Eurographics Workshop on Visual Computing for Biology and Medicine (2017)
S. Bruckner, A. Hennemuth, and B. Kainz (Editors)

Watergate: Visual Exploration of Water Trajectories in Protein Dynamics Supplementary Material

V. Vad[1†], J. Byška[2†], A. Jurčík[3], I. Viola[1], E. M. Gröller[1], H. Hauser[2], S. M. Marques[3,4], J. Damborský[3,4], and B. Kozlíková[3]

[1]TU Wien, Austria
[2]University of Bergen, Norway
[3]Masaryk University, Czech Republic
[4]International Clinical Research Center, St. Anne's University Hospital, Czech Republic

1. Introduction

To better demonstrate the usability of Watergate, the supplementary material contains the description the second case study conducted by the protein engineers. In order to better support the reproducibility of Watergate, we also provide the readers with details about the bandwidth matrix estimation used in our approach. We also measured the performance of the DensityIsosurface representation and we provide the comparison with other existing solutions.

1.1. Case Study

In the second case study we analyzed a molecular dynamics simulation containing the DhaA haloalkane dehalogenase protein interacting with water molecules in the presence of 2,3-dichloropropan-1-ol, DCP, which is the product of the dehalogenation reaction of 1,2,3-dichloropropane. This simulation contains 100.000 time steps, which corresponds to 200 ns. Within the simulation, the DCP ligand repeatedly travels in between the protein active site and the bulk solvent, spending a significant portion of time outside the protein. The aim of this study was to verify whether the protein's main tunnel was filled with water molecules in the absence of DCP inside or not.

Watergate filtered out all water molecules which did not enter into the protein's structure during the simulation. The remaining 200 water molecules were further examined. First, the biochemists used the TimeLine View to understand when and how long the water molecules were inside the protein and the active site region. Figure 1 shows the TimeLine View vertically sorted according to the time the water molecules spent inside the protein.

Except for several water molecules which entered the protein and stayed for a longer time period (long lines), there was a significant amount of water molecules which entered the protein only for a fraction of time. These can be observed as very short lines or points in the bottom part of the TimeLine View. Additionally, we can observe that there are two significant periods when the water molecules entered the protein in a larger amount (marked with blue rectangles).

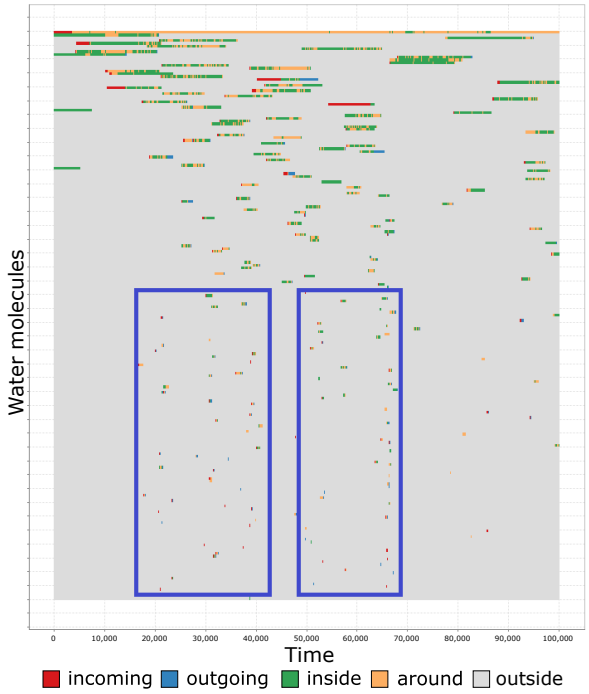


Figure 1: *TimeLine View sorted according to the time period spent by water molecules inside the protein. Blue rectangles show two parts of the simulation where the water molecules were intensively flowing to the protein and immediately left again.*

It is visible that these water molecules mostly started to enter the protein around time step 20.000 (40 ns) and this trend continued approximately until time step 65.000 (130 ns). This corresponds to the period when DCP was mostly outside the protein structure (see Figure 2).

