Computation of Dynamic Channels in Proteins

Petr Beneš, Petr Medek, Ondřej Strnad, Jiří Sochor
Faculty of Informatics
Masaryk University
Brno, Czech republic
xbenes2@fi.muni.cz, peme@peme.cz, xstrnad2@fi.muni.cz, sochor@fi.muni.cz

***Abstract*—In this paper, we propose a new method which considers the movement of a protein molecule as a whole for the computation of so called dynamic channels in a molecular dynamics trajectory. The method is based on maximizing the information about the empty space over time and is built on basic computational geometry principles. The dynamic channels highlight pulsing and flexible parts of the molecule. It is believed that such parts allow a ligand to pass into or out from the active site. The method was tested on real protein data and the results indicate that it presents new information about the molecule.**

***Keywords*-protein; dynamic channel; molecule; trajectory; computational geometry**

I. Introduction

The shape of a protein molecule is complicated and contains many cavities and pockets. In our research, we are primarily interested in specific cavities connecting a part of a protein molecule (active site) with the surface of the protein. Such cavities are denoted as channels. Channels are used by a substrate molecule to pass into the active site where it can react and may also be used by the products to leave the active site.

The structure of a protein molecule does not remain static over time. Atoms are continuously moving. With this movement, cavities and channels are also changing. The movement of atoms (protein dynamics) is represented as a set of states of the protein molecule which we call snapshots. The whole set is called a trajectory and may contain thousands of snapshots.

Recent methods for computation of channels typically detect channels separately in each snapshot. The channels computed in one snapshot are optimized for bottleneck radius in this particular snapshot only, but not in the whole trajectory. Over time, as atoms are moving, the channel may pulse and thus its parts may alternate from really narrow to wide. In each snapshot then, only a part of a channel may be wide while other parts of the same channel are narrow. Evaluating snapshots separately implies that such a channel would not be identified by existing methods. During a given time interval the channel may be wide in different parts. If these parts are considered altogether, we can find that the channel was wide along its whole length and is maximized in width for the whole trajectory (see Fig. 1).

Such a channel which is detectable in multiple snapshots is called dynamic channel. The method proposed in this paper is designed to detect dynamic channels. This approach considers snapshots together which ensures that dynamic channels are optimized for the whole trajectory. In other words, a dynamic channel may be composed of parts from different snapshots.

The dynamic channel is an approximation because it does not take the order of snapshots into account. In spite of this, such information is valuable since the trajectory is the approximation of the reality as well and since it covers only a short interval of protein life.

Each part of the dynamic channel is wide in a certain snapshot and thus it is expected that the protein molecule is flexible in that parts. This means that if the substrate molecule would pass through a dynamic channel, the atoms in the protein molecule may easily move and create the necessary empty space.

Our method assumes that the whole protein molecule does not change its position significantly. The data obtained from molecular dynamics simulations usually satisfy this condition. If not, there are various alignment techniques which are able to omit the global movement of the molecule.

Preliminary testing on haloalkane dehalogenase DhaA indicates that the method provides reasonable results. However, this paper does not address the issue of biochemical relevance of computed channels – it presents the method and its capabilities.

II. Related work

A channel in a protein molecule is defined [1] as a centerline and a volume. The centerline is a three-dimensional continuous curve and the volume is formed by the union of spheres with centers on the centerline and with an appropriate radius so that they do not intersect any atom in the molecule. The example of a channel is demonstrated in Fig. 1.

There are many methods which deal with the issue of detecting cavities in protein molecules. For instance, the method introduced in [2] is based on the alpha shape theory. The latest approaches can be found for instance in [3], [4]. The information about cavities is important, but these methods do not consider the cavities as channels.

