

α -Surface and Its Application to Mining Protein Data

Xiong Wang

Department of Computer Science
California State University, Fullerton
Fullerton, CA 92834-6870, USA
wang@ecs.fullerton.edu

Abstract

Given a finite set of points in three dimensional Euclidean space R^3 , the subset that forms its surface could be different when observed in different levels of details. In this paper, we introduce a notion called α -surface. We present an algorithm that extracts the α -surface from a finite set of points in R^3 . We apply the algorithm to extracting the α -surfaces of proteins and discover patterns from these surface structures, using the pattern discovery algorithm we developed earlier. We then use these patterns to classify the proteins. Experimental results show the good performance of the proposed approach.

1. Introduction

Protein classification is a very important research topic with deep implications [1, 2, 3]. Traditionally, proteins are classified according to their functions. However, recently, many approaches have been proposed to classify proteins according to their structures, e.g. sequences [3], secondary structures [3], and three dimensional structures [6]. In [5, 6], we developed an algorithm that discovers frequently occurring patterns in a set of 3D graphs. We applied the algorithm to protein classification. While we succeeded in classifying two families of proteins with high recall and precision, experimental results showed that it was difficult to extend the approach to classifying more than two families of proteins. One reason is that proteins are large molecules, typically with several hundreds or even thousands of atoms. Many of the substructures that occur frequently in multiple proteins are not specifically related to their functions.

Significant studies have shown that the structure of the surface of a protein relates more to the function of the protein. In this paper, we define α -surface of a finite set of points in R^3 and present an algorithm for extracting α -surfaces from finite point sets. We apply the algorithm to extracting α -surfaces of proteins. We then employ the

pattern discovery algorithm that we developed earlier to find frequently occurring patterns on the α -surfaces and use these patterns to classify the proteins. The rest of the paper is organized as follows: In Section 2, we define α -surface. In Section 3, we describe the surface extracting algorithm. Section 4 discusses how the surface extracting algorithm and the pattern discovery algorithm are applied to protein classification. Section 5 presents some experimental results. We conclude the paper in Section 6.

2. α -Surfaces

Definition 2.1 Given a point O in R^3 and a real number α ($0 < \alpha < \infty$), an α -ball is the set of points $B(O, \alpha) = \{P | P \in R^3 \text{ and } ||P - O|| < \alpha\}$, where $||P - O||$ is the Euclidean distance between P and O . A closed α -ball $\overline{B}(O, \alpha)$ is the α -ball $B(O, \alpha)$ plus its bounding sphere, i.e. $\overline{B}(O, \alpha) = \{P | P \in R^3 \text{ and } ||P - O|| \leq \alpha\}$.

Definition 2.2 Given a finite set \mathcal{D} of points in R^3 and a real number α ($0 < \alpha < \infty$), the α -surface \mathcal{S} of \mathcal{D} is defined as $\mathcal{S} = \{P | P \in \mathcal{D} \text{ and } (\exists O \in R^3 \text{ such that } B(O, \alpha) \cap \mathcal{D} = \emptyset \text{ and } P \in \overline{B}(O, \alpha))\}$. When $B(O, \alpha) \cap \mathcal{D} = \emptyset$ and $P \in \overline{B}(O, \alpha) \cap \mathcal{D}$, we say that α -ball $B(O, \alpha)$ touches P . $P \in \mathcal{S}$ is called a surface point with respect to α (simply a surface point when the context is clear).

Fig. 1 illustrates the notion in R^2 .

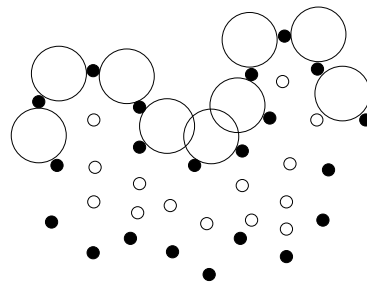


Figure 1. An α -surface in R^2 .

The definition of α -surfaces is general. In the context of mining protein data, we need some adjustment. First of all, the surface of a protein is important to its function, because a protein reacts to its surrounding through its surface. Thus we are not concerned with those parts of α -surfaces that are not *visible*, namely those surface points that are enclosed inside the proteins. Secondly, when α is small, the α -surface of \mathcal{D} could be split to two pieces. A protein is one molecule. Its surface should be in one piece. We specify the adjustment in the following definition.

