Bioinformatics, 2024
Advance Access Publication Date: 17 December 2024
Structural Bioinformatics

Structural bioinformatics

ProteinRunway: A Blender-based UI for Molecular Dynamics Trajectory Visualization

Maria Krunic, Evgeniya Polezhaeva, Andrey Radev, Emma Rousseau

Faculty of Bioscience Engineering, MSc Bioinformatics, KU Leuven, Leuven 3000, Belgium.

Abstract

Motivation: Recent years have seen an avalanche of new tools for protein structure prediction, based on developments in machine learning and AI. They have led to new prediction methods and databases of predicted 3D structures. However, these databases often do not include information about the functional dynamics within proteins. In addition, the Blender environment provides a wide array of powerful tools to visualize 3D objects. There is great potential to compare and understand protein structures and their dynamics by presenting them in this graphical tool. Extracting the essential dynamics of proteins and offering a flexible visualization tool would bridge the gap between static representations and dynamic functional information.

Results: The ProteinRunway project consists of a Snakemake[1] workflow that is tuned for a specific database of proteins provided by Šoštarić N. *et al.*[2] and a Blender extension that can visualize the output of the workflow. The proteins' static topologies and Molecular dynamics (MD) trajectories undergo Normal Mode Analysis (NMA) to extract vibrational normal modes and create a simplified visualization of the major protein movements. Several segmentation algorithms are applied to determine functional domains, and the different clustering results are collected in a single Tab-Separated Value (TSV) file that can be attached to the Blender UI. Once rendered, the protein's normal modes can be visualized as an animation and the segmented domains can be wrapped in convex hulls to provide a cartoon-like object whose movements can be easily tracked.

The code that the workflow executes is organized in an object-oriented library that is intended as a starting point for a larger project. It has been designed to allow easy addition of new segmentation methods and to encapsulate the potentially complex processing into easy-to-use function invocations.

Availability: The entire code of the project is available at https://github.com/AndrewRadev/protein-runway, and the reference documentation can be accessed at https://andrewradev.github.io/protein-runway/.

Contact: maria.krunic@student.kuleuven.be, andrey.radev@student.kuleuven.be, evgeniya.polezhaeva@student.kuleuven.be, emma.rousseau1@student.kuleuven.be

Introduction

Proteins are an extremely diverse group of molecules, ranging from simple oligopeptides to aggregated complexes. While the sequence of amino acids that determines its composition is of primary importance, more and more research is being devoted to understanding protein function on a macroscopic level through its overall structure. For example, Foldseek is a recent computational algorithm that relies on querying protein structure databases by three-dimensional structure instead of amino acid sequence to discover similar proteins.[3] Furthermore, proteins, just like every other molecule, are inherently dynamic. A protein's structure becomes significantly more informative once molecular dynamics are added to the equation. While each atom vibrates due to Brownian motion, movement at the level of protein domains often visualizes the protein's overall function. Such movements can be approximated and studied using molecular dynamics simulations. Data from these simulations enable researchers to predict conformational changes, identify potential binding sites, and gain insights into mechanisms of enzymatic activity or protein-protein interactions.

Investigating protein molecular dynamics on a functional level requires understanding several layers of information. First, it is necessary to

determine how a protein is segmented as the primary movement that determines the functionality is a result of the tertiary and quaternary structure. There are several computational algorithms that can do this by relying on either static representations of proteins like Protein Data Bank (PDB) files, trajectories from the aforementioned molecular dynamics simulations, or true experimental data from NMR spectroscopy or X-ray crystallography. Given that protein movement on an atomic level can be quite complex, other computational analyses can also be integrated to allow for simplification into its principal components. Normal mode analysis (NMA) does this by utilizing small oscillation approximation to reduce protein movement into normal modes where vibrations of molecules are captured in the lower frequencies.[4] Once all this information is acquired, it can then be utilized for the final step of visualizing these results in a human-readable manner.

The most popular software for molecular visualization is PyMOL, which contains several tools relevant to biological investigation of molecular structure, particularly within the realm of proteins. However, the potential of Blender[5], a computer graphics software for 3D visualization, is growing within the scientific sphere. It is mostly known for 3D design in projects ranging from animated films to video games, but its extremely powerful interface and open-source design allow for even wider applications than originally intended. There are several Blender extensions that are already available for download and allow the user, for example, to fetch proteins by their PDB code and visualize them within the 3-dimensional scene. Apart from its extensive functionality for 3D rendering, Blender's support for third-party plug-ins also provides for increased flexibility and creativity.

