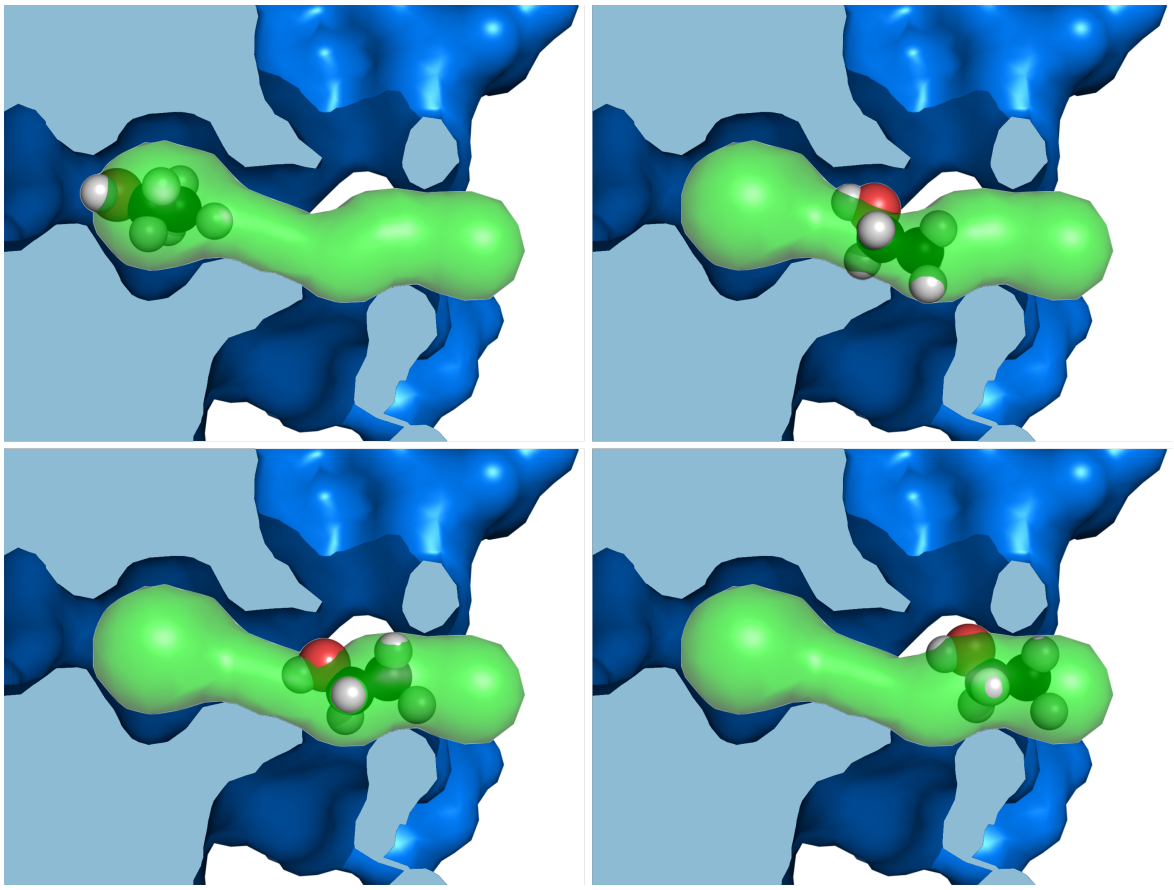


Figure 5.11: Geometrical analysis of passage of ethanol through a found protein tunnel (represented in green) found in pdb file 1BL8. Sliced protein is shown in blue. The left upper image depicts the initial configuration. The next three images show, how the molecule of ethanol progressively moves to the tunnel endpoint in the bottom right image.

5.4 Running times of RRT implementations

All the experiments were computed on computer with specifications shown in section 8.2. The running times for detection of tunnels in static protein structures are shown in Table 5.7 on the facing page. The average time of a single RRT run contains also the time, which was spent on comparison to tunnels that were found by CAVER. The comparison sometimes consumed even half of the running time, especially in cases, when a lot of tunnels were compared (proteins 1BL8 and 1TQN). The results show an expectable trend — with lower number of tunnels, the running time is shorter.

