Eurographics Workshop on Visual Computing for Biology and Medicine (2014)
I. Viola, K. Bühler, and T. Ropinski (Editors)

Visualizing movements of protein tunnels in molecular dynamics simulations

Barbora Kozlíková[†] and Adam Jurčík and Jan Byška and Ondřej Strnad and Jiří Sochor

Faculty of Informatics, Masaryk University, Brno, Czech Republic

Abstract

Analysis and visualization of molecules and their structural features help biochemists and biologists to better understand protein behavior. Studying these structures in molecular dynamics simulations enhances this understanding. In this paper we introduce three approaches for animating specific inner pathways composed of an empty space between atoms, called tunnels. These tunnels facilitate the transport of small molecules, water solvent and ions in many proteins. They help researchers understand the structure-function relationships of proteins and the knowledge of tunnel properties improves the design of new inhibitors. Our methods are derived from selected tunnel representations when each stresses some of the important tunnel properties — width, shape, mapping of physico-chemical properties, etc. Our methods provide smooth animation of the movement of tunnels as they change their length and shape throughout the simulation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—AnimationVisible line/surface algorithms

1. Introduction

Molecular analysis and visualization are currently two of the most interesting and important areas in biochemistry. Researchers more often analyze *molecular dynamics* (MD) simulations rather than static molecules, as the results are more biochemically relevant. This involves analyzing and displaying hundreds of thousands of atoms in order to reveal important features of protein molecules. While the analysis leading to the detection and categorization of the protein inner empty space can be very time and memory consuming, the visualization of results should be performed at interactive frame rates in order to make the visual analysis applicable.

The protein empty space can be categorized with respect to its shape and other properties. We distinguish between *cavities, tunnels, channels, pores* and *pockets*. Due to intrinsic protein dynamics, these specific inner void structures change their shape and properties over time [KM02, CPB*12, KPK*09]. The detection and visualization of these structures can facilitate the study of important biochemical phenomena as well as designing effective drugs or new catalysts [PGB*12, PKC*09, KCB*13, GBD13]. The design is mostly based on the study of chemical reactions between proteins and small ligand molecules. These reactions take place in a specific cavity called an *active site*. Thus, studying the entrance pathways leading to the active site has been in the scope of researchers over the last years.

In this paper we concentrate on the description of our novel techniques for animating tunnels in molecular dynamics. We discuss three different tunnel representations emphasizing different tunnel properties, such as tunnel width, shape and physico-chemical properties.

2. Related Work

Several geometric solutions for the detection of protein pathways and inner cavities appear in literature. Numerous algorithms for detecting and visualizing molecular cavities appeared as far back as several decades ago, e.g., in [HM90, AW91, Del92]. Cavities can be detected using different approaches — a grid algorithm described, e.g., in [VG10], an approximation algorithm [SNW*96] or a space partitioning structure, e.g., the Delaunay triangulation (DT) applying the alpha-shape [LWE98] or beta-complex [CKW*11] theory. The existing visualization techniques presenting the computed inner structures primarily concentrate on cap-

[†] Corresponding author, kozlikova@fi.muni.cz

DOI: 10.2312/vcbm.20141188

turing their shape and volume. When stressing the volume, void structures can be filled with a set of intersecting spheres [HG08]. The boundary mesh representation of cavities can be computed and rendered [LB92] as well. Several techniques utilize the volumetric representation of proteins which enable the direct display of cavities using ray casting [ZB07, KFR*11], for example. Parulek and Viola applied the theory of implicit surfaces to visualize molecular surfaces [PV12] and cavities [PTRV13].

When performing a visual analysis of cavities in MD, they must be animated in order to better illustrate the change in their properties (width, shape, stability, etc.) over time. Recently, several publications appeared in this field. Methods described in [PTRV13, KKRE14] are used to present changes of the shape and size of detected cavities in real-time. However, both publications concentrate on an interactive visual analysis in the form of interconnected graph statistics which capture the various properties of cavities rather than animating the 3D cavity over time. Lindow et al. in [LBBH12] and in [LBBH13] also track detected cavities over time but it is not obvious whether or not the algorithm produces an interpolated animation of the trajectory.

In this paper we present novel solutions concentrating on the smooth animation of a 3D representation of tunnels. Some of these methods are robust enough to be applied to other structures, such as cavities, as well. Our solutions are based on the dynamic tunnel detection and visualization algorithms presented in [MBS07,KAS07]. The detection algorithm tracks the tunnel using its medial axis, i.e., the centerline [POB*06]. Then, the dynamic tunnel can be visualized as a set of intersected spheres positioned onto the centerline or by visualizing the tunnel as a sequence of tetrahedra obtained using a 3D Delaunay triangulation (used to detect tunnels) through which the centerline passes.

In the tetrahedra-based visualization approach, we consider tunnels as meshes represented by the tetrahedra boundary. To present the smooth animation of such a representation, our main task is to morph (find correspondence and interpolate) [LV98] two meshes between consecutive frames. In the case of tunnels, the meshes have a rather different topology due to the triangle flips between frames of the original DT [SMH04]. As a general solution, the correspondence between faces of meshes can be determined using minimal dynamic displacement criterion [HMTT88]. Other algorithms utilize the topology as well as the geometry of the meshes. These algorithms project their inputs onto an intermediate surface (e.g., a unit sphere [KCP92] or a circular disc [KSK97]) and perform the vertex to vertex correspondence by merging the topologies, resulting in a supermesh. The vertex count of the generated supermesh can be optimized [ALS04]. It may also be important (as in our case) to preserve the volume of the mesh throughout morphing. In this case, the interior of given shapes (e.g., triangulations) rather than their boundaries can be blended [IMH05].