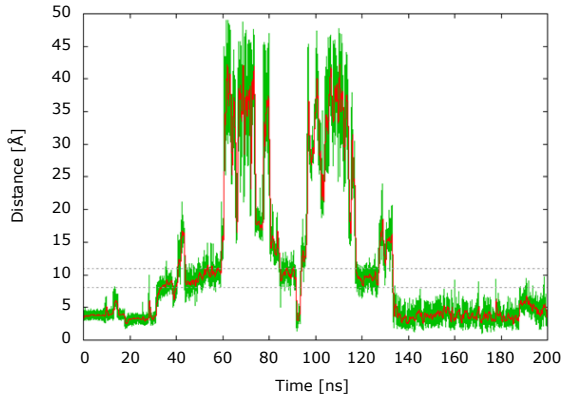


Figure 2: *Time variation of the ligand's distance to the active site (distance as the green line and the respective moving average as the red line). The horizontal dashed lines represent the tunnel mouth, the approximate thresholds to the protein's surface.*

The same trend is visible when we rank the TimeLine View according to time when the water molecules first entered the protein (see Figure 3). It is clear that in two parts of the simulation the water molecules entered the protein in a larger amount and stayed inside only for a very limited number of time steps. This is even more visible around time step 65.000.

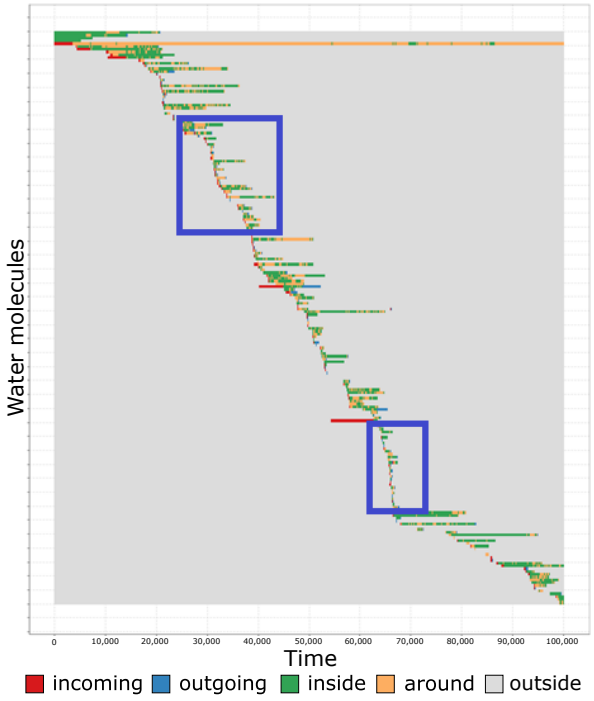


Figure 3: *TimeLine View sorted according to the time when the water molecules first entered the protein. Blue rectangles highlight parts of the simulation, when the water molecules almost simultaneously started to enter the protein and mostly stayed only for few time steps.*

The TimeLine View suggested interesting parts of the simulation and the corresponding water molecules which were worthwhile to explore in more detail. In the next step, the biochemists wanted to find out which tunnels these water molecules used to enter or exit the protein. Instead of the traditional exploration in 3D view, they used the WaterFlow Map which immediately revealed that most of the water molecules entered the protein via the main tunnel (see Figure 4).