In the case of tunnel detection in dynamic protein structure, the overall running time was only 5 minutes, while 100 snapshots were analyzed with 10 runs per snapshot. So for 1000 runs of the RRT algorithm, the average time for a single run was less than

Protein pdb name	Number of CAVER tunnels	Mode count of RRT tunnels	Average time of a single RRT run
1AKD	5	5	15 s
1BL8	17	15	41 s
1CQW	4	4	7 s
1MXT	6	5	10 s
1TQN	17	18	50 s
2ACE	3	4	17 s

Table 5.7: The average running times for detection of tunnels in static protein structures.

1 second.

The geometrical analysis of molecule traversability was analyzed on two tunnels. If the tested molecule was able to pass through the tunnel, the computation took only few seconds for both tunnels.

Chapter 6

Discussion

With the increasing amount of data about structures of biomolecules like proteins and DNA, computations in the field of structural bioinformatics are a very popular research area. Detection of tunnels in static and dynamic protein structures is a widely studied problematics, for which several different approaches were utilized. One of the approaches uses sampling–based motion planning algorithms, which are well known from the robotic domain. This thesis focuses on the employment of a sampling–based algorithm, Rapidly–exploring Random Tree, which can be modified for the biological domain and the tunnel detection in both static and dynamic protein structures.

Apart from tunnel detection, this thesis also presents a novel approach to geometrical analysis of passage of small molecules through protein tunnels and this method is also based on Rapidly–exploring Random Tree. The results, which were obtained within this diploma thesis, are discussed in the following sections.

6.1 Tunnel detection in static protein structures

The RRT implementation, which was developed for the detection of static protein structures, was used for the analysis of six different protein molecules. The results from the static tunnel detection show that the RRT method was able to detect similar tunnels, which were found by CAVER 3.0, a state of the art software tool for protein tunnel detection. The RRT algorithm proved especially well in the detection of tunnels which are straight, wide and short. Tunnels with these properties were found even in 100 % of trials. On the other hand, tunnels that are rather serpentine, narrow and long were much harder to detect and in some cases, no tunnel with such properties was found. Nevertheless, the long and narrow tunnels might be biochemically unimportant,

because transportation of ligand through such tunnels is not very probable in real world and so, missing these tunnels does not have to be a problem.

The RRT method also had problems with the detection of the same tunnels as CAVER in sparse protein structures (e.g., pdb file 1AKD), which means that they have many possible paths in their structure that can be explored by the tree. Although RRT was able to detect similar count of tunnels as CAVER, there were cases in which none of the found tunnels was identical to CAVER tunnels, resulting in lower success of RRT.

The surface of individual proteins was represented by spheres, which were generated from set of surface points that were computed by computer program PyMOL. This approach seemed to work well in almost all situations, but the tunnel endpoints were often detected later than with CAVER and so, the found tunnels are longer. Even though it did not make a big difference in most of the cases, several additional tunnels were detected in some runs with protein 1BL8 in places, where CAVER had already terminated its search. Therefore, the determination of tunnel endpoints could be improved for better correspondence with CAVER.

The RRT algorithm was separated into two phases mainly because of the fact that we wanted to let the tree explore the protein structure as much as possible. But since the protein surface was represented by spheres with radii of 3 Å, which could reach a little into the protein structure, we did not determine tunnel endpoints immediately after the surface spheres were achieved, because these spheres could be theoretically reached also in paths, which are near to the surface, but not leading out of the protein. The tunnel endpoints were therefore detected after the tree extended out of protein structure, which results in longer tunnels.

Although the surface representation worked well in most cases, it failed in the situation, when the active site was very close to the surface of protein and initial configuration was therefore inside of the spheres that represent the surface (pdb file 1BL8). In such case, no tunnels were detected, until the initial configuration was manually moved to a different location. Active sites of enzymes are often located near its surface and so, similar situation need to be approached with caution. A possible solution to these cases could be representation of the surface by α–shape.