This project aimed to integrate these three realms of 1) molecular dynamics data, 2) protein domain prediction, and 3) protein visualization into one workflow. As with all bioinformatics projects, this necessitated efficient code organization, robust testing, and clear documentation. These three different domains were connected using several tools, culminating in a Blender plug-in organized within an object-oriented framework to maximize extensibility and reusability for future development.

Methods

NMA was performed using ProDy[6], a powerful python library that writes normal modes to Normal Modes Data (NMD) files for visualization in the graphical program VMD[7]. For segmentation, we used three different algorithms. Both Chainsaw[8] and Merizo[9] are deep learning methods that try to predict domain assignment based on a static protein structure. Chainsaw uses a program called STRIDE[10] to assign secondary structure as a first step. We encountered challenges in using Merizo with our datasets due to its inability to recognize specific histidine residue codes (HIP, HID, HIE), which represent different protonation states, as well as the oxidized cysteine residue code CYX. To address this issue, we worked with our own fork of the project to implement the necessary modifications[11] and may contribute the fix to the main repository in the future.

GeoStaS is a method from the R library Bio3D[12], which attempts to determine domain segmentation based on an atomic movement correlation matrix.

The central workflow that processed all the data files was built using Snakemake and the final products were visualized with the open-source project Blender. The MDAnalysis[13,14] library is used in both the library code and in the blender extension to read and write various structural and trajectory files.

Results

The project consists of three separate domains:

- A data processing workflow that is tailored to the specific set of proteins we tested with.
- An object-oriented library that can be reused with a different workflow and extended to support different functionalities.
- A Blender extension that is independent from the processing code and only interacts with the rest of the project through the two final output files that the workflow creates.

Workflow

The data processing was organized using the Snakemake workflow management system to ensure reproducibility and scalability. The pipeline integrates several tools, including Merizo, Chainsaw and the Bio3D R package for protein segmentation analysis. It includes custom Python modules with intermediate visualization options in PyMOL, ensuring flexibility when more advanced tools like Blender are unavailable, and benchmarks all tasks for reflecting on their performance.

The process begins with setting up vendor tools such as Merizo and Chainsaw and quality assurance steps to verify their proper installation.

The input files, including protein topology and trajectory snapshots of MD simulations data, are organized in a specific directory structure, ensuring consistency in data management. Once the setup is complete, the pipeline generates static and dynamic PDB files based on the topology and trajectory snapshots data. Some of them include all protein atoms, while some are limited to only the alpha carbons. To accommodate Bio3D, DCD files are also built for the trajectories. These generated files form the foundation for further analysis.

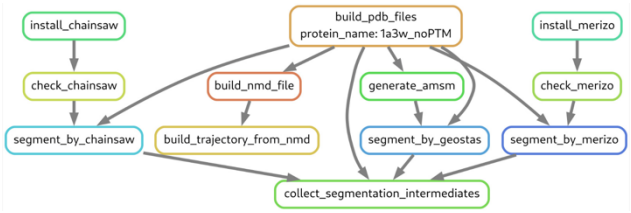

Figure 1. Simplified representation of the Snakemake workflow

Then, clustering is performed using three methods: Chainsaw, Merizo and GeoStaS (Bio3D). The latter requires picking a number of clusters in advance, since it segments molecules via K-means or hierarchical clustering. Each of these tools generate segmentation outputs, which are

compared and aggregated to produce a comprehensive view of the protein's dynamic regions. An example of the differences in output produced by the segmentation methods using the yeast protein 1A3W is included in Figure 2.

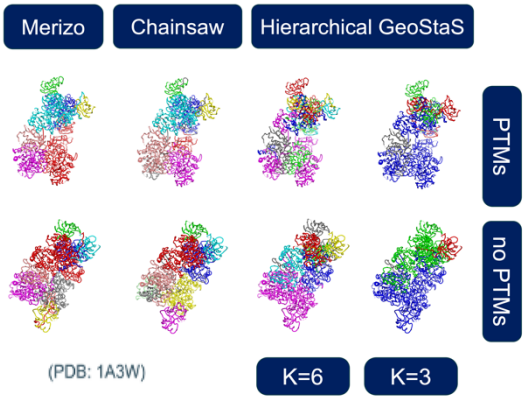

Figure 2. Results from three segmentation methods on the yeast protein 1A3W with and without post-translational modifications.