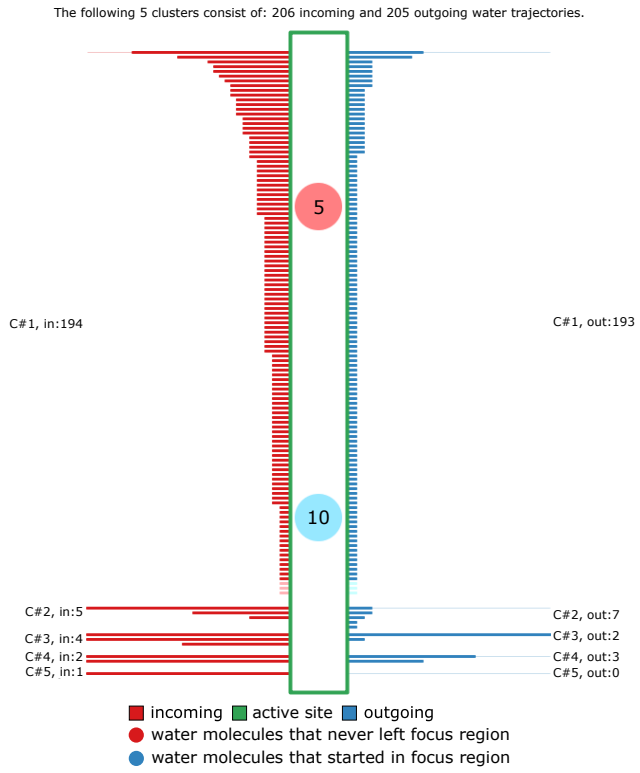
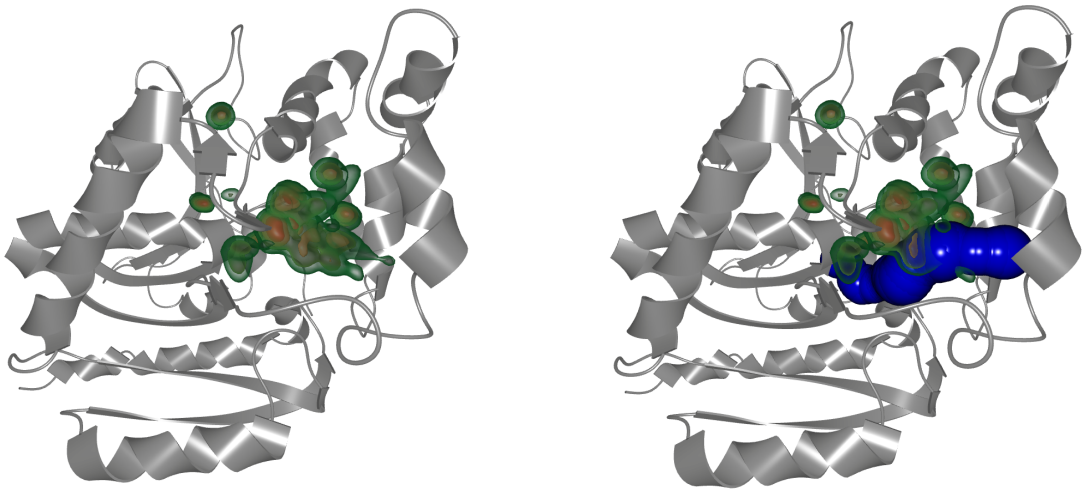


Figure 4: *WaterFlow Map showing the selected water molecules and their entrance tunnels.*

This confirmed the importance of the main tunnel for the transport of ligands and solvent (water), and in the final stage of the analysis, the results were compared in more detail with the information about DCP ligand and its passage through the main tunnel. For this purpose, the biochemists used the graph representation plotting the distance of DCP ligand from the active site (see Figure 2). The graph clearly shows that DCP left the active site region twice, which is represented by two peaks of the green line. The first peak starts around 40 ns and ends around 80 ns, the second peak starts around 100 ns and ends around 130 ns. When comparing this with our TimeLine View, it clearly shows the correlation between these two time periods. In both periods the ligand left the protein and the main tunnel immediately started to be used by a non-trivial amount of water molecules. However, most of them entered the protein's structure only for a small fraction of time. This clearly shows the relationship between the ligand's position and the water molecules: when the ligand leaves the active site, the water molecules tend to fill in the void space of the main tunnel, and when DCP reenters



(a) **(b)**

Figure 5: *(a) DensityIsosurface representation for selected water molecules. (b) Additional visualization of the main tunnel computed by CAVER 3.0 tool.*

the tunnel, the number of waters flowing through this tunnel drops again.

Finally, the DensityIsosurface representation showed the occupancy of the protein inner space by the selected water molecules (see Figure 5(a)). Additionally, the biochemists overlapped the isosurfaces with the tunnel computed using CAVER 3.0 tool (Figure 5(b)).

It is visible that the water molecules were located inside or in the close vicinity of the CAVER tunnel representation. It has to be stated that this tunnel representation underestimates its shape and volume, therefore the isosurfaces are not completely inside this representation. Additionally, it shows the tunnel in only a single time step and its shape is changing over time. This visualization confirmed that the selected water molecules were floating through the main tunnel. This representation helped to understand the spatial distribution of water molecules within the main tunnel.

1.2. Kernel Density Estimation

Before describing the details about the bandwidth matrix estimation used in our approach, we explain the rationale behind the selection of the KDE approach.