During the development of the RRT algorithm, we thought that the need to determine limits for total count of nodes for each phase will make the analysis of a new protein difficult. Nevertheless, both of these parameters (S and T) were the same for

most of the proteins (10 000 nodes for first phase and 5 000 nodes for second phase) and alterations were necessary just in the case of one smaller and one larger protein. Both values are therefore a good starting point for possible parameter optimization in the analysis of different proteins. Furthermore, because these two parameters need to be determined, it is controlled during the run of the algorithm, how many times in a row has the new node q_{new} collision with protein atoms in function *isTreeGrowing()*. If this number exceeds a given limit, the current phase is terminated. This should prevent the tree from exploring an already searched paths of the protein and it worked well in vast majority of the trials.

The proposed RRT method for the tunnel detection in static protein structures has better speed performance than another already employed RRT–based solution [9]. The improvement in speed was achieved by separation of our implementation into two phases and by better positioning of added plugs in the first phase, which prevented the tree from expanding into undesirable areas.

6.2 Tunnel detection in dynamic protein structures

The detection of tunnels in dynamic protein structure brought results, which were very similar to the results of CAVER 3.0. Ten runs of the RRT algorithm per snapshot and selection of frequent tunnels that are in at least 50 % of these runs proved to be a good approach to the analysis of dynamic protein structures.

It has to be mentioned that the proposed solution does not take into account the subsequent and preceding snapshots of individual structures and analyzes all frames separately. Although this approach might be acceptable, in order to detect a tunnel in the structure, it is necessary to find the path all the way from initial coordinates to the protein surface. For example, if the initial coordinates are not directly connected to the surface of protein, but there is a cavity that gradually changes its position towards the protein surface in subsequent snapshots, the proposed solution would miss this dynamic tunnel. Figure 6.1 on the next page shows this situation. The implemented solution can serve as a starting point for development of methods, which would take subsequent snapshots into consideration.

The proposed solution to tunnel detection in dynamic protein structures was very fast, the average running time per RRT run in this case was less than one second. The main reason is that the tunnels in analyzed protein structures were quite short and also

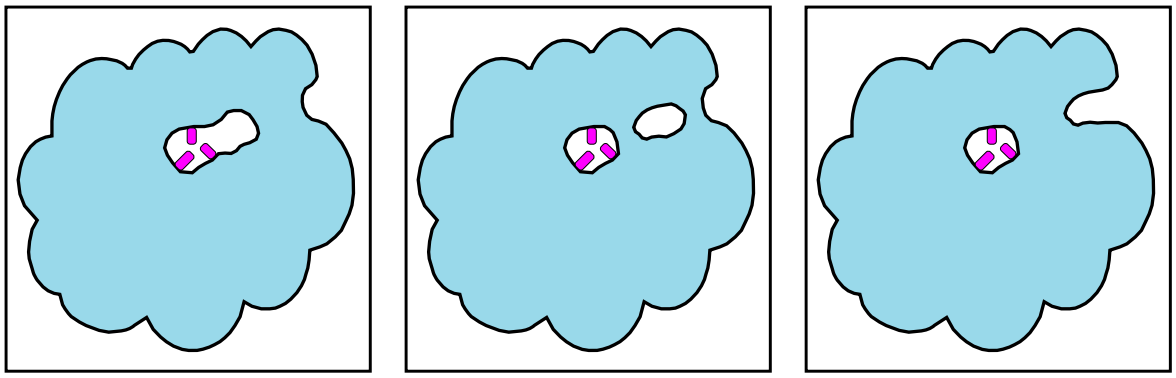


Figure 6.1: A specific situation in the detection of tunnels in dynamic protein structure. Protein is represented in blue, three magenta residues are at the initial configuration. If a cavity changes its position towards the protein surface in subsequent snapshots of the structure, the proposed RRT method would not detect it as a dynamic tunnel.

only two or three tunnels were usually detected in each run. Therefore, the analyses did not require so many tunnel comparisons as in the case of static tunnel detection, which significantly prolongs the running time.

6.3 Geometrical analysis of small molecules passage through protein tunnels

The RRT–based principle, which was designed and implemented for the geometrical analysis of small molecule passage through protein tunnels, successfully found paths for small molecules. Nevertheless, even the radii of water molecule atoms had to be decreased, otherwise the molecule could not even find the initial collision–free configuration. The proposed model, in which atoms are assigned with their van der Waals radii, is probably too strict, since carbon atoms alone were too big to fit the found tunnels.