Since our meshes are derived from DT, we can avoid using the rather general (and expensive) morphing method. Instead, we employ the original tetrahedral representation to substantially lower the time and memory requirements of the smooth mesh interpolation, resulting in robust and real-time animation.

Our solution utilizes the tunnel representation obtained by the CAVER 3.0 [CPB*12] algorithm. This algorithm is based on a Voronoi diagram representation of the molecule and on the Delaunay triangulation. However, this representation does not take into account the different size of atoms. To overcome this problem, CAVER 3.0 uses the approximation approach where each atom is replaced by a set of spheres of radii corresponding to the radius of the smallest atom (hydrogen). Then, the new representation uses only spheres of the same radii and the basic Voronoi diagram can be applied. The second option is to utilize an additively weighted Voronoi diagram (AWVD). The results of these two approaches are comparable — the approximation method produces qualitatively similar results as the AWVD method.

3. Definitions

The novelty of this article is based on designing and developing new approaches for the animation of tunnels throughout molecular dynamics. The sequence of snapshots of molecular dynamics obtained from simulations captures the discrete states of the molecule in time. The time step used within the simulation differs according to the demands of biochemists. Generally, the simulation step is in femtoseconds and the length of the simulation reaches nanoseconds or even microseconds. Thus, we can obtain thousands and thousands of snapshots which can be viewed and evaluated. This requires an enormous amount of time so any technique simplifying this process is quite valuable. However, this problem is beyond the scope of this paper. Here we concentrate on animating the movement of proteins and their tunnels and interpolating between the discrete snapshots of the molecular dynamics. This helps to understand the continuous movements of the molecule and changes in shape of the tunnels.

In this paper we propose three different approaches to tunnel animation which are based on existing methods for visualization of tunnels in one static snapshot. Each of these methods has its pros and cons, which will be discussed later in this section. These methods are:

- tunnels represented by a set of intersecting spheres
- tunnels represented by a mesh derived from the set of intersecting spheres
- tunnels represented by a set of subsequent tetrahedra

First, it is necessary to introduce several definitions. These are crucial for understanding the proposed algorithms.

As mentioned before, the arrangement of the three-dimensional structure of proteins allows for the presence of specific inner voids (active sites) which play a crucial role in protein reactivity. Chemical reactions can occur on the protein surface or deep inside the protein. The latter case requires the active site to be connected with the protein surface by a pathway. The applicability of the methods is independent of the type of these pathways, so in the following sections we will use the term tunnel only.

A **tunnel** is defined as a continuous empty space leading from the inner void out to the protein surface. The tunnel surface is limited by its surrounding atoms. Tunnels can be calculated using different approaches and algorithms (e.g., [POB*06, CPB*12, SVB*13, YFW*08]). Our solution is derived from Voronoi diagrams and the Delaunay triangulation, which results in the representation of a tunnel by a set of successive tetrahedra (see Figure 1, top). The vertices of these tetrahedra are formed by the centers of atoms lining the tunnel. From this representation, the tunnel **centerline** can be derived — it crosses the inner triangles of the tetrahedra. The centerline can be utilized to create a second tunnel representation — a set of intersecting spheres. In this representation, spheres are positioned onto the centerline and have maximum radii with respect to the surrounding atoms (see Figure 1, bottom). This representation is widely used because it describes the tunnel width well. However, it is not as suitable in situations when a biochemist wants to map various physico-chemical properties (e.g., hydrophobicity, charge) onto the tunnel surface or to explore the inner tunnel environment. For such tasks, a mesh representation of the tunnel is more suitable (because of simpler mapping). To preserve the tunnel width, a representation covering the tunnel spheres with a mesh and resembling a tube (see Figure 1, bottom) has been developed.

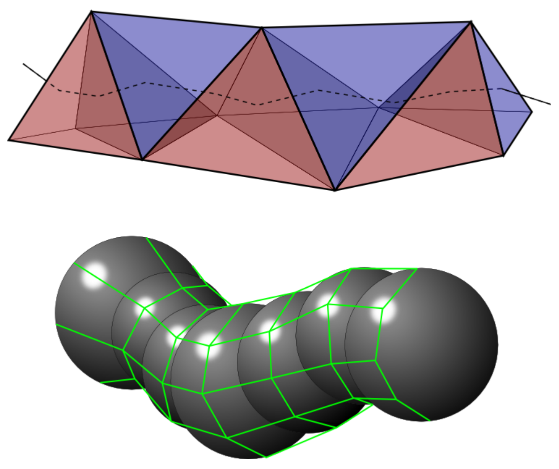
Figure 1: *Top — tunnel represented as a set of neighboring tetrahedra. The dotted polyline represents the tunnel centerline. Bottom — the tunnel as a set of intersecting spheres (grey) and illustration of the tube mesh covering these spheres (green).*

A tunnel represented by a set of subsequent tetrahedra consists of triangles which can be classified into two groups according to their position in the tetrahedra. Triangles situated in the inner part of the tetrahedra (invisible from the outside) are called **gates** (see Figure 2, green) and they intersect the tunnel centerline. Each gate is defined by three atoms whose centers form the vertices of the gate. Except for the first and the last gate of the tunnel, the gates are always shared by two neighboring tetrahedra. The second group contains those triangles on the boundary of the tunnel. We call them **bounding faces** (see Figure 2, grey).