Definition 2.3 Let α ($0 < \alpha < \infty$) be a real number and \mathcal{S} be the α -surface of a finite set \mathcal{D} . \mathcal{S} is connected, if for any two surface points $P_1, P_2 \in \mathcal{S}$ there are a finite number of α -balls: $B(O_1, \alpha), B(O_2, \alpha), \dots, B(O_n, \alpha)$, such that:

- (i) $B(O_i, \alpha) \cap \mathcal{D} = \emptyset$ ($1 \leq i \leq n$).
- (ii) $\overline{B(O_i, \alpha)} \cap \overline{B(O_{i+1}, \alpha)} \cap \mathcal{S} \neq \emptyset$ ($1 \leq i \leq n - 1$).
- (iii) $P_1 \in \overline{B(O_1, \alpha)}$ and $P_2 \in \overline{B(O_n, \alpha)}$.

Notice that, (ii) requires two contiguous α -balls to touch at least one common surface point. Imagine that the α -ball is solid, so are the points in \mathcal{D} , and we roll the α -ball along the surface of \mathcal{D} . Intuitively, if an α -surface is connected, we can roll an α -ball from one surface point to another along the surface.

3. Extracting α -Surfaces

Starting from the point P_m with the maximum X -coordinate in \mathcal{D} , the surface extracting algorithm rolls the α -ball to any surface point that can be touched in a breadth first manner. Obviously, P_m is a surface point with respect to any α . Since the radius of the α -ball is fixed, the position of the α -ball is determined uniquely by the center. Thus rolling the α -ball means that given the current center of the α -ball O determine a new center O' , such that the α -ball touches the next surface point. In R^2 , this is an easy task, since there are only two directions to roll the α -ball. In R^3 , however, determining the new center O' becomes a complicate problem. The difficulty comes from choosing the rolling direction. If we fixed the α -ball at one surface point, there are infinite directions in which we can roll the α -ball. We discuss three primitive ways of rolling the α -ball from the current position to the next surface point.

Case 1. Rolling from P_m : The first case occurs only once at the beginning, i.e. when the α -ball touches a single point P_m . Notice that the next surface point is within distance 2α of P_m (see Fig. 2). For each neighboring point P , we utilize the plane that is determined by O, P_m , and P to fix the rolling direction and pick up the next surface point to be the point that the α -ball can touch by rolling the smallest angle, e.g. P_1 in Fig. 2.

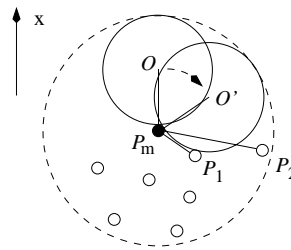


Figure 2. Rolling from P_m .

Case 2. Rolling side by side: In the second case, the α -ball touches at least two surface points. To determine the rolling direction, we pick up two of these points to be the current surface points. If we fixed the α -ball at both surface points, there are only two directions to roll the α -ball. In Fig. 3, suppose the α -ball currently touches P_0 and P_1 , and P_c is the point in middle of P_0 and P_1 . The next surface point is the point that the α -ball can touch by rolling the smallest angle from the plane determined by P_0, P_1 , and O , namely $\angle OP_c O'$ is smallest.

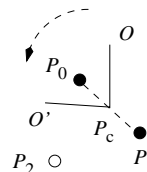


Figure 3. Rolling side by side.

Case 3. Rolling away from one end: Rolling side by side does not guarantee to find all surface points that are within the neighbor of the current surface point, e.g. when the neighboring surface point is collinear with the two current surface points. In such cases we need to roll the α -ball away from one of the two current surface points. This is similar to the first case (Rolling from P_m).

The surface extracting algorithm maintains a queue \mathcal{Q} which holds a subset of the α -surface \mathcal{S} that are under expansion. The basic rolling procedure of the algorithm rolls the α -ball around one surface point in \mathcal{Q} , so that all its neighboring points in \mathcal{S} will be touched at least once by the α -ball. These neighbors are added to \mathcal{Q} . Fig. 4 illustrates the procedure.

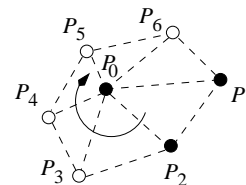


Figure 4. Rolling around one point.

Since the neighboring surface points are within distance 2α of the current surface point, to speed up the process, we partition \mathcal{D} at the very beginning. Let x_{min} (x_{max}) be the minimum (maximum) X coordinate of all the points in \mathcal{D} , respectively. Let x_0, x_1, \dots, x_n be defined as the following:

(i) $x_0 = x_{min}$,

(ii) $x_{i+1} = x_i + 2\alpha$ ($0 \leq i \leq n - 1$), and

(iii) $x_{n-1} \leq x_{max} < x_n$.