Both of the presented results of molecule planning were performed on tunnels from protein 1BL8. It has to be mentioned that neither the water molecule, nor ethanol are substrates of this protein and these tunnels were just used to verify the implemented solution. When we tried our method on protein 1CQW (haloalkane dehalogenase) and a substrate (1,2–dibromoethane) of this enzyme, the radii of substrate atoms had to be decreased by more than 50 % of their van der Waals radii just to fit into initial configuration and we were not able to find a collision–free path to the tunnel endpoint.

The biggest downside of the hard sphere model is the inability to properly simulate

real ligands that are rather flexible. It should be emphasized that in reality both protein and ligand adapt their structures during the binding process. The proposed solution is simulating a transport of rigid small molecule between two locations in the static protein structure. Nevertheless, this strictly geometrical approach can serve as a starting point for advanced solutions. A great improvements might be the calculations with flexible molecules and also the addition of electrostatic interactions, which can describe attractive or repulsive forces between protein and ligand.

If the molecule was able to pass through the protein tunnel, the computation was very fast and the whole trial was completed within few seconds (computer specifications are presented in section 8.2). For the collision detection, only protein atoms, which were in close distance to the tested tunnel, were used. The fact that only a small part of protein atoms were employed further improved the speed of our implementation. The utilization of the guided RRT–Path approach [49] also significantly improves the performance of the algorithm, since only relevant samples are generated for the motion planning task.

A big advantage of our implementation of this geometrical analysis is the possibility to use tunnels, which were detected by different methods. The RRT–based solution uses the tunnel to generate random samples and for example tunnels detected by CAVER can be used for this purpose.

Chapter 7

Conclusions

The detection of protein tunnels and other computations related to protein tunnels are a very attractive field of structural bioinformatics. There are several different approaches to the task of tunnel detection and the vast majority of them are geometrical methods that usually neglect the physicochemical properties of individual molecules, which are represented by a hard sphere model. This thesis focused on utilization of a well known sampling–based algorithm, Rapidly–exploring Random Tree. We have proposed modification to this algorithm for various computations with protein tunnels.

In this diploma thesis, a novel RRT–based method for tunnel detection in static protein structures has been developed. It successfully detects tunnels in protein molecules and the results were in selected cases very similar to the results of CAVER 3.0, an established state of the art method for tunnel detection that employs Voronoi diagrams. The proposed RRT method had the best performance with detection of short and wide tunnels, while long and narrow tunnels were harder to detect.

The developed RRT implementation was then utilized for tunnel detection in dynamic protein structures, which were represented by a sequence of snapshots. Each snapshot was analyzed by several RRT trials and the most frequently found tunnels were selected as dynamic. This method was again comparable to CAVER 3.0 as it detected identical tunnels in more than 80 % of the snapshots.

This thesis also presents a novel RRT–based approach to the geometrical analysis of tunnel traversability of protein tunnels by small non–spherical probes. This model represents molecules as rigid, hard sphere objects and it uses a guided approach to solve the task of finding a path through a given tunnel. Although the developed principle successfully works, the radii of atoms in molecules, which were used in computations, needed to be decreased in order to fit the tunnel and find a path in the tunnel. The

ability to analyze the tunnel traversability by non–spherical probes is not yet available in most of the developed methods.

The proposed RRT methods have many future research possibilities. In the case of tunnel detection in dynamic protein structures, a solution that would take the subsequent snapshots into account can be designed. A very interesting improvements to the geometrical analysis of molecule path planning in protein tunnels might be the computation with flexible molecules (i.e., considering additional degrees of freedom) and also with electrostatic interactions between individual molecules.

Sometimes, the downside of the proposed solutions is the probabilistic nature of the RRT method, so the results vary with every trial and in rare cases, the RRT algorithm did not detect any tunnels, even though in other trials it detected several tunnels. The proposed RRT implementations might therefore be run more than once. Since the speed performance was quite fast, it is not problematic to have a few trials for each analysis.