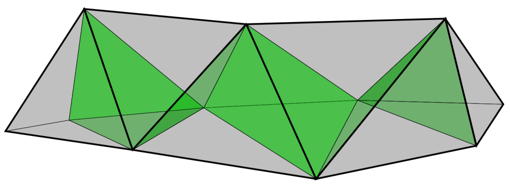
Figure 2: *Tunnel gates (green) and tunnel bounding faces (gray) forming together the tetrahedra of the tunnel.*

4. First approach — sphere tunnel animation

The simplest method smoothly animates the movement of a tunnel represented as a set of intersecting spheres. Here, we must solve the correspondence between the tunnel spheres in two snapshots. Problems can occur in situation when the number of spheres representing the tunnel in these two snapshots is different. In this case, the correspondence can be derived from the tunnel gates because each gate contains exactly one sphere from the original tunnel sphere representation. More formally, lets identify two tunnels, t_1 and t_2. These tunnels are defined by their lists of gates $G_1 = [g_{11}, ..., g_{1n}], G_2 = [g_{21}, ..., g_{2m}]$ respectively. Our goal is to create the tunnel t representing the transition between t_1 and t_2 and to determine its list of gates $G = \{\{a, b\} : a \in G_1, b \in G_2\}$. This means that each gate from G references one gate from G_1 and one gate from G_2. To attain this, we can select one of the following strategies:

- Linear stretch
- Linear interpolation with respect to surrounding atoms
- Cubic interpolation with respect to surrounding atoms

4.1. Linear stretch

First, we mark the lists of gates G_1 and G_2 with respect to their length (number of gates). The longer list is marked as G_{max} and the shorter one as G_{min}. The final list G will have the same length as G_{max} and each item h_i will be computed as $h_i = \{G_{max}[i], G_{min}[round(i * \frac{n_{min}}{n_{max}})]\}$, where n_{min} and n_{max} is the number of items in G_{min} and G_{max}. As a result, each item from G_{max} will be referenced only once, and each item from G_{min} will be referenced at least once. In the final animation, this occurs as dividing some gates (sphere) or collapsing more spheres into one.

4.2. Linear interpolation with respect to atoms

In this case, we take into account the atoms forming gates of G_1 and G_2 (see Figure 3). First, we add to G those pairs of gates $\{a \in G_1, b \in G_2\}$ where a and b are formed by the same triplet of atoms. Such gates are subsequently removed from the original G_1 and G_2 lists which causes the break up of these lists to several smaller lists. The newly created lists are then mapped using the previously described linear stretch approach. Using this approach, we can avoid the situation when gates consisting of the same atoms are mapped to other gates (the linear stretch applied to the whole lists of gates cannot ensure this condition).

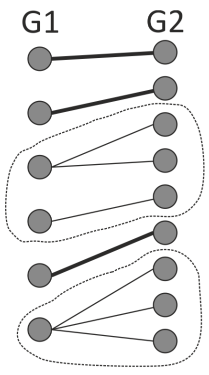

Figure 3: *Correspondence between two lists of gates when using linear interpolation with respect to surrounding atoms. Thick lines represent pairs of gates formed by the same atoms, the dotted lines mark the subsets of gates processed by the linear stretch approach.*

4.3. Cubic interpolation with respect to atoms

The two aforementioned approaches can produce slightly scattered animations. However, thanks to their simplicity, they can be used for the real-time animation of tunnels in complex molecular structures consisting of hundreds of thousands of atoms. In order to produce a smooth and plausible animation, we introduce another approach which uses the cubic interpolation between tunnels in different snapshots. Here, we calculate the interpolated tunnel from four consecutive snapshots. This means finding the correspondence between four lists of gates, G_1, ..., G_4. This is performed pairwise — for each pair of lists from neighboring snapshots (G_1 and G_2, G_2 and G_3, G_3 and G_4) using the linear interpolation with respect to the surrounding atoms. Then, we simply join the results.

When computing the interpolation spline, gates from G_2 and G_3 define the positions of spline end points (lets mark them *start* and *end*) and gates from G_1 and G_4 determine the normals in these end points. It is necessary to add a normal to *start* and *end* points. On the other side, not every normal must be assigned to some end point. This can happen when some point p from G_2 is assigned with two or more points from G_1. In such a case only one point from G_1 (e.g., the first one) is chosen, others are ignored (see empty circles in Figure 4).

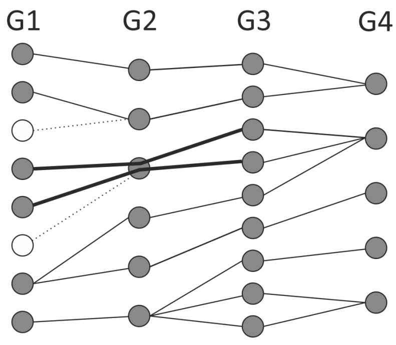

Figure 4: *Correspondence for the tunnel in four consecutive snapshots.*

However, there can be situations when more points from G_1 are used. Such a situation occurs when the corresponding point p from G_2 maps onto more points from G_3. Then, the sphere represented by this point p is divided and branching of the tunnel appears. For these branches it is advisable to use different normals. We can use, for example, the first n gates (if possible) from G_1 belonging to p, where n is the number of branches. This situation is highlighted by thick lines in Figure 4.