Our intention was to estimate the underlying probability density from the trajectory positions without any parametric model assumptions. For this purpose, there are two possible solutions described in literature. The simplest option is to estimate a histogram and after its normalization we could get a discretized estimate of the density. The other option is to use KDE. It is known that KDE has a better approximation power (Chapter 20 in [Was10]) with assuming to know the optimal bandwidth but with much higher computational cost. To tackle this problem, an approximation technique has been used in the literature for decades. Instead of computing the KDE directly, a linear binning grid is used [Wan94], and the resulted grid is linearly filtered. This technique could be seen as a smoothed histogram estimation. From the visualization perspective, the KDE method is more preferable since it gives a continuous result. Therefore, more precise percentile estimation is achievable. For a low resolution grid the histogram gives similar results, however we miss the fine details. If we increase the resolution of the binning grid, the histogram technique converges to the sample distribution, meanwhile the binned KDE converges to the directly computed version. To demonstrate this, we created a test scheme out of our framework. We used ∼1.5 M trajectory points (Figure 6), then computed the histogram, and linearly binned the KDE estimation on grids with resolutions 32^3, 128^3, and 256^3. For both methods we rendered isosurfaces corresponding to the 25% percentile. The results can be seen in Figure 7.

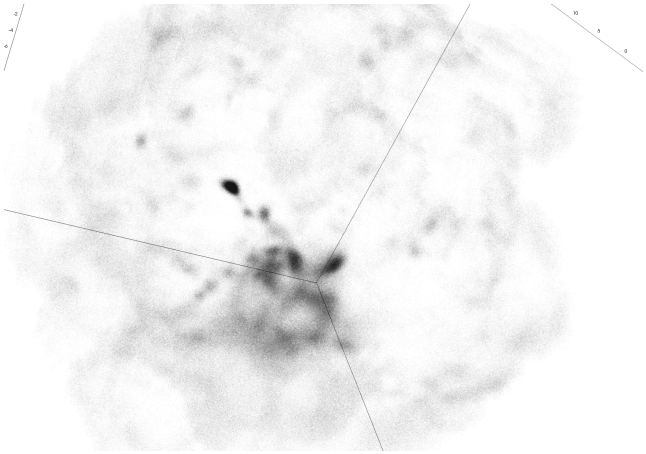


Figure 6: *∼1.5 M trajectory points, rendered with low opacity.*

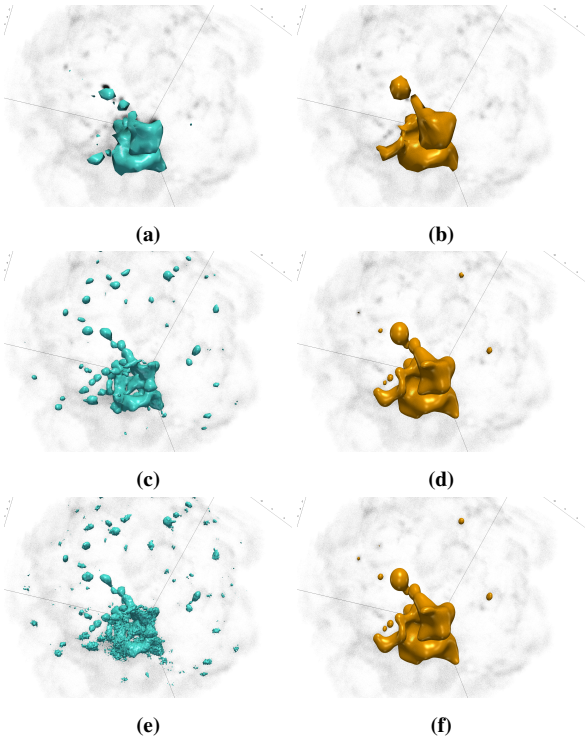


Figure 7: *Isosurfaces representing 25% percentiles. (a),(c),(e) histogram estimation, (b),(d),(f) linearly binned KDE, (a),(b) grid resolution is* 32^3*, (c),(d) grid resolution is* 128^3*, (e),(f) grid resolution is* 256^3*.*

1.2.1. Binned Evaluation and Bandwidth Matrix Estimation

Instead of evaluating function f directly, we use a fast approximation of Equation 2 (in paper). An early version of this approximation was proposed by Wand [Wan94]. He summed the input samples into a linearly weighted binning grid. Then the estimation of Equation 2 was achieved by a discrete convolution of the binning grid and the discretized $K_{\mathbf{H}}$ matrix, using the Fast Fourier Transform (FFT). In our framework, we use its recently published generalization to unconstrained bandwidth matrices [GG17].