 ISBN: 978-1-61208-137-3

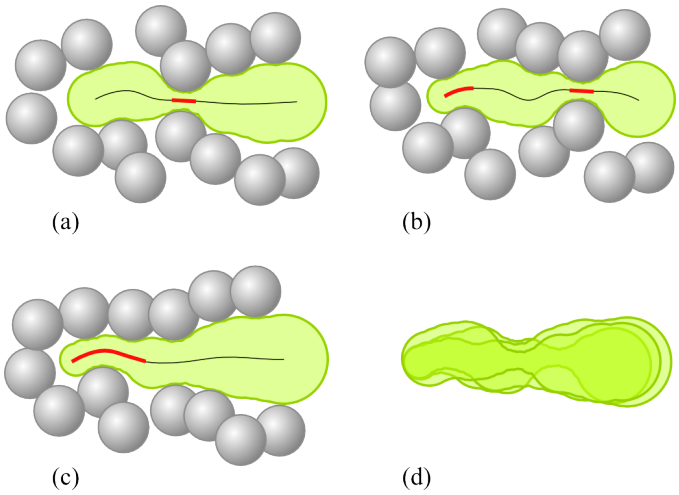

Figure 1. Dynamic channel. (a), (b), (c) A part of a channel is very narrow in each snapshot (emphasized). (d) All channels considered at once form a dynamic channel which is maximized in width across all three snapshots.

There are various approaches to channel computation. Most of them deal with the static case only. The first method which was capable of detecting channels was introduced in [5]. The method was based on sampling the molecule using a three-dimensional grid and suffered from several disadvantages. The methods that followed were based on computational geometry – the first of them was proposed in [1]. Other methods [6], [7] released later are similar. In all these methods, the space partitioning is represented by the Voronoi diagram (VD) or its dual Delaunay triangulation (DT). These structures are processed by an algorithm for finding the shortest path in a graph (typically Dijkstra's algorithm) and a channel which is optimal for given criteria is found. In principle, the centerlines of channels computed using these methods lead along Voronoi edges. A grid-based method introduced in [8] is able to find static molecular channels and voids.

The movement of atoms in a protein molecule is usually obtained by complex physical simulations called molecular dynamics [9]. In reality, the movement is continuous, but the simulation provides discrete results in the form of a set of snapshots as referred above. A variant of molecular dynamics, Random accelerated molecular dynamics (RAMD, [10]) could be also used to detect channels. In RAMD, a small molecule is placed into the active site and a simulation is started. During the simulation, additional random forces are applied to the small molecule and it is probable that it reaches the surface and leaves the protein molecule. If the small molecule leaves, an escape path exists and is detected. The disadvantage of this approach is that it is time consuming and it does not ensure that an escape path will be found.

Other method introduced in [11] computes channels in an existing trajectory. A given number of widest channels is computed in each snapshot and they are partitioned into clusters according to their similarity after all snapshots are processed. Each cluster contains channels from different snapshots which are similar and thus these channels can be considered as states of one channel over time. The similarity of channels is determined by a user-defined distance function. This approach has an obvious disadvantage. The method is not able to detect dynamic channels since it is focused on the computation of channels with the biggest bottleneck radius in each snapshot. The problem is that if we compute a limited number of channels in each snapshot, the dynamic channel may not be present among these channels, because in each snapshot it may not be wide along its whole centerline (see Fig. 1).

Recent methods for computation of channels are based on computational geometry principles. The fundamental geometric structures Voronoi diagram and Delaunay triangulation are of key importance for the proposed method. The duality which exists between these structures allows the easy converting between them. In three dimensions, the important correspondences which are used for our purposes are such that a tetrahedron in the Delaunay triangulation corresponds to Voronoi vertex, a triangle face shared by two neighbouring tetrahedra in DT corresponds to Voronoi edge. Certainly, there are other correspondences, but they are not necessary for the purposes of this paper and are omitted. For more details we refer to [12]. The illustration of the duality in three dimensions can be seen in Fig. 2.

In the dynamic case, when processing the whole trajectory, it is necessary to compute the Voronoi diagram in each snapshot. The issue of computation of the Voronoi diagram in the dynamic environment is described in [13]. Instead of recomputing the Voronoi diagram, the algorithm only performs necessary updates. The complexity of this algorithm is dependent on the number of changes in the Voronoi diagram. This approach assumes that input trajectories of moving spheres are continuous and thus its use in the method presented in this paper is limited. Therefore we recompute the Voronoi diagram for each snapshot using the QuickHull algorithm [14].

III. Proposed method

A dynamic channel can be as well as static channel ([1]) defined as a centerline and a volume which is formed by the union of empty spheres inserted in each point on the centerline. In addition to previous definition, the dynamic channel contains the information about snapshot number for each point on the centerline – a sphere inserted at that point does not intersect nor contain any of the atoms in that particular snapshot.