5. Second approach — tube mesh tunnel animation

The second representation is derived from the previous one, which uses a set of spheres. This representation, which we call a tube mesh because of its shape, preserves the tunnel width, which is one of the most important properties of tunnels. Moreover, this technique is more suitable when mapping other properties (physico-chemical) onto the tunnel surface. When the mesh is visualized as a semi-transparent surface, it also enables a biochemist to walk through the tunnel and observe the surrounding atoms.

When searching for the correspondence between two tube meshes, we can utilize the results of the method described in Section 4. Therefore, even when the final representation forms the mesh, the interpolation is performed on tunnel spheres. When the correspondence is determined, the intermediate mesh is constructed and interpolated.

The process of transforming a set of spheres into a tube mesh is as follows (see Figure 5):

1. For each sphere we find the plane passing through its center, which is orthogonal to the tunnel centerline. By intersecting this plane with its corresponding sphere we obtain a circle.

2. For each pair of neighboring circles we create a mesh connecting them. This mesh resembles a tube.
3. To close the tunnel, the first and the last circle is capped by cones with a height set to the radius of these circles.
4. We use Loop subdivision to smooth the mesh. Here it is sufficient to set the subdivision step to 1.

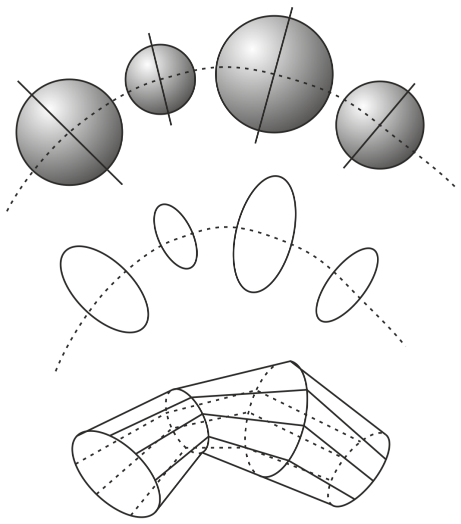
Figure 5: *Basic steps of creating the tunnel tube mesh from a set of spheres.*

6. Third approach — tetrahedra tunnel animation

The third tunnel visualization is derived from its tetrahedral representation, and it displays a mesh that can be of an arbitrary subdivision level. The most challenging part is again to determine the correspondence between individual vertices on the surface mesh of the tunnel in two different configurations. We can utilize the fact that the mesh was created from tetrahedra and thus we can search for the correspondence on the original tetrahedral representation. Then, the information will be transferred backwards to the original mesh. The idea of finding the correspondence between two tunnels represented by their thetrahedra is based on detecting three polylines on the tunnel surface. When the polylines on both tunnel configurations are detected, we can interpolate between them and, as the last step, reconstruct the tunnel mesh.

6.1. Correspondence between mesh vertices

The whole process of searching the correspondence between two meshes representing a tunnel in different time steps can be divided into three main phases.

6.1.1. Searching for the polylines on the tunnel surface

Each tunnel is represented as a linear sequence of tetrahedra. Such a tunnel contains exactly three possible polylines (linear sequences of edges of bounding faces) that fulfill the following conditions (see Figure 6):

- The polyline starts in one of the vertices of the first gate.
- The polyline ends in one of the vertices of the last gate.
- The polyline does not contain any edge of any gate. In other words, it follows edges belonging only to one tetrahedron. Each tetrahedron possesses exactly one such edge which is not shared with any neighbor (except for the first and the last tetrahedron) which makes the solution unique.

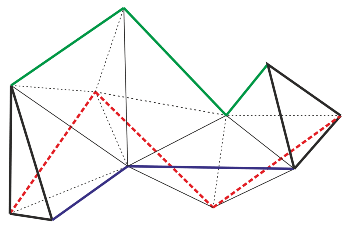
Figure 6: *Three polylines (red, green, blue) on the tunnel surface.*

6.1.2. Correspondence between snapshots

When the triplets of polylines in two tunnel configurations are known, we can search for the correspondence between them. In other words, for each polyline in the first configuration we must determine its corresponding polyline in the second configuration. Here we can utilize the fact that each vertex of any gate belongs to one of the three polylines so we can start searching the correspondence from a specific gate and its vertices in both configurations. The correspondence between vertices of gates can be then determined by measuring the distance between them (see Figure 7). In our case, we choose the gates which are closest to the active site from both configurations as the starting gates. This is due to the fact that this area of the protein is the least variable within molecular dynamics simulations. Thus we expect that the distance between vertices is minimal.

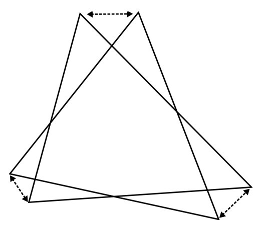
Figure 7: *The correspondence between vertices of a gate in two snapshots. Each triangle represents the state of the gate in one snapshot and the distance between vertices is calculated.*

6.1.3. Correspondence between vertices of polylines

This step is straightforward as we have the information about the correspondence of polylines, obtained from the previous step. Here we need to interpolate two linear lists of vertices. Firstly, we define the mapping between gates (using one of the methods described in Section 4). We obtain the list $G = \{\{a,b\} : a \in G_1, b \in G_2\}$. Then two vertices v_1 and v_2 correspond to each other if there are gates x and y such that v_1 is the vertex of x and v_2 is the vertex of y and $\{x,y\} \in G$.

6.2. Interpolation

When animating protein tunnels we face several situations which require specific interpolation. In this section we describe these situations along with the solution we have chosen for their interpolation.

