3-dimensional Protein Secondary Structure Alignment Principle and Application

Maulahna Robinson

CS723 Bioinformatics

Jing He

April 28th, 2025

Secondary structure alignment focuses on identifying similarities in the local folding patterns of proteins. These folding patterns form structures such as alpha-helices, beta-sheets, and others. A mathematical approach can be used to align vectors in a 3D space. To rotate v1 and v2 requires the axis angle representation, with the formula $v_1 = \frac{v_1}{||v1||}$, $v_2 = \frac{v_2}{||v_2||}$. With this formula, the vectors will now face the same direction. When aligning two 3D objects that each consist of multiple corresponding points, two main methods are often used.

The first method used for alignment is the Kabsch Algorithm. The purpose of the Kabsch Algorithm is to find the optimal rotation and translation that minimizes the root-mean-square deviation (RMSD) between two sets of points. This algorithm can be broken down into five steps.

1.) Center the point sets by subtracting the centroids.
2.) Compute the covariance matrix between corresponding points.
3.) Perform Singular Value Decomposition (SVD) on the covariance matrix.
4.) Use the SVD components to compute the optimal rotation matrix.
5.) Apply rotation and translation to align the point sets.

The second method used for alignment is the Iterative Closest Point Algorithm (ICP). The purpose of this algorithm is to align 3D points that may not have known correspondences. This algorithm can also be broken down into five steps:

1.) Initialize a guess for the transformation.

2.) For each point in Set A, find the closest point in Set B.

3.) Compute a transformation (rotation and translation) that minimizes the distance.

4.) Apply the transformation.

5.) Repeat until convergence.

Compared to the Kabsch Algorithm, ICP is broader and more general since it does not require one-to-one correspondence between points. A few programs can make the alignment process easier, including PyMOL, BioPython, and Open3D.

For the alignment program, the program is supposed to allow the users to input two PDB files containing information about the two helices. The program extracts the points from the PDB file and begins to calculate the transformations required for alignment, translation, and rotation. Translate the second helix axis so it matches the first helix axis. After the translation, rotate the second matrix so it also matches the orientation of the first axis. After the helices match, the transformed coordinates can be saved to a PDB file. For convenience, a sample of the program output is provided below, displaying the first ten lines of the alignment.

```
ATOM      1  CA  ASN A    1     147.833 146.632 204.730
ATOM      2  CA  ASN A    2     147.856 146.616 204.690
ATOM      3  CA  ASN A    3     147.882 146.598 204.640
ATOM      4  CA  ASN A    4     147.909 146.579 204.580
ATOM      5  CA  ASN A    5     147.939 146.558 204.530
ATOM      6  CA  ASN A    6     147.971 146.536 204.470
ATOM      7  CA  ASN A    7     148.004 146.512 204.400
ATOM      8  CA  ASN A    8     148.040 146.487 204.330
ATOM      9  CA  ASN A    9     148.077 146.461 204.260
ATOM     10  CA  ASN A   10     148.116 146.434 204.190
```

To run the program, clone the repository from GitHub to your local machine. In the command line, use the command “python main.py” to run the program. It will output results in a file called “aligned_helix2.pdb”. If the output file has not been created, check to see if it already exists. If it already exists, delete the file and run the program again. For the program to run smoothly, the output file will be removed from the GitHub repository.

GitHub Repository:
https://github.com/Maulaera/CS723-Project.git