To summarize all the completed guidelines, this thesis has presented a novel RRT–based approach to tunnel detection in static and also dynamic protein structures and provided comparison with CAVER 3.0, an established tunnel detection method. The results of the tunnel detection implementations were in some cases comparable to results of CAVER. A new method for geometrical analysis of tunnel traversability by rigid molecules was designed and implemented. The principle of geometrical analysis proved to successfully find path for rigid molecules in tunnels, nevertheless the van der Waals radii of individual atoms were to big to fit the tunnels and so, the radii of atoms needed to be scaled down. There are still possible improvements to the proposed methods and the models of molecules can be further developed for a more real world–like behavior.

Chapter 8

Appendices

8.1 Contents of the attached disc

The disc, attached to this thesis, contains:

- A pdf file with text of this thesis.
- Folder *programs*, which contains source codes of the RRT implementations for tunnel detection and geometrical analysis of molecule passage through protein tunnels. It also contains the utilized libraries (*MPNN* and *OZCollide*) and various supporting source codes for creation of surface spheres or computation of comparison metric.
- Folder *inputs*, with the files that were used as inputs in all proposed solutions
- Folder *results*, with images and PyMOL session files, which were used in the Results section

8.2 Computer specifications

All the experiments were run on the same computer laptop with the following specifications:

Acer Aspire 5820TG, Intel® Core™i5 CPU 430M, clocked at 2.26 GHz, 8 GB RAM

Bibliography

[1] Brocchieri, L.; Karlin, S. Protein length in eukaryotic and prokaryotic proteomes. *Nucleic Acids Research*. **2005**, *33*, 3390–3400.

[2] *Protein Data Bank*. `http://www.rcsb.org/pdb/explore.do?structureId=1AOI`; cited: April 14th 2017.

[3] Yee, A. A.; Savchenko, A.; Ignachenko, A.; Lukin, J.; Xu, X.; Skarina, T.; Evdokimova, E.; Liu, C. S.; Semesi, A.; Guido, V.; Edwards, A. M.; Arrowsmith, C. H. NMR and X-ray crystallography, complementary tools in structural proteomics of small proteins. *Journal of the American Chemical Society*. **2005**, *127*, 16512–16517.

[4] Park, E.; Campbell, E. B.; MacKinnon, R. Structure of a CLC chloride ion channel by cryo-electron microscopy. *Nature*. **2017**, *541*, 500–505.

[5] *RCSB Protein Data Bank - RCSB PDB*. `http://www.rcsb.org/pdb/home/home.do`; cited: April 16th 2017.

[6] Kingsley, L. J.; Lill, M. A. Substrate tunnels in enzymes: Structure-function relationships and computational methodology: Protein Tunnel Structure-Function Relationship. *Proteins: Structure, Function, and Bioinformatics*. **2015**, *83*, 599–611.

[7] Elbanhawi, M.; Simic, M. Sampling-Based Robot Motion Planning: A Review. *IEEE Access*. **2014**, *2*, 56–77.

[8] Park, F. C.; Lynch, K. M. Modern robotics, mechanics, planning and control. *Cambridge University Press*. **2017**.

[9] Jankovec, T. Bachelor thesis: Path planning in protein structures. *Czech Technical University in Prague, Faculty of Electrical Engineering*. **2016**.

[10] Vonásek, V.; Kozlíková, B. Application of sampling-based path planning for tunnel detection in dynamic protein structures. *Methods and Models in Automation and Robotics (MMAR), 2016 21st International Conference*. **2016**, 1010–1015.

[11] Maton, A.; Hopkins, J.; McLaughlin, C. W.; Johnson, S.; Warner, M. Q.; LaHart, D.; Wright, J. D. Human biology and health. *Prentice Hall*. **1997**.

[12] Nelson, D. L.; Cox, M. M. Lehninger Principles of Biochemistry, fifth edition. *W.H. Freeman and company*. **2008**.

[13] Gutteridge, A.; Thornton, J. M. Understanding nature's catalytic toolkit. *Trends in Biochemical Sciences*. **2005**, *30*, 622–629.

[14] Lodish, H.; Berk, A.; Matsudaira, P.; Kaiser, C. A.; Krieger, M.; Scott, M. P.; Zipursky, L.; Darnell, J. Molecular Cell Biology, fifth edition. *W.H. Freeman and company*. **2003**.