6.2.1. Interpolation using circular pathways

Lets consider the situation when, during the animation, one gate has to be rotated 180° (see Figure 8, left part). In this case, the interpolation, which tries to maintain the minimal distance between the initial and final vertex positions, causes the triangle to be shrunk to one point in the first half of the animation and in the second half this point expands to a new triangle with the desired final positions of its vertices. This behavior is unwanted because it evokes changes in the tunnel width which are incorrect.

To overcome this problem, we shift the points on circular pathways instead of the shortest ones (see Figure 8, right part). The shifting consists of a translation of the circle center and rotation of the point. The center of each such circle is positioned in the center of its corresponding gate (when belonging to more gates at once, the average position is computed). The center of the circle follows the line pathway heading from the center of the starting gate to the center of the final gate. The plane where the circle lies is determined by two vectors vec_1 and vec_2 where vec_1 is derived from the position of the point in the starting position and the center of the first gate and vec_2 is computed identically at the end position.

6.2.2. Interpolation between tunnels crossing atoms

Another situation occurs quite often when a part of the tunnel (or even the whole tunnel) substantially changes its position within the protein. This is caused by movements of the atoms defining the tunnel gates. In this situation we again use the interpolation, maintaining the minimal distance between interpolated points, and the tunnel could pass through atoms, which is undesirable (see Figure 9).

The correct visualization should avoid this artifact. Therefore, the animation displays the tunnel jumping to the new position without intersecting any atom. This better corresponds to actual tunnel behavior. Moreover, the snapshots

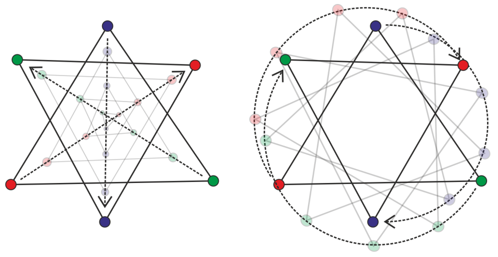
Figure 8: *Interpolation between two positions of a gate, rotated by 180°. Left — interpolation maintaining the minimal distance between start and end positions. Right — interpolation using circular pathways. Transparent parts in both figures represent the intermediate states of the animation.*

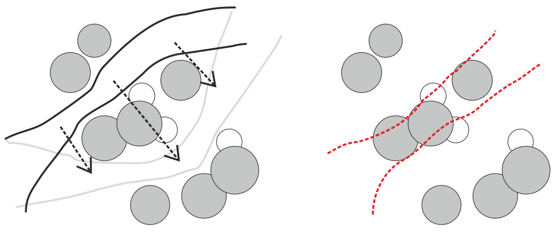
Figure 9: *Interpolating the tunnel position using minimal distance rule. In this case the tunnel can intersect with atoms.*

of molecular dynamics capture the protein state in discrete time steps and we cannot claim that we precisely know what happened to the tunnel between the neighboring snapshots. We only know that the tunnel cannot pass through any atom at any time.

However, the jumping between individual snapshots causes avoidable flickering. When the tunnel between snapshots changes only slightly, we can use the interpolation to produce a smooth animation. To determine when the tunnel changes slightly or substantially, we have to set two thresholds. Both can be derived from the distance between tunnel tetrahedra in two snapshots. When the distance between tunnel tetrahedra in two consecutive snapshots is less than 0.9 Ångströms (size of hydrogen atom), it is ensured that the interpolation does not cross any atom and we can interpolate without any restrictions. The second threshold, experimentally set to 5 Ångströms, defines that the tunnel changes substantially. In this case, the tunnel can jump directly to a new position or, to make the animation smoother, it can gradually disappear and appear again on different site. When the distance is between these two thresholds, we face a situation where the tunnel crosses through atoms (see Figure 9). We solved this by introducing the successive disappearance and appearance of the given tunnel part within the animation as illustrated in Figure 10.

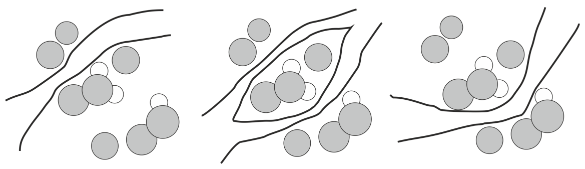

Figure 10: *Illustration of morphing the tunnel position between three snapshots. The given part, which should be moved, narrows and finally disappears. This part appears again in the final position, where it successively broadens.*

6.3. Mesh reconstruction

All of the interpolation methods described until now worked with a triplet of polylines representing the tunnel. However, the tunnel was originally represented as a mesh. To present the animated tunnel mesh, we must reconstruct its triangulation from the polylines. This is performed for each pair of polylines separately (when marking the polylines as A, B and C, the pairs will be A-B, B-C and C-A) (see Figure 11).