We cut the range $[x_{min}, x_{max}]$ to segments $[x_i, x_{i+1}]$ ($0 \leq i \leq n - 1$) with length 2α . Similarly, let y_{min} (y_{max}) be the minimum (maximum) Y coordinate and z_{min} (z_{max}) be the minimum (maximum) Z coordinate, respectively. We cut the ranges $[y_{min}, y_{max}]$ and $[z_{min}, z_{max}]$ to segments with length 2α . Each partition $Pt_{i,j,k}$ is a cube $Pt_{i,j,k} = \{(x, y, z) | x_i \leq x < x_{i+1}, y_j \leq y < y_{j+1}, \text{ and } z_k \leq z < z_{k+1}\}$. For any given point $P = (x, y, z) \in \mathcal{D}$, let $i = \lfloor \frac{x - x_{min}}{2\alpha} \rfloor$, $j = \lfloor \frac{y - y_{min}}{2\alpha} \rfloor$, and $k = \lfloor \frac{z - z_{min}}{2\alpha} \rfloor$. P belongs to partition $Pt_{i,j,k}$ and the points that are within distance 2α of P are all located in the 27 partitions surrounding $Pt_{i,j,k}$.

Assuming that the points in \mathcal{D} are evenly distributed, the complexity of the surface extracting algorithm is $O(\frac{|\mathcal{D}|^2}{N})$, where $|\mathcal{D}|$ is the size of \mathcal{D} and $N = \lceil \frac{x_{max} - x_{min}}{2\alpha} \rceil \times \lceil \frac{y_{max} - y_{min}}{2\alpha} \rceil \times \lceil \frac{z_{max} - z_{min}}{2\alpha} \rceil$ is the total number of partitions.

4. Classifying Proteins

To evaluate the performance of the surface extracting algorithm, we applied it to classifying three families of proteins. We classified the proteins using the 10-way cross-validation scheme. That is, each family was divided into 10 groups of roughly equal size. Ten tests were conducted. In each test, a group was taken from a family and used as test data; the other nine groups were used as training data for that family. We first utilized the surface extracting algorithm to find the surfaces of the proteins in the training data. We then employed the pattern discovery algorithm we developed before to find frequently occurring patterns from these surfaces. Finally, we used these patterns to classify the proteins in the test data.

The pattern discovery algorithm proceeds in two phases to search for patterns on the surfaces. In the first phase, the algorithm decomposes the surfaces to substructures and hashes the substructures to a three dimensional hash table. Let \mathcal{S} be a surface outputted by the surface extracting algorithm. For any point $P \in \mathcal{S}$, we consider P and its k -nearest neighbors in \mathcal{S} as a substructure and attach a local coordinate system SF to P . We hash the node-triplets from the substructure to a three dimensional hash table. The hash bin address is determined by the lengths of the three edges

of the triangle formed by the triplet. The three edges are sorted ascendantly on their lengths. The basic idea is that if two triangles match each other the longest edge of a triangle must match the longest edge of the other triangle. Likewise, the shortest edge of a triangle must match the shortest edge of the other. Stored in the hash bin are a protein identification number, a substructure number, and the local coordinate system SF . By the end of the first phase, all surfaces are stored in the hash table. In the second phase, the algorithm considers each substructure as a candidate pattern and rehashes it to evaluate its number of occurrences in the training data. In this phase, we again take each node-triplet from the candidate pattern and utilize the lengths of the sorted three edges to access the hash table. All the triplets that were stored in the accessed hash bin are recognized as matches and their local coordinate systems SF are recovered based on the global coordinate system that defines the candidate pattern. The triplet matches are augmented to larger substructure matches if they come from the same substructure and their recovered local coordinate systems match each other. A candidate pattern M occurs on the surface of a protein if M matches any substructure from the surface within one mutation¹.

For each candidate pattern, let n_i ($i = 1, 2, 3$) be its occurrence numbers in the training data of family i . We exclude those candidate patterns such that $n_1 = n_2 = n_3$. All the other candidate patterns are collected as useful patterns. Each pattern is associated with a weight pro^i where

$$pro^i = \frac{n_i}{\min\{n_1, n_2, n_3\} + 1}$$

We add a denominator to the weight because we observed that some patterns are common to proteins from different families. Although they may still occur more frequently in some family, they really are not specific to any family.

When classifying a test protein Q , we first find its surface using the proposed surface extracting algorithm and construct substructures as described above. Let M_1, M_2, \dots, M_p be all the patterns found in the training data and $pro_1^i, pro_2^i, \dots, pro_p^i$ be their corresponding weights for family i . Namely,

$$pro_j^i = \frac{n_{ji}}{\min\{n_{j1}, n_{j2}, n_{j3}\} + 1}$$

where n_{ji} is the occurrence number of pattern M_j ($j = 1, 2, \dots, p$) in the training data of family i . Family i obtains a score

$$\mathcal{N}^i = \frac{\sum_{j=1}^p d_j \times pro_j^i}{\sum_{j=1}^p pro_j^i}$$

¹A candidate pattern M matches a substructure S with n mutations if by applying an arbitrary number of rotations and translations as well as n node insert/delete operations to M , one can transform M to S (see [5, 6] for details).

where

$$d_j = \begin{cases} 1 & \text{if } M_j \text{ occurs in } Q \\ 0 & \text{otherwise} \end{cases}$$

The protein Q is classified to the family i with maximum \mathcal{N} . We add the denominator to make the score fair to all families. Notice that the maximum possible score for any family is 1. This is necessary because the three families do not have the same number of representatives in the training data. Furthermore, the sizes of the proteins are very different. If we can not decide a winner, then the “no-opinion” verdict is given.