In kernel density estimation it is crucial to properly choose the bandwidth matrix. There is still ongoing research about the statistical estimation of a proper bandwidth matrix. A closed form estimation, suggested by Chacon and Duong [CD10], is defined as

$$\mathbf{H}_{ChD} = \left((2/3)\,n^{-1}\right)^{2/5} 2\boldsymbol{\Sigma}, \tag{1}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix. Using a closed form estimation is fast and results in an unconstrained matrix. The resulting density estimation with $\mathbf{H}_{ChD}$ tends to be over-smoothed, however.

In order to gain a more accurate bandwidth-matrix estimation, we followed the approach of the recently published work of Gramacki and Gramacki [GG17]. Their method aims to minimize a functional, called the Least Squares Cross Validation (LSCV) [ST87].

Since a closed form solution for the minimum of the LSCV is not known, a numerical minimizer is used to estimate $\mathbf{H}$. The authors pointed out that LSCV can also be efficiently approximated with a binning grid and the FFT. The authors avoided high-dimensional derivative computations and used a simplex method for optimization. In contrast to this, we used a better pilot matrix estimation (they used axis aligned matrices as pilots, unlike $\mathbf{H}_{ChD}$), and our experiments showed that in our case usually only few optimization steps were necessary. Therefore, we used the Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm for optimization, the gradients were approximated by a forward finite difference scheme.

Gramacki and Gramacki demonstrated their work on two-dimensional data. In our work we implemented the 3D version which, to the best of our knowledge, is the first 3D implementation of this recent method. We have also exploited the fact that the FFT, element-wise multiplication, and summation operations can be parallelized very efficiently on GPUs. Our C++/CUDA source code for 3D Kernel Density Estimation can be downloaded from `https://github.com/viktor-vad/KernelDensityEstimation3DCUDA.git`.

1.3. Performance Comparison

Despite the performance gain was not our primary goal, we conducted experiments to measure the computational time of KDE estimation for different grid sizes, and different number of samples. In each case, we measured the elapsed time for computing the binned grid, estimating the bandwidth matrix, estimating the density volume, estimating the percentile isovalues, and managing the GPU resources.

As it can be seen in Table 1, the elapsed time mainly depends on the grid size. The reason is that the binning part is highly linear, and it has been well parallelized, meanwhile the performance bottlenecks are the FFT computations, whose complexity depends solely on the grid size.

Table 1: *Performance Comparison of KDE in 3D. The values are in milliseconds.*

#samples	Grid size					
	Watergate			Gramacki and Gramacki		
	64^3	128^3	256^3	64^3	128^3	256^3
10,000	327	721	5,562	1,922	9,185	70,012
50,000	317	687	3,803	1,961	8,842	69,805
100,000	330	691	3,882	1,930	9,071	72,072
200,000	385	991	6,316	2,025	9,060	71,649
600,000	385	982	6,262	1,988	8,924	70,859
1,000,000	397	977	6,282	2,000	9,030	72,680
1,570,037	409	986	6,388	2,020	8,916	72,587

As we also wanted to compare the performance gain of our solution with the original one, the table contains also the performance measurement of the method by Gramacki and Gramacki. As already stated, the original version of this algorithm is demonstrated only for 2D case, therefore we extended it to 3D for performing comparisons.

All the tests were conducted on a configuration equipped with Intel Core i7-4790k (4GHz) CPU, 32 GB RAM and NVidia GTX 980 GPU (4GB GDDR).

References

[CD10] CHACÓN J. E., DUONG T.: Multivariate plug-in bandwidth selection with unconstrained pilot bandwidth matrices. *TEST 19*, 2 (2010), 375–398. 4

[GG17] GRAMACKI A., GRAMACKI J.: FFT-based fast bandwidth selector for multivariate kernel density estimation. *Comput. Stat. Data Anal. 106* (2017), 27 – 45. 4

[ST87] SCOTT D. W., TERRELL G. R.: Biased and unbiased cross-validation in density estimation. *J Am Stat Assoc 82*, 400 (1987), 1131–1146. 4

[Wan94] WAND M. P.: Fast computation of multivariate kernel estimators. *J. Comp. Graph. Stat. 3*, 4 (1994), 433–445. 3, 4

[Was10] WASSERMAN L.: *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, 2010. 3