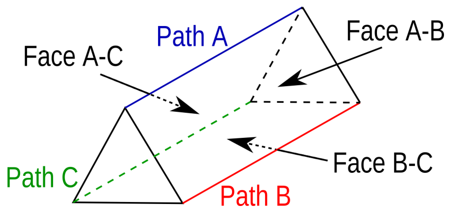

Figure 11: *Three polylines and their corresponding faces.*

Algorithm 1 Triangle strip reconstruction between polylines

```
Require: $A = [a_1, a_2, ..., a_n]$, $B = [b_1, b_2, ..., b_m]$
 1: $p_a \leftarrow a_1$
 2: $p_b \leftarrow b_1$
 3: for all $g \in 0, ..., NumOfGates - 1$ do
 4:     if $g = last(p_a)$ then
 5:         Add triangle $p_a, p_{a+1}, p_b$
 6:         $p_a \leftarrow p_{a+1}$
 7:     end if
 8:     if $g = last(p_b)$ then
 9:         Add triangle $p_b, p_{b+1}, p_a$
10:         $p_b \leftarrow p_{b+1}$
11:     end if
12: end for
```

The algorithm 1 reconstructs the triangle strip between two polylines. Lets consider G as the list of tunnel gates, sorted according to their position in the tunnel, and two polylines, A and B, which are determined by their lists of vertices: $A = [a_1, a_2, ..., a_n]$, $B = [b_1, b_2, ..., b_m]$. Each vertex a_i (b_i respectively) can correspond to one or more gates. For each such vertex we select its corresponding gate with the highest index (with respect to the ordering taken in G). For vertex a_i, mark this gate as $last(a_i)$. Then, the algorithm processes all gates of the tunnel. We utilize the fact that each gate can be *last* for exactly one vertex. Additionally, the position of this vertex will be updated — shifted to its new $i+1$ position (see Figure 12).

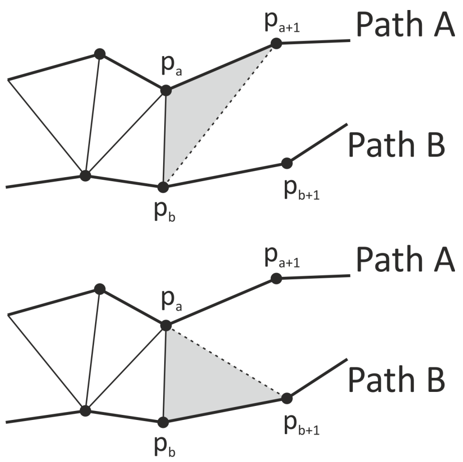

Figure 12: *Creating triangle strip between two polylines. Each illustration corresponds to one condition in the pseudocode 1.*

The reconstructed mesh is quite rough. When visualizing this surface mesh, we use the Loop subdivision scheme [Loo87] to produce a smooth surface. When using the scheme for animations, it is crucial to balance the smoothness of the surface and the performance. Experimentally we decided to set the subdivision step to 2.

6.4. Results

All three proposed solutions for tunnel animation were tested on MD simulations obtained from biochemists. They capture the real movements of the DhaA haloalkane dehalogenase wild type and its different mutations. The largest tested dataset contains 20,000 snapshots of DhaA dynamics. The length of explored tunnels stretches from 15 to 50 Ångströms. The results obtained by the testing of the proposed algorithms can be viewed from different perspectives. We observed the performance of these techniques in order to evaluate their applicability in real-time. Then, we concentrated on the usability and the predicted value from the biochemical point of view.

Table 1 shows the performance of all techniques (illustrated in Figure 13). It shows the comparison between animation frame rates when animating more tunnels at once. As the animation of tunnel spheres is the most simple technique, its performance when animating more tunnels decreases only slightly. The performance of the other two approaches decreases when animating more tunnels at once.

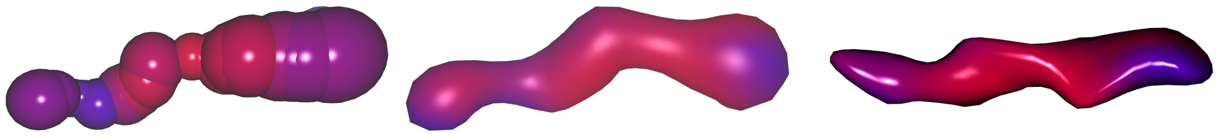

Figure 13: *Three different visualization styles of tunnels utilized in our animation techniques — from the left: spheres, tube or tetrahedra. Coloring according to hydrophobicity of surrounding amino acids is mapped onto the tunnel surface. The distribution of coloring is better in the tube and mesh representations.*

# of tunnels	spheres	tube	tetrahedra
1 tunnel	98.8	90.5	85.6
5 tunnels	97.3	64.3	39.3
10 tunnels	96.8	48.9	25.4

Table 1: *Comparison between animation frame rates (in FPS) when animating more tunnels at once.*

However, the length of the simulation does not influence the overall performance as the interpolation is calculated at most from four subsequent snapshots.

The test was performed on an Intel Core i5 3470 3.2GHz CPU, 16GB RAM, Win7 64-bit, and it is obvious that all three techniques reached real-time rates enabling interactive visual analysis of reasonable and meaningful amounts of tunnels in molecular dynamics. From a biochemical point of view, the animation speed is mostly very limited because the main aim of the animation is to explore the tunnel and understand its behavior. Moreover, exploring more tunnels at once is impractical and does not lead to a better understanding of the protein structure.

All methods were evaluated by biochemists as well. The first approach animating the tunnel spheres was perceived very positively because it captures the tunnel width and is easy to understand. Moreover, this representation is common in existing visualization tools so it is familiar to biochemists. The only problem of this technique is appears mainly when interpolating between tunnels crossing atoms. When shrinking the tunnel on one side or broadening it on the other side, the tunnel seems to be discontinuous — tunnel spheres become too small and do not intersect (see Figure 14 top). This could be solved by adding another spheres to the tunnel. However, the visual appearance of such solution was not acceptable by the biochemists. Another solution provides the tube mesh tunnel representation which better describes the tunnel shape (see Figure 14 bottom). The tube mesh and tetrahedra representations also enable more intuitive mapping of physico-chemical properties onto the tunnel surface. Moreover, the tetrahedra tunnel can capture the mutual circumfluence of neighboring tunnels. On the other hand, in the representation derived from tetrahedra, the tunnel changes more dynamically which is sometimes confusing for biochemists.