5. Experimental Results

We have implemented the surface extracting algorithm using C++ on a Sun Ultra 10 workstation running Solaris 8 operating system. We selected three families of proteins from SCOP [2]. SCOP is accessible at <http://scop.mrc-lmb.cam.ac.uk/scop/>. The three families pertain to Transmembrane Helical Fragments, Matrix Metalloproteases – catalytic domain, and Immunoglobulin – I set domains. In determining the structure of a protein, we consider only the C_α , C_β and N atoms. These atoms form the polypeptide chain backbone of a protein where the polypeptide chain is made up of residues linked together by peptide bonds. We classified the proteins as discussed in Section 4. When adjusting α in the surface extracting algorithm, we found that $\alpha = 7.5$ yielded the best result. When constructing substructures (patterns), we found the substructures with 6 points yielded the best result. In each of these substructures, there was a surface point together with its 5 nearest neighbors on the α -surface. The algorithm produced a set of surface points that were on average 25% of the size of a protein.

We use recall (\mathcal{R}) and precision (\mathcal{P}) to evaluate the effectiveness of our classification algorithm. Recall is defined as

$$\mathcal{R} = \frac{\mathcal{T} - \sum_{i=1}^3 \mathcal{M}^i}{\mathcal{T}} \times 100\%$$

where \mathcal{T} is the total number of test proteins and \mathcal{M}^i is the number of test proteins that belong to family i but are not assigned to family i by our algorithm (they are either assigned to family j , $j \neq i$, or they receive the “no-opinion” verdict). Precision is defined as

$$\mathcal{P} = \frac{\mathcal{T} - \sum_{i=1}^3 \mathcal{M}^i}{\mathcal{T} - \mathcal{V}} \times 100\%$$

where \mathcal{V} is the number of test proteins that receive the “no-opinion” verdict. With the 10-way cross validation scheme, the average \mathcal{R} over the ten tests was 93.7% and the average \mathcal{P} was 95.9%. It was found that 2.3% test proteins on average received the “no-opinion” verdict during the classification. In our previous work [6], we also tried to classify three

families of proteins and the recall and precision dropped to 80%. The results reported here are much better.

6. Conclusions

We have given a formal definition of surface points of a finite point set in R^3 and presented an algorithm for extracting such surface points. We applied this algorithm, together with previously developed algorithms for 3D pattern discovery, to protein classification. In [4], we reported a preliminary result of this research without discussing the surface extracting algorithm. We also improve the classification approach in this paper, thanks to the comments from the reviewers. For our future work, we will conduct comprehensive experiments on more protein families to find interesting patterns on their surfaces. We will also extend our algorithm to applications in three dimensional visualization.

7. Acknowledgments

The author thanks Dr. Jason T. L. Wang for his comments and Dr. Jane Richardson for helpful discussion during the Atlantic Symposium on Computational Biology, Genome Information Systems & Technology. The author also appreciates the comments from the anonymous reviewers.

References

- [1] R. King, A. Karwath, A. Clare, and L. Dephaspe. Genome scale prediction of protein functional class from sequence using data mining. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 384–389, Boston, MA, 2000.
- [2] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247(4):536–540, 1995.
- [3] J. T. L. Wang, B. A. Shapiro, and D. Shasha. *Pattern Discovery in Biomolecular Data: Tools, Techniques and Applications*. Oxford University Press, New York, NY, 1999.
- [4] X. Wang. Mining protein surfaces. In *2001 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 20–24, Santa Barbara, CA, 2001.
- [5] X. Wang, J. T. L. Wang, D. Shasha, B. A. Shapiro, S. Dikshitulu, I. Rigoutsos, and K. Zhang. Automated discovery of active motifs in three dimensional molecules. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 89–95, Newport Beach, CA, 1997.
- [6] X. Wang, J. T. L. Wang, D. Shasha, B. A. Shapiro, I. Rigoutsos, and K. Zhang. Finding patterns in three dimensional graphs: Algorithms and applications to scientific data mining. *IEEE Transactions on Knowledge and Data Engineering*, To appear.