NMA is another critical component of the workflow where the main components of biomolecule dynamics are predicted. This step involves generating NMD files from dynamic PDB structures and producing visualizable trajectories compatible with tools like PyMOL. These trajectories provide insights into the protein's conformational flexibility, which are essential for understanding its functional dynamics.

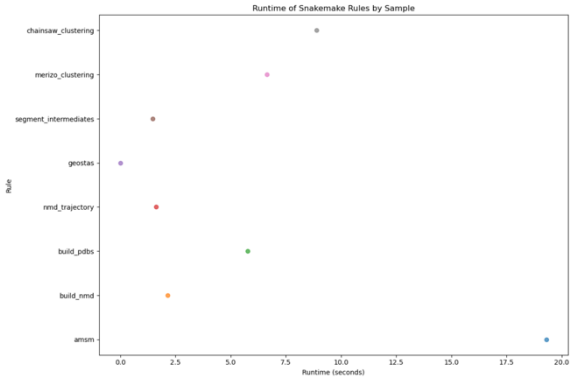

Figure 3. An example runtime measurement plot for the Snakemake workflow.

Throughout the workflow, benchmarks are collected to evaluate performance. Runtime data is aggregated into a summary file, and scatter plots visualize the distribution of runtimes across different pipeline rules (Figure 3). These measurements help identify bottlenecks and guide optimization efforts.

Library

In the first version of the workflow, all processing code was contained in procedural scripts. While straightforward, this made it inflexible and difficult to test in isolation. Therefore, the next step consisted of extracting the logic and organizing it into an object-oriented framework (Figure 4).

The generation of normal modes serves as a great example of what can be gained from this modular structure. ProDy's normal mode analysis output is specifically tailored for viewing in the VMD program. It does not easily interoperate with other kinds of software, so building a custom visualization required parsing that file and creating a trajectory by applying the normal vectors to the initial structure, frame by frame. To control the number of frames and the scale of the vectors, we implemented a *NormalModes* class that encapsulates this logic and provides various parameters to build an MDAnalysis trajectory in a flexible way. Future extensions and iteration were therefore made easier to implement, since changing the shape of the visualization is as easy as changing the specifics of a function invocation. The act of calculating modes in a separate procedure respects the single responsibility principle[15]: the class handles parsing the NMA output to generate a trajectory, while the procedure invokes ProDy to build the initial NMD file.

Another important target for extraction was the segmentation process. Much of the work is done by external tools and is procedural in nature. In addition to segmentation, it was necessary to establish a standardized protocol for communicating the results, which requires the implementation of a set of adaptor interfaces. A segmentation base class was designed with a simple constructor and a "parse" method with a specific interface. A function can be given a list of *SegmentationParser* objects, and it produces the final segmentation TSV with the results, ready to be uploaded to the Blender extension. This lets users of the library program to an interface, not to an implementation; what they do in their own parser classes does not interfere with the collection process.

The last main objective was to prevent abstraction leakage of implementation details across the project. MDAnalysis in particular was referred to in many different places, since we regularly need to read or write structures and trajectories. Collecting it all in a *Trajectory* class allowed us to encapsulate those details and choose our own interface for trajectory manipulation, reading, and writing. Moreover, it made testing simpler, since a *Trajectory* object was much easier to create and use as input to other classes and methods. If the test suite increases in size, using a mock *Trajectory* object for unit tests can greatly increase the speed of test execution.

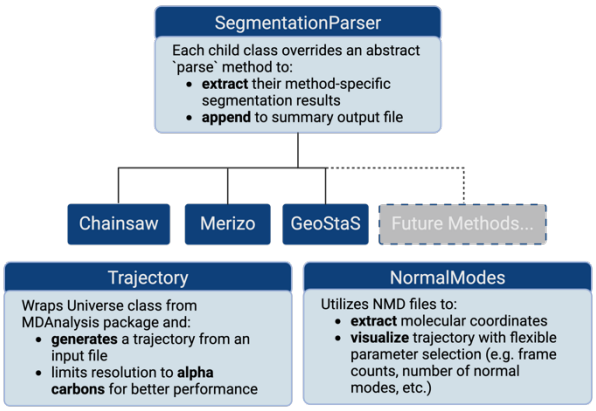

Figure 4. Structure of the object-oriented framework.