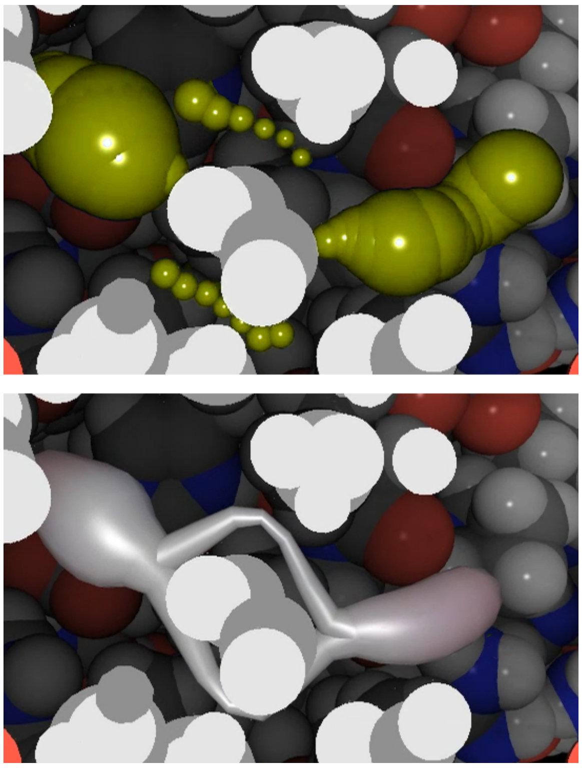

Figure 14: *Tunnel discontinuities when using the tunnel sphere representation (top). Tube mesh representation (bottom) solves this problem.*

6.5. Conclusion

We have proposed three methods for animating protein tunnels in molecular dynamics simulations. These methods are derived from existing visualization techniques designed primarily for static structures. We extended these methods by adding specific morphing techniques which enable the smooth animation of tunnels. This helps biochemists and biologists in visual analysis and exploration of detected dynamic tunnels. Methods were designed and tested in cooperation with biochemists and can serve as useful guidance for large molecular dynamic simulations.

6.6. Acknowledgments

We would like to thank to the biochemists from Loschmidt Laboratories at the Masaryk University for their invaluable help with testing and evaluating the proposed techniques.

References

[ALS04] AHN M., LEE S., SEIDEL H.: Connectivity transformation for mesh metamorphosis. *In SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2004), 75–82. 2

[AW91] ALARD P., WODAK S. J.: Detection of cavities in a set of interpenetrating spheres. *Journal of Computational Chemistry 12*, 8 (1991), 918–922. 1

[CKW*11] CHO Y., KIM J.-K., WON C.-I., RYU J., KIM C.-M., KIM D.-S.: BetaMol: Molecular modeling, analysis, and visualization software based on the beta-complex derived from the Voronoi diagram. In *Voronoi Diagrams in Science and Engineering (ISVD), 2011 Eighth International Symposium on* (2011), IEEE, pp. 48–57. 1

[CPB*12] 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 8*, 10 (2012), e1002708. 1, 2, 3

[Del92] DELANEY J. S.: Finding and filling protein cavities using cellular logic operations. *Journal of Molecular Graphics 10*, 3 (1992), 174–177. 1

[GBD13] GORA A., BREZOVSKÝ J., DAMBORSKÝ J.: Gates of enzymes. *Chemical Reviews 113* (2013), 5871–5923. 1

[HG08] HO B. K., GRUSWITZ F.: HOLLOW: generating accurate representations of channel and interior surfaces in molecular structures. *BMC Structural Biology 8*, 1 (2008), 49. 2

[HM90] HO C. M., MARSHALL G. R.: Cavity search: an algorithm for the isolation and display of cavity-like binding regions. *Journal of Computer-Aided Molecular Design 4*, 4 (1990), 337–354. 1

[HMTT88] HONG T. M., MAGNENAT-THALMANN N., THALMANN D.: A general algorithm for 3-D shape interpolation in a facet-based representation. *Proceedings of Graphics Interface '88* (1988). 2

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 1134–1141. 2

[KAS07] KOZLÍKOVÁ B., ANDRES F., SOCHOR J.: Visualization of tunnels in protein molecules. In *Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa* (2007), ACM, pp. 111–118. 2

[KCB*13] KOUDELÁKOVÁ T., CHALOUPKOVÁ R., BREZOVSKÝ J., PROKOP Z., ŠEBESTOVÁ E., HESSELER M., KHABIRI M., PLEVAKA M., KULIK D., SMATANOVÁ I. K., ŘEZÁČOVÁ P., ETTRICH R., BORNSCHEUER U. T., DAMBORSKÝ J.: Engineering enzyme stability and resistance to an organic cosolvent by modification of residues in the access tunnel. *Angewandte Chemie International Edition*, 52 (2013), 1959–1963. 1

[KCP92] KENT J. R., CARLSON W. E., PARENT R. E.: Shape transformation for polyhedral objects. In *ACM SIGGRAPH Computer Graphics* (1992), vol. 26, ACM, pp. 47–54. 2

[KFR*11] KRONE M., FALK M., REHM S., PLEISS J., ERTL T.: Interactive exploration of protein cavities. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 673–682. 2