Definition: Let $M = \{m_1, ..., m_k\}$ be a set of snapshots in the trajectory. A dynamic channel T is defined as $T = \bigcup_{x \in a_T} s(x, r, i)$ where a_T is a three-dimensional curve (centerline of T) and $s(x, r, i)$ is a sphere with center x

and radius r which does not intersect any atom in snapshot $m_i \in M$.

The method we propose is based on the two algorithms described in [1] and [15]. Therefore we first briefly describe the main idea of these algorithms. Then, we describe a novel method which is the main contribution of this paper.

The space partitioning of the protein molecule is stored in the Delaunay triangulation (DT) computed for its atoms. In [1], the DT computed for the set of atom centers is converted to edge-weighted graph G. The nodes of G are formed by Voronoi vertices dual to tetrahedra in the DT. In G, an edge between two nodes exists, if the corresponding tetrahedra in DT share a face (Fig. 2). The value of the edge is equal to the distance from the edge to the surface of the nearest atom. Graph G is then processed using the Dijkstra's algorithm. The cost function maximizes the bottleneck radius of the channel.

The algorithm introduced in [15] is designed to track the centerline of an existing channel in any snapshot in a trajectory. In each snapshot, the algorithm locates the set of tetrahedra intersected by the centerline of the channel and determines Voronoi edges dual to faces shared by neighbouring tetrahedra in this set. As a result of the algorithm, a new centerline composed of Voronoi edges is returned. The centerline is optimized in width while preserving its location. The method uses so-called walks in the Delaunay triangulation to speed up the tracking progress.

We propose a new method for the detection of dynamic channels defined above. The method can be divided into three main parts. Firstly, a graph G_{ini} which represents the molecule appropriately is created. The topology of the graph G_{ini} remains constant during the computation. Secondly, all snapshots in the trajectory are processed. In each snapshot, edges in G_{ini} are updated so that after the processing of the whole trajectory, the value of each edge is maximized. The third step is to compute paths in G_{ini} from the starting node to any of the boundary nodes. The thorough description of these steps follows.

A. G_{ini} creation

There are many possibilities while creating G_{ini}, but not all of them, however, represent the molecule conveniently. In this paper, we propose following two G_{ini} variants which we expect to provide accurate results. We either use the Voronoi diagram (VD) from a selected snapshot of the trajectory or create G_{ini} based on a 3D grid. In case of grid, the bounding volume of the molecule is sampled uniformly using given sampling density. Samples in the grid are used as nodes in G_{ini}. Two nodes are connected by an edge if the corresponding samples are neighbours in the 3D grid. Both G_{ini} variants are discussed in Results section.

The node in G_{ini} that is the nearest to the active site is marked as starting node. The active site is specified by a user as three-dimensional coordinates or by surrounding

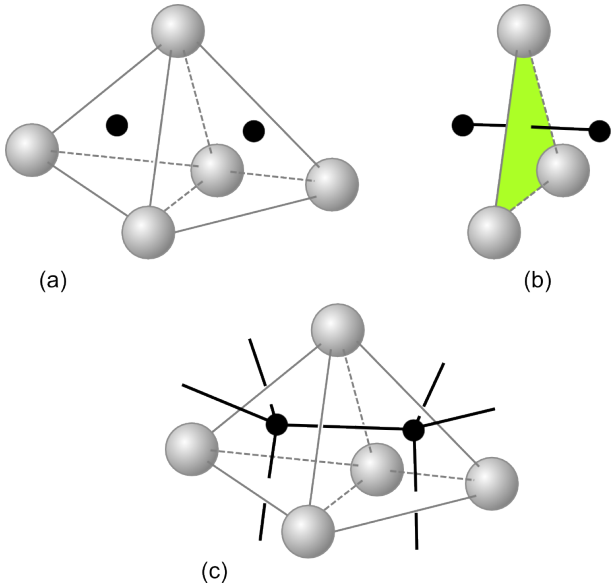

Figure 2. Voronoi – Delaunay duality in three dimensions. (a) Tetrahedra dual to Voronoi vertices. (b) A triangle face shared by two neighbouring tetrahedra is dual to a Voronoi edge. (c) Complete Voronoi – Delaunay duality.

atoms in the molecule. If atoms are used, the coordinates are typically computed by averaging the atom positions in the first snapshot in the trajectory.

For further computation of channels it is necessary to mark certain nodes in G_{ini} as boundary. These nodes lay near the surface of the molecule. In both G_{ini} variants, nodes of edges which are outside or intersect the convex boundary of the molecule are marked as boundary.