[15] Stryer, L.; Berg, J. M.; Tymoczko, J. L. Biochemistry. *W.H. Freeman and company*. **2002**.

[16] Fischer, E. Einfluss der Configuration auf die Wirkung der Enzyme. *Berichte der deutschen chemischen Gesellschaft*. **1894**, *27*, 2985–2993.

[17] Koshland, D. E. Application of a Theory of Enzyme Specificity to Protein Synthesis. *Proceedings of the National Academy of Sciences of the United States of America*. **1958**, *44*, 98–104.

[18] Fraser, J. S.; Clarkson, M. W.; Degnan, S. C.; Erion, R.; Kern, D.; Alber, T. Hidden alternative structures of proline isomerase essential for catalysis. *Nature*. **2009**, *462*, 669–673.

[19] Damborský, J.; Petřek, M.; Banáš, P.; Otyepka, M. Identification of tunnels in proteins, nucleic acids, inorganic materials and molecular ensembles. *Biotechnology Journal*. **2007**, *2*, 62–67.

[20] Brezovský, J.; Chovancová, E.; Gora, A.; Pavelka, A.; Biedermannová, L.; Damborský, J. Software tools for identification, visualization and analysis of protein tunnels and channels. *Biotechnology Advances*. **2013**, *31*, 38–49.

[21] Schwede, T.; Peitsch, M. C. Computational structural biology: Methods and applications. *World scientific*. **2008**.

[22] Petřek, M.; Otyepka, M.; Banáš, P.; Košinová, P.; Koča, J.; Damborský, J. CAVER: a new tool to explore routes from protein clefts, pockets and cavities. *BMC Bioinformatics*. **2006**, *7*, 316.

[23] Levitt, D. G.; Banaszak, L. J. POCKET: a computer graphics method for identifying and displaying protein cavities and their surrounding amino acids. *Journal of Molecular Graphics*. **1992**, *10*, 229–234.

[24] Ho, B. K.; Gruswitz, F. HOLLOW: generating accurate representations of channel and interior surfaces in molecular structures. *BMC structural biology*. **2008**, *8*, 49.

[25] Voss, N. R.; Gerstein, M. 3V: cavity, channel and cleft volume calculator and extractor. *Nucleic Acids Research*. **2010**, *38*, W555.

[26] *Voronoi diagrams*. `http://d.hatena.ne.jp/kaiseh/20091010/1255198573`; cited: April 29th 2017.

[27] Chovancová, E.; Pavelka, A.; Beneš, P.; Strnad, O.; Brezovský, J.; Kozlíková, B.; Gora, A.; Šustr, V.; Klvaňa, M.; Medek, P.; Biedermannová, L.; Sochor, J.; Damborský, J. CAVER 3.0: a tool for the analysis of transport pathways in dynamic protein structures. *PLoS computational biology*. **2012**, *8(10)*.

[28] Petřek, M.; Kosinová, P.; Koča, J.; Otyepka, M. MOLE: a Voronoi diagram-based explorer of molecular channels, pores, and tunnels. *Structure (London, England: 1993)*. **2007**, *15*, 1357–1363.

[29] Yaffe, E.; Fishelovitch, D.; Wolfson, H. J.; Halperin, D.; Nussinov, R. MolAxis: efficient and accurate identification of channels in macromolecules. *Proteins*. **2008**, *73*, 72–86.

[30] Lindow, N.; Baum, D.; Hege, H. C. Voronoi-Based Extraction and Visualization of Molecular Paths. *IEEE Transactions on Visualization and Computer Graphics*. **2011**, *17*, 2025–2034.

[31] LaValle, S. M. Rapidly-exploring random trees: A new tool for path planning. *Technical Report, Computer Science Department, Iowa State University*. **1998**.

[32] Cortés, J.; Siméon, T.; Ruiz de Angulo, V.; Guieysse, D.; Remaud-Siméon, M.; Tran, V. A path planning approach for computing large-amplitude motions of flexible molecules. *Bioinformatics (Oxford, England)*. **2005**, *21 Suppl 1*, i116–125.