[KKRE14] KRONE M., KAUKER D., REINA G., ERTL T.: Visual analysis of dynamic protein cavities and binding sites. In *Pacific Visualization Symposium (PacificVis), 2014 IEEE* (2014), IEEE, pp. 301–305. 2

[KM02] KARPLUS M., MCCAMMON J. A.: Molecular dynamics simulations of biomolecules. *Nature Structural & Molecular Biology 9*, 9 (2002), 646–652. 1

[KPK*09] KLVAŇA M., PAVLOVÁ M., KOUDELÁKOVÁ T., CHALOUPKOVÁ R., DVOŘÁK P., PROKOP Z., STSIAPANAVA A., KUTÝ M., KUTÁ SMATANOVÁ I., DOHNÁLEK J., KULHÁNEK P., WADE R. C., DAMBORSKÝ J.: Pathways and mechanisms for product release in the engineered haloalkane dehalogenases explored using classical and random acceleration molecular dynamics simulations. *Journal of Molecular Biology 392*, 5 (2009), 1339–1356. 1

[KSK97] KANAI T., SUZUKI H., KIMURA F.: 3D geometric metamorphosis based on harmonic map. In *Computer Graphics and Applications, 1997. Proceedings., The Fifth Pacific Conference on* (1997), IEEE, pp. 97–104. 2

[LB92] 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 10*, 4 (1992), 229–234. 2

[LBBH12] LINDOW N., BAUM D., BONDAR A., HEGE H.-C.: Dynamic channels in biomolecular systems: Path analysis and visualization. In *Biological Data Visualization (BioVis), 2012 IEEE Symposium on* (2012), IEEE, pp. 99–106. 2

[LBBH13] LINDOW N., BAUM D., BONDAR A.-N., HEGE H.-C.: Exploring cavity dynamics in biomolecular systems. *BMC Bioinformatics 14*, Suppl 19 (2013), S5. 2

[Loo87] LOOP C.: Smooth subdivision surfaces based on triangles. *Thesis (M.S.)–Dept. of Mathematics, University of Utah* (1987). 7

[LV98] LAZARUSY F., VERROUSTZ A.: 3D metamorphosis: a survey. *The Visual Computer 14* (1998), 8–9. 2

[LWE98] LIANG J., WOODWARD C., EDELSBRUNNER H.: Anatomy of protein pockets and cavities: measurement of binding site geometry and implications for ligand design. *Protein Science 7*, 9 (1998), 1884–1897. 1

[MBS07] MEDEK P., BENEŠ P., SOCHOR J.: Computation of tunnels in protein molecules using Delaunay triangulation. *Journal of WSCG 15*, 1-3 (2007), 107–114. 2

[PGB*12] PROKOP Z., GORA A., BREZOVSKÝ J., CHALOUPKOVÁ R., ŠTĚPÁNKOVÁ V., DAMBORSKÝ J.: *Engineering of Protein Tunnels: Keyhole-lock-key Model for Catalysis by the Enzymes with Buried Active Sites.* Lutz, S., Bornscheuer, U.T. (Eds.), Protein Engineering Handbook, Wiley-VCH, Weinheim, 2012. 1

[PKC*09] PAVLOVÁ M., KLVAŇA M., CHALOUPKOVÁ R., BANÁŠ P., OTYEPKA M., WADE R., NAGATA Y., DAMBORSKÝ J.: Redesigning dehalogenase access tunnels as a strategy for degrading an anthropogenic substrate. *Nature Chemical Biology*, 5 (2009), 727–733. 1

[POB*06] 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 7*, 316 (2006). 2, 3

[PTRV13] PARULEK J., TURKAY C., REUTER N., VIOLA I.: Visual cavity analysis in molecular simulations. *BMC Bioinformatics 14*, Suppl 19 (2013), S4. 2

[PV12] PARULEK J., VIOLA I.: Implicit representation of molecular surfaces. In *Pacific Visualization Symposium (PacificVis), 2012 IEEE* (2012), IEEE, pp. 217–224. 2

[SMH04] SCHALLER G., MEYER-HERMANN M.: Kinetic and dynamic delaunay tetrahedralizations in three dimensions. *Computer Physics Communications 162*, 1 (2004), 9–23. 2

[SNW*96] SMART O. S., NEDUVELIL J. G., WANG X., WALLACE B., SANSOM M. S.: HOLE: a program for the analysis of the pore dimensions of ion channel structural models. *Journal of Molecular Graphics 14*, 6 (1996), 354–360. 1

[SVB*13] SEHNAL D., VAŘEKOVÁ R. S., BERKA K., PRAVDA L., NAVRÁTILOVÁ V., BANÁŠ P., IONESCU C.-M., OTYEPKA M., KOČA J.: MOLE 2.0: advanced approach for analysis of biomacromolecular channels. *Journal of Cheminformatics 5* (2013), 39. 3

[VG10] VOSS N. R., GERSTEIN M.: 3V: cavity, channel and cleft volume calculator and extractor. *Nucleic Acids Research 38* (2010), W555–W562. 1

[YFW*08] YAFFE E., FISHELOVITCH D., WOLFSON H. J., HALPERIN D., NUSSINOV R.: MolAxis: a server for identification of channels in macromolecules. *Nucleic Acids Research 36*, Web-Server-Issue (2008), 210–215. 3

[ZB07] ZHANG X., BAJAJ C.: Extraction, quantification and visualization of protein pockets. In *Computational Systems Bioinformatics Conference* (2007), vol. 6, pp. 275–286. 2