B. G_{ini} maximization

The maximization process of values of edges works as follows. All edges in G_{ini} are processed in each snapshot. For each snapshot m_i, each particular edge e in G_{ini} is tracked using the previously mentioned procedure for tracking a channel [15]. Recall that the procedure returns the set of Voronoi edges $e_{track} = \{e_{t_1}, ..., e_{t_n}\}$ which are dual to the tetrahedra intersected by e. The bottleneck of edges in e_{track} is determined and compared against the value of e. If the bottleneck is larger, then the value of e is updated. Additionally, the actual snapshot m_i for this edge as well as its optimized geometry e_{track} is stored for further reconstruction. The process of updating one edge is depicted in Fig. 3.

After processing of all snapshots, the value of each edge in G_{ini} is maximal in the whole trajectory.

It is clear that it would be time consuming to apply the tracking to each single edge e in G_{ini} in each snapshot. Since the edges e_{track} returned as a result of tracking procedure for e are dual to tetrahedra in DT, the last tetrahedron intersected (i.e., the tetrahedron containing end node of e)

 ISBN: 978-1-61208-137-3

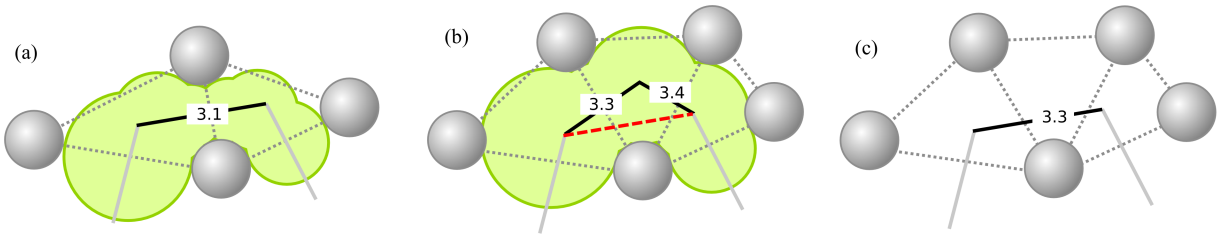

Figure 3. Edge update. (a) Edge in G_{ini} with so far maximal value. (b) Tracked edges e_{track}; bottleneck value: 3.3Å. (c) Edge value is updated.

can be stored with the node in G_{ini} and reused when edges emanating from this node are processed. Together with the breadth-first search (BFS) processing of edges from the starting node it is ensured that when an edge is processed, the tetrahedron for the starting node of the edge is known and the expensive tetrahedron location test has to be performed for the starting node only.

C. G_{ini} Dijkstra's algorithm processing

Finally, updated G_{ini} is processed by the Dijkstra's algorithm in the same way as in the static case [1]. Typically, the path in G_{ini} with the biggest bottleneck which connects the starting node (located in the active site) with arbitrary boundary node is computed. Such path defines the centerline of a dynamic channel. Naturally, for each edge in the centerline of the resulting channel, the information about its geometry as well as the number of snapshot in which the edge is valid is known.

The Dijkstra's algorithm cost function is modified in the way such that channels with the biggest possible bottleneck are computed. In addition, it is ensured that if a boundary node is selected during the algorithm progress, the path with the biggest bottleneck connecting the starting node with the boundary is found. At this point, the computation is terminated and the resulting channel is reported. Alternatively, certain edges in G_{ini} can be disabled and the Dijkstra's algorithm may be run again to find another different dynamic channel. Such approach is widely used when computing channels in static molecules, see [16].

D. Complexity

The number of nodes in G_{ini} is equal to the number of Voronoi edges from any snapshot in case the VD is used. If the grid solution is used, the number of edges is appropriate to the sampling density. Let i be the number of edges in G_{ini}. The theoretical maximum number of tetrahedra processed to track the i edges in G_{ini} in one snapshot is $\mathcal{O}(i \cdot n^2)$. However, for the analysed real data, the expected complexity is significantly better. Each edge in G_{ini} intersects only a limited number of tetrahedra in the DT computed for each snapshot. With the BFS applied, the expected time to track all i edges in G_{ini} is linear with respect to the number of tetrahedra in the DT in each snapshot. After processing all snapshots, the Dijkstra's algorithm is run. Its complexity is $\mathcal{O}(i^2)$ in the worst case. Again, the expected time is smaller since the algorithm can terminate before processing all edges in G_{ini} if a path which ends in a boundary node is computed.