Blender extension

Once a simplified trajectory and a segmentation file are created, the last step is to visualize them in the graphical user interface, although there

are some technical challenges to this. Blender is a multi-purpose program meant to work with potentially very complex models, so it is important to "speak its language" to achieve acceptable performance. Generating spheres for each individual atom can freeze the entire user interface for up to a minute, therefore it is impractical to render thousands of separate objects without some computational preparation.

The Atomic Blender[16] project was the inspiration for how to handle this issue. The idea is to create a single representative sphere for all atoms of a molecule with a particular material. A material represents visual properties like color, texture, and reflectivity. Blender can use this object as a template and mirror its visuals in all other desired locations instead of processing them individually.

Thanks to this performance shortcut, creating the static structure was almost instantaneous, but this only created a static image. The process of generating the animation consisted of looping through the frames of the trajectory, updating the coordinates of the atoms and using snapshots of these as numbered "frames". Once again, the naive approach was very slow, but no clever shortcuts were available to specifically address this. To circumvent this, the logic was placed in a "modal" operator that only inserts one frame per invocation and then yields control to the user interface. A timer was set up to check at sub-second intervals whether the job was finished or could be invoked again. Additionally, to give the users feedback, a progress bar fills up as the animation frames are inserted into the scene. The feedback makes waiting more palatable, and the UI is available during this period. The import action in progress can be seen in Figure 5.

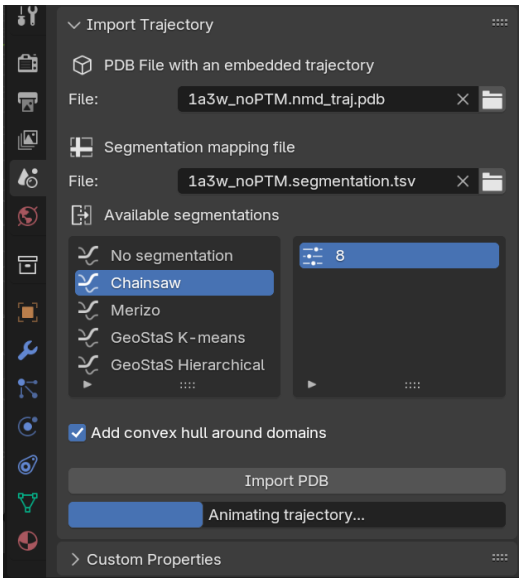

Figure 5. The import panel in the Blender UI, in the process of inserting animation frames for a provided trajectory.

This panel is the main interface of the plugin. A user adds both the trajectory and the segmentation file, and once the latter is processed, a list of methods and parameterizations of the segmentation algorithm is displayed. Segmentation colors each fragment in different hues, making it easy to distinguish the functional domains.

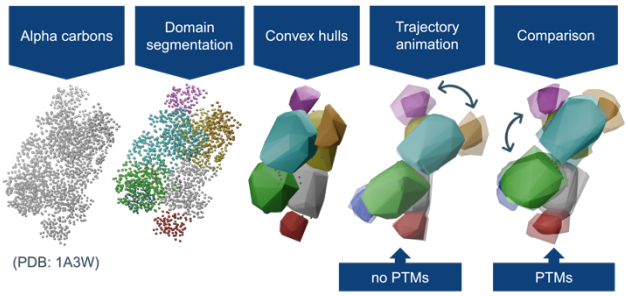

Figure 6. Illustration of the computational and visualization steps leading to the final form generated by Blender.