[33] Haranczyk, M.; Sethian, J. A. Navigating molecular worms inside chemical labyrinths. *Proceedings of the National Academy of Sciences*. **2009**, *106*, 21472–21477.

[34] Kozlíková, B.; Krone, M.; Falk, M.; Lindow, N.; Baaden, M.; Baum, D.; Viola, I.; Parulek, J.; Hege, H.-C. Visualization of Biomolecular Structures: State of the Art Revisited. *Computer Graphics Forum*. **2016**, n/a–n/a.

[35] Coleman, R. G.; Sharp, K. A. Travel depth, a new shape descriptor for macromolecules: application to ligand binding. *Journal of Molecular Biology*. **2006**, *362*, 441–458.

[36] Cavasotto, C. N.; Orry, A. J.; Abagyan, R. A. The challenge of considering receptor flexibility in ligand docking and virtual screening. *Current Computer-Aided Drug Design*. **2005**, *1*, 423–440.

[37] Forli, S.; Huey, R.; Pique, M. E.; Sanner, M.; Goodsell, D. S.; Olson, A. J. Computational protein-ligand docking and virtual drug screening with the AutoDock suite. *Nature protocols*. **2016**, *11*, 905–919.

[38] Trott, O.; Olson, A. J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading. *Journal of computational chemistry*. **2010**, *31*, 455–461.

[39] Ruiz-Carmona, S.; Alvarez-Garcia, D.; Foloppe, N.; Garmendia-Doval, A. B.; Juhos, S.; Schmidtke, P.; Barril, X.; Hubbard, R. E.; Morley, S. D. rDock: A Fast, Versatile and Open Source Program for Docking Ligands to Proteins and Nucleic Acids. *PLOS Computational Biology*. **2014**, *10*, e1003571.

[40] Grosdidier, A.; Zoete, V.; Michielin, O. EADock: Docking of small molecules into protein active sites with a multiobjective evolutionary optimization. *Proteins: Structure, Function, and Bioinformatics*. **2007**, *67*, 1010–1025.

[41] *AutoDock Vina - molecular docking and virtual screening program*. `http://vina.scripps.edu/`; cited: May 2nd 2017.

[42] Furmanová, K.; Jarešová, M.; Byška, J.; Jurčík, A.; Parulek, J.; Hauser, H.; Kozlíková, B. Interactive exploration of ligand transportation through protein tunnels. *BMC Bioinformatics*. **2017**, *18*, 22.

[43] Devaurs, D.; Bouard, L.; Vaisset, M.; Zanon, C.; Al-Bluwi, I.; Iehl, R.; Simeon, T.; Cortes, J. MoMA-LigPath: a web server to simulate protein-ligand unbinding. *Nucleic Acids Research*. **2013**, *41*, W297–W302.

[44] Cortés, J.; Le, D. T.; Iehl, R.; Siméon, T. Simulating ligand-induced conformational changes in proteins using a mechanical disassembly method. *Physical Chemistry Chemical Physics*. **2010**, *12*, 8268–8276.

[45] Jaillet, L.; Corcho, F. J.; Pérez, J.-J.; Cortés, J. Randomized tree construction algorithm to explore energy landscapes. *Journal of Computational Chemistry*. **2011**, *32*, 3464–3474.

[46] Yershova, A.; LaValle, S. M. Improving Motion-Planning Algorithms by Efficient Nearest-Neighbor Searching. *IEEE Transactions on Robotics*. **2007**, *23*, 151–157.

[47] *Igor Kravtchenko. Ozcollide: Advanced, fast and free collision detection library impelemented in C++*. `www.tsarevitch.org/ozcollide/`; cited, but not accesible: May 11th 2017, originally downloaded: November 2015.

[48] Beneš, P.; Medek, P.; Sochor, J. Computation of channels in protein dynamics. *Proceedings of the IADIS International Conference Applied Computing, IADIS Press Rome*. **2009**, 251–258.

[49] Vonásek, V.; Faigl, J.; Krajník, T.; Přeučil, L. RRT-path – A Guided Rapidly Exploring Random Tree. *SpringerLink*. **2009**, 307–316.