E. Dynamic channels and channel states

Once a dynamic channel is computed, its actual geometry can be visualized in the corresponding valid snapshots. This means that a user would view the snapshots in a trajectory and the corresponding parts of a channel would be visualized in their respective snapshots.

In addition, the behaviour of the empty space near the whole centerline of the dynamic channel can be computed using the method in [15]. In each snapshot, the centerline is tracked and the resulting geometry can be visualized. In this manner, the user can get a state of the dynamic channel in each snapshot and can get a complex view on the behaviour of the molecule near the dynamic channel. Alternatively, the method proposed in [18] could be used. The method can effectively update the triangulation near the centerline of the dynamic channel and use it for the determination of states of dynamic channel in each snapshot.

IV. Results

To show that spatial changes appear in the molecule, we analyzed the width of all edges in G_{ini} for each snapshot. As the input data, trajectories of haloalkane dehalogenase DhaA were used (wild-type wt, mutated with codenames 04, 14, 15; more details on the data can be found in [17]). The number of updates and the average edge value were determined. As mentioned in the previous section, the update happens only when the value of an edge in the current snapshot is larger than the value of the corresponding edge in G_{ini}. As shown in Fig. 4 (a) the average edge value has increased after processing each snapshot. Moreover, the increase is relatively high which indicates that there are significant changes in the behaviour of empty space inside the protein. In addition, the results indicate that the value of edges is increasing despite the fact that the number of edge updates is relatively low and decreases (see Fig. 4 (a, b)). After processing a certain number of snapshots we can compute the channel in G_{ini} with the largest bottleneck – the

 ISBN: 978-1-61208-137-3

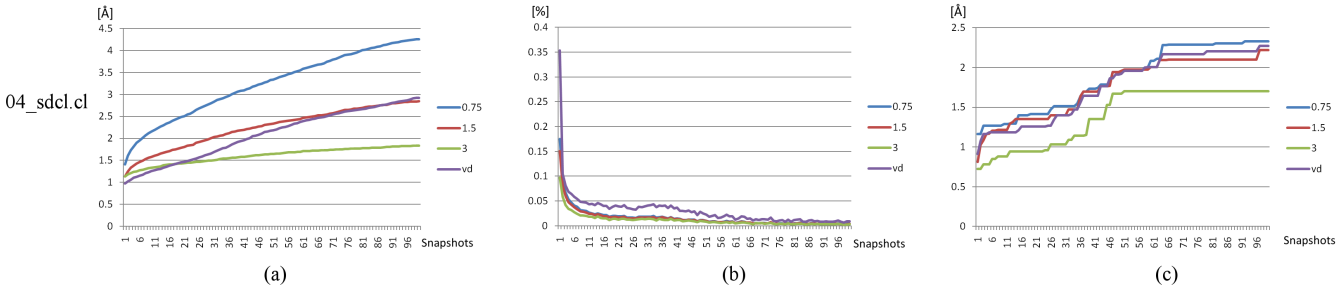

Figure 4. The analysis of 04_rdcl.cl and 04_sdcl.cl (details in [17]). (a) The average width of edge in G_{ini} after processing certain number of snapshots. (b) The percentage of edges updated after adding each snapshot. (c) The bottleneck radius of the widest dynamic channel computed after adding i-th snapshot. Similar trends were observed for other analysed trajectories.

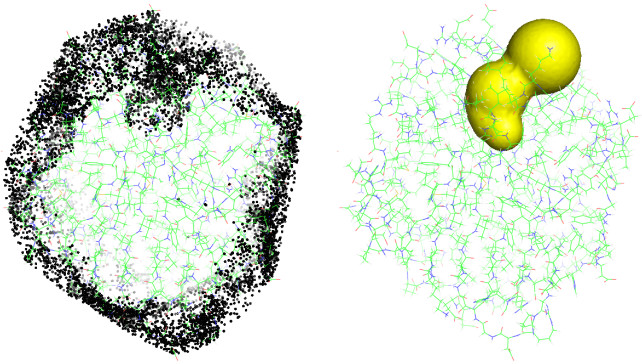

Figure 5. Example on 04_sdcl.cl (a) Edges with value above 3.0Å emphasized – the midpoint of each such edge visualized as a small sphere. The visualization is clipped with a clipping plane perpendicular to the view in order to reveal the inner parts of the molecule. (b) Computed dynamic channel. The molecule is visualized as a line model, first snapshot of the trajectory. Visualized in PyMOL [19].