The final convenience was to add a convex hull around the individual domains using Blender's built-in "convex hull" action. The steps leading up to this visualization are illustrated in Figure 6. It is useful to recognize that the result is not necessarily an ideal shape. In Figure 7, we can see the yeast protein 1A3W segmented by the Chainsaw algorithm on the left, and by GeoStaS K-means (K=8) on the right. While Chainsaw draws roughly spherical domains, the ones by GeoStaS are bulkier and hide some of the internal structure of the "pincers" at the top of the protein. This could be due to GeoStaS attempting to capture rotational movements by considering correlated atomic trajectories in the opposite direction as part of a rigid domain that rotates. Nevertheless, if those two groups of atoms are clustered together, a convex hull that wraps them hides much of the structural details. An ideal representation would have a surface clinging tightly to the outside of the two protrusions, but what is perceived as a shape greatly depends on the visual density of the cloud of points. Topological analysis could potentially be an appropriate method to create a better wrapping surface for this domain.

In order to package the extension, its dependencies must also be included. GitHub Actions were used to set up a pipeline that collects all Python dependencies into archives known as "wheels" and packages them with the extension code, ready for installation. This process supports the major operating systems: Linux, Windows, and macOS for the x86 64-bit architecture. Additionally, a separate build was created for macOS on ARM x64 due to the platform's transition towards ARM processors. While manual creation of the Blender extension is feasible, automating the process to generate these archives and provide them as easy downloads offers a more user-friendly solution.

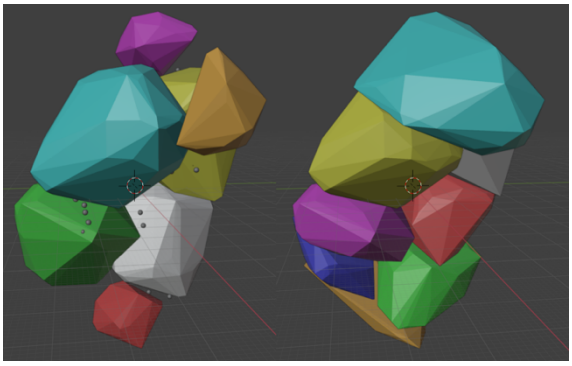

Figure 7. A pyruvate kinase (1A3W) segmented by Chainsaw (left) and by the GeoStaS algorithm with a k-means clustering with K=8 (right).

Discussion

Blender is a powerful graphics tool, but that power comes at the cost of complexity. Our extension is simple and focused on doing one task, but no other Blender plugin we found combines these particular features. The aforementioned Atomic Blender does a fantastic job of highlighting and coloring individual atoms but is unable to render trajectories. MolecularNodes[17] is another mature extension we took inspiration from. It provides a wide variety of rendering tools to make protein models and animations for professional illustrations. However, it lacks segmentation and normal mode analysis functionalities. Using specialized software like PyMOL is a valid option, but one misses out on the general-purpose abilities of a graphics software like Blender. There is no one tool that does a perfect job.

What we can aim for is to establish a solid baseline for future work. We already pointed out the limitations of GeoStaS in particular, but none of the segmentation algorithms fully solve the domain separation problem. The main goal was to ensure that adding new methods was straightforward, rather than selecting a "best" method. More flexibility in deciding which algorithms to apply for specific proteins might be welcome. An idea for a future extension is to prepare a “guide” file with algorithm settings, allowing the user to configure the pipeline on a per-model basis. Ideally, the Blender extension itself could provide a panel to modify segmentations or parameterization that can be written back to the pipeline. This would increase the coupling between these two major components of the toolkit, but it might speed up interacting with the project.

More configuration of the visuals would likely be welcome for users that are looking for illustrations they can share in presentations or papers. There is only one set of colors that can easily be picked directly and no control over the radius and visibility of the atomic spheres. These kinds of visual augmentations should be straightforward to add, but would take time, and would require feedback from the users of the project. Personal preferences regarding interfaces and convenience will vary, so ongoing refinement would be better than upfront decisions.

Acknowledgements

We would like to thank Prof. Vera van Noort for providing the project idea and for offering valuable feedback and suggestions throughout our work. We also want to thank Stefaan Verwimp and Dr. Michiel van Setten for their additional guidance in the project, along with Dr. Nikolina Šoštarić for providing the data.

Conflict of Interest: none declared.

References

[1] Mölder F, Jablonski KP, Letcher B *et al.* Sustainable data analysis with Snakemake. *F1000Research* 2021, **10**:33. doi: 10.12688/f1000research.29032.1

[2] Sostaric, N. and van Noort, V. (2021) “Molecular dynamics shows complex interplay and long-range effects of post-translational modifications in yeast protein interactions”. Zenodo. doi: 10.5281/zenodo.4650406.