Table I
COMPARISON OF BOTTLENECK RADII. THE BOTTLENECK OF A DYNAMIC CHANNEL IS COMPARED WITH THE CHANNEL WHICH HAS THE MAXIMUM BOTTLENECK IN THE WHOLE SET OF CHANNELS COMPUTED SEPARATELY IN EACH SNAPSHOT.

Trajectory	**Bottleneck radius**	
	Dynamic channel (VD G_{ini} variant)	**Set of channels** (computed separately, maximum)
wt_sdcl.cl	2.798 Å	2.021 Å
04_sdcl.cl	3.051 Å	1.968 Å
14_sdcl.cl	2.568 Å	2.079 Å
15_sdcl.cl	2.601 Å	1.725 Å

dynamic channel. The results show that there is an increase in the bottleneck radius of the dynamic channel. Such an increase with respect to the number of snapshots processed is shown in Fig. 4 (c). In practical case, the computation of a dynamic channel would be performed only once after processing the whole sequence.

In the experiments, the computed dynamic channels for both proposed G_{ini} variants led in general through the same parts of the molecule. Therefore, for the following tests, we chose the VD variant that we consider to be more prospective for its better computational time.

Fig. 5 shows the visualization of a computed dynamic channel. Edges in G_{ini} with value above certain threshold are visualized as small spheres located at midpoints of each edge (a). The surface of the channel is shown in (b).

The computed dynamic channel was compared against the channel (from the set of channels computed separately in each snapshot) with the biggest bottleneck radius. The comparison shows that the radii of dynamic channels computed for selected trajectories are significantly larger implying that the molecule is flexible at that regions and potentially a small ligand molecule may pass through. Table I illustrates the results of the comparison for protein trajectories of wild-type (wt) and mutated (04,14,15) haloalkane dehalogenase DhaA molecules.

For example, the time requirement to update G_{ini} (maximization) in one snapshot (approx. 4500 atoms), in the case VD is used, is below 8 seconds on the common desktop computer (single-threaded, 2.0GHz, 2GB RAM). We have to point out, that the time for processing the trajectory is linearly dependent on the number of snapshots in the trajectory.

After snapshots are processed, the Dijkstra's algorithm is to be run. The complexity, again, is not dependent on the number of snapshots previously processed since the topology of G_{ini} remains constant over time. The Dijkstra's algorithm running time on the previously mentioned sample G_{ini} and computer is below 5 seconds. Unlike the update of G_{ini} in each snapshot, the Dijkstra's algorithm is run only once after all snapshots are processed.

V. CONCLUSION

We have presented a method capable of computing dynamic channels. Dynamic channels are optimized for the whole trajectory and not for one particular snapshot only. Our method is able to consider the whole trajectory at once – computed dynamic channels are composed of parts from different snapshots.

 ISBN: 978-1-61208-137-3

Dynamic channels are able to highlight pulsing local neighbourhood and parts of the molecule with high flexibility. A ligand may pass through such pulsing areas.

In combination with previous methods, the proposed solution may help chemists to find possible paths which could provide an access to the active site. The residues surrounding the selected channel in the molecule could be replaced so that the active site becomes either more easily accessible or, on the contrary, inaccessible through the particular part of the molecule.

We expect that, by using this method, new information about the behaviour of various protein molecules will be revealed. After the development of sophisticated visualization methods the algorithm will be integrated into protein visualization software Caver Viewer (http://www.caver.cz).

The testing version of the algorithm was implemented in Java programming language and is available for download from http://decibel.fi.muni.cz/~xbenes2/dynChannels.

ACKNOWLEDGMENT

This work was supported by The Ministry of Education of The Czech Republic, Contract No. LC06008 and by The Grant Agency of The Czech Republic, Contract No. P202/10/1435 and GA201/09/0097. We would like to acknowledge Loschmidt Laboratories, Masaryk University for provided protein data.

REFERENCES

[1] P. Medek, P. Beneš, and J. Sochor, "Computation of tunnels in protein molecules using delaunay triangulation," *Journal of WSCG*, vol. 15(1-3), pp. 107–114, 2007.

[2] J. Liang, H. Edelsbrunner, and C. Woodward, "Anatomy of protein pockets and cavities: measurement of binding site geometry and implications for ligand design." *Protein Sci*, vol. 7, no. 9, pp. 1884–1897, September 1998. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/9761470

[3] K. Rother, P. W. Hildebrand, A. Goede, B. Gruening, and R. Preissner, "Voronoia: analyzing packing in protein structures." *Nucleic Acids Research*, vol. 37, no. Database-Issue, pp. 393–395, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/nar/nar37.html

[4] V. Le Guilloux, P. Schmidtke, and P. Tuffery, "Fpocket: An open source platform for ligand pocket detection," *BMC Bioinformatics*, vol. 10, pp. 168+, June 2009. [Online]. Available: http://dx.doi.org/10.1186/1471-2105-10-168

[5] M. Petřek, M. Otyepka, P. Banáš, P. Košinová, J. Koča, and J. Damborský, "Caver: a new tool to explore routes from protein clefts, pockets and cavities," *BMC Bioinformatics*, vol. 7, no. 1, pp. 316+, June 2006. [Online]. Available: http://dx.doi.org/10.1186/1471-2105-7-316

[6] M. Petřek, P. Košinová, J. Koča, and M. Otyepka, "Mole: A voronoi diagram-based explorer of molecular channels, pores, and tunnels." *Structure*, vol. 15, no. 11, pp. 1357–1363, November 2007. [Online]. Available: http://dx.doi.org/10.1016/j.str.2007.10.007

[7] E. Yaffe, D. Fishelovitch, H. J. Wolfson, D. Halperin, and R. Nussinov, "Molaxis: Efficient and accurate identification of channels in macromolecules," *Proteins: Structure, Function, and Bioinformatics*, vol. 73, no. 1, pp. 72–86, 2008. [Online]. Available: http://dx.doi.org/10.1002/prot.22052

[8] B. Ho and F. Gruswitz, "Hollow: Generating accurate representations of channel and interior surfaces in molecular structures," *BMC Structural Biology*, vol. 8, no. 1, p. 49, 2008. [Online]. Available: http://www.biomedcentral.com/1472-6807/8/49

[9] B. J. Alder and T. E. Wainwright, "Studies in Molecular Dynamics. I. General Method," *jcp*, vol. 31, pp. 459–466, Aug. 1959.

[10] S. K. Ldemann, V. Lounnas, and R. C. Wade, "How do substrates enter and products exit the buried active site of cytochrome p450cam? 1. random expulsion molecular dynamics investigation of ligand access channels and mechanisms," *Journal of Molecular Biology*, vol. 303, no. 5, pp. 797 – 811, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/B6WK7-45F50WB-39/2/503cb97cac5e6a3a94899d6bc61ed2a8

[11] P. Beneš, P. Medek, and J. Sochor, "Computation of channels in protein dynamics," *In Proceedings of the IADIS International Conference Applied Computing 2009*, vol. 2, pp. 251–258, 2009.

[12] F. P. Preparata and M. I. Shamos, *Computational geometry: an introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1985.

[13] M. L. Gavrilova and J. Rokne, "Updating the topology of the dynamic voronoi diagram for spheres in euclidean d-dimensional space," *Comput. Aided Geom. Des.*, vol. 20, no. 4, pp. 231–242, 2003.

[14] D. D. Barber, C.B. and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software*, vol. 22(4), pp. 469–483, 1996.

[15] P. Beneš, P. Medek, and J. Sochor, "Tracking single channel in protein dynamics," *In WSCG Communication Papers proceedings*, vol. 2, pp. 109–114, 2010.

[16] P. Benes, P. Medek, and J. Sochor, "Computation of more channels in protein molecules," *In Proceedings of Visual Computing for Biomedicine*, vol. 7, pp. 45–51, 2008.

[17] M. Klvaňa, "Structure-dynamics-function relationships of haloalkane dehalogenases," Ph.D. dissertation, Faculty of Science, Masaryk University, Brno, Czech Republic, 2010.

[18] M. Zemek and J. Skala, "Fast method for computation of channels in dynamic proteins," in *Vision, Modeling and Visualization 2008*. Heidelberg: Akademische Verlagsgesellschaft Aka, 2008, pp. 333–343.

[19] W. L. Delano, "The pymol molecular graphics system," Palo Alto, CA, USA., 2002.

 ISBN: 978-1-61208-137-3