[3] van Kempen, M., Kim, S.S., Tumescheit, C. et al. Fast and accurate protein structure search with Foldseek. Nat Biotechnol 42, 243–246 (2024). https://doi.org/10.1038/s41587-023-01773-0

[4] Bauer JA, Pavlović J, Bauerová-Hlinková V. Normal Mode Analysis as a Routine Part of a Structural Investigation. Molecules. 2019 Sep 10;24(18):3293. doi: 10.3390/molecules24183293. PMID: 31510014; PMCID: PMC6767145.

[5] Community, B. O. (2018). *Blender - a 3D modelling and rendering package*. Stichting Blender Foundation, Amsterdam. Retrieved from http://www.blender.org

[6] She Zhang, James M Krieger, Yan Zhang, Cihan Kaya, Burak Kaynak, Karolina Mikulska-Ruminska, Pemra Doruker, Hongchun Li, Ivet Bahar, *ProDy* 2.0: increased scale and scope after 10 years of protein dynamics modelling with Python, *Bioinformatics*, V. 37, Is. 20, October 2021, Pages 3657–3659, doi: 10.1093/bioinformatics/btab187

[7] Humphrey, W., Dalke, A. and Schulten, K., "VMD - Visual Molecular Dynamics", J. Molec. Graphics, 1996, vol. 14, pp. 33-38., http://www.ks.uiuc.edu/Research/vmd/

[8] Jude Wells, Alex Hawkins-Hooker, Nicola Bordin, Ian Sillitoe, Brooks Paige, Christine Orengo, Chainsaw: protein domain segmentation with fully convolutional neural networks, *Bioinformatics*, Volume 40, Issue 5, May 2024, btae296, doi: 10.1093/bioinformatics/btae296

[9] Lau, A.M., Kandathil, S.M. & Jones, D.T. Merizo: a rapid and accurate protein domain segmentation method using invariant point attention. *Nat Commun* **14**, 8445 (2023). doi: 10.1038/s41467-023-43934-4

[10] Frishman D, Argos P. Knowledge-Based Protein Secondary Structure Assignment Proteins: Structure, Function, and Genetics 23:566-579 (1995)

[11] https://github.com/AndrewRadev/Merizo

[12]Grant BJ, Skjaerven L, Yao XQ. The Bio3D packages for structural bioinformatics. Protein Sci. 2021 Jan;30(1):20-30. doi: 10.1002/pro.3923. Epub 2020 Aug 17. PMID: 32734663; PMCID: PMC7737766.

[13] R. J. Gowers, M. Linke, J. Barnoud, T. J. E. Reddy, M. N. Melo, S. L. Seyler, D. L. Dotson, J. Domanski, S. Buchoux, I. M. Kenney, and O. Beckstein. MDAnalysis: A Python package for the rapid analysis of molecular dynamics simulations. In S. Benthall and S. Rostrup, editors, *Proceedings of the 15th Python in Science Conference*, pages 98-105, Austin, TX, 2016. SciPy, doi:10.25080/majora-629e541a-00e

[14] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein. MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations. *J. Comput. Chem.* **32** (2011), 2319-2327, doi:10.1002/jcc.21787. PMCID:PMC3144279

[15] Martin, Robert C. (2003). Agile Software Development, Principles, Patterns, and Practices. Prentice Hall. p. 95. ISBN 978-0135974445.

[16] https://extensions.blender.org/add-ons/atomic-blender-pdb-xyz/

[17] Brady Johnston, Johannes Elferich, Russell B. Davidson, Yuxuan Zhuang, Yinying Yao, Thibault Tubiana, Patrick Kunzmann, Rich, Olivier Laprevote, TheJeran, ludovic autin, JCZwiggelaar, Domenico Marson, Kai Niklas Spauszus, Brener Ramos, James Hooker, Jessica A. Nash, Joyce Kim, Louis Colson, Marcelo C. R. Melo. (2024). BradyAJohnston/MolecularNodes: v4.2.9 for Blender 4.2+ (v4.2.9). Zenodo. doi: 10.5281/zenodo.14241983