2006－AL－105（3）
2006／3／17

幾何的接尾辞木：タンパク質３次元構造検索のための新しい索引構造

渋谷 哲朗

東京大学医科学研究所ヒトゲノム解析センター
〒108-8639 東京都港区白金台 4-6-1
E-mail: tshibuya@hgc.jp

概要: タンパク質構造解析はポストゲノム時代の非常に重要な研究分野であり、非常に多くのタンパク質構造が次々と解明されるに従いより高速でより正確な３次元構造検索のための索引構造を設計することが急務となっている。本論文ではタンパク質３次元構造を検索するための幾何的接尾辞木と名づけた新しい索引構造を提案する。このデータ構造を用いることで、クエリーとする３次元構造に対して、RMSD (root mean square deviation) あるいは URMSD (unit-vector root mean square deviation) とよばれる指標による指定値以上に類似するデータベース中のタンパク質の３次元構造の部分構造を効率的にすべて列挙することができる。データベースのサイズを n とするとそれに含まれる部分構造は $O(n^2)$ 存在するにも関わらず、幾何的接尾辞木を記憶するのに必要なメモリサイズは $O(n)$ である。また、このデータ構造はナイーブに構築すると $O(n^3)$ の計算が必要であるが、本論文では $O(n^2)$ の構築アルゴリズムを提案する。さらに効率的な検索アルゴリズムも提案する。最後に簡単な計算機実験を通じてこのデータ構造の有用性を検証する。

Geometric Suffix Tree: A New Index Structure for Protein 3-D Structures

Tetsuo Shibuya

Human Genome Center, Institute of Medical Science, University of Tokyo
4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan.
E-mail: tshibuya@hgc.jp

Abstract: Protein structure analysis is one of the most important research issues in the post-genomic era, and faster and more accurate query data structures for such 3-D structures are highly desired for research on proteins. This paper proposes a new data structure called the geometric suffix tree to index protein 3-D structures. By using the geometric suffix tree for a set of protein structures, we can search for all of their substructures whose RMSDs (root mean square deviations) or URMSDs (unit-vector root mean square deviations) to a given query 3-D structure are not larger than a given bound. Our data structure requires $O(n)$ space where n is the sum of lengths of the set of proteins, though there are $O(n^2)$ substructures. We propose an $O(n^2)$ construction algorithm for it, while a naive algorithm would require $O(n^3)$ time to construct it. Moreover we propose an efficient search algorithm. We also show computational experiments to demonstrate the practicality of our data structure. The experiments show that the construction time of the geometric suffix tree is almost linear to the size of the database in practice, when applied to a protein structure database.

1 Introduction

Analyzing 3-D structures of proteins is very important in molecular biology and more and more protein structures are solved today with the aid of state-of-the-art technologies such as nuclear magnetic resonance (NMR) techniques, as seen in the increasing number of PDB [4] entries: 34,626 on January 17, 2006. It is said that structurally similar proteins tend to have similar functions even if their amino acid sequences are not similar to each other. Thus it is very important to find proteins with similar structures (even in part) from the growing database to analyze protein functions.

Structure similarity search methods for protein structure databases can be classified into two types. One is by comparing each database entry with the query. There are many comparison algorithms for protein structures [10], and the results could be very accurate, but it will require enormous amount of time to apply against very large databases. The other approach is by indexing with some important features of structures [1, 3, 6, 5, 8, 12]. In ordinary, these methods can search queries more efficiently, but with less accuracy than the pairwise comparison-based methods. The accuracy of comparison of two protein structures is often measured by RMSD (root mean square deviation) [2, 9, 17] or sometimes by URSMD (unit-vector root mean square deviation [7, 15]; see section 2.1 for more details). But it has been considered too difficult to design indexing structures that strictly consider the RMSD or the URMSD.

In this paper, we propose a new data structure called geometric suffix tree that succeeds in finding all the substructures whose RMSD or URMSD to a query is not larger than some given bound. As the name implies, our data structure is very similar to the suffix tree that is very famous as an indexing data structure for strings. The geometric suffix tree can be stored in $O(n)$ space where n is the sum of the lengths of the proteins in database. It takes $O(n^3)$ time if we construct the data structure naively, but we propose an $O(n^2)$ construction algorithm for it. Furthermore, the experiments will show that the construction time of the geometric suffix tree is almost linear to the size of the database in practice, when applied to a protein structure database. Moreover, we propose an efficient search algorithm for substructure queries. This data structure is also useful for finding structural motifs, clustering substructures, and so on.

Organization of this paper is as follows. In section 2, we explain related work as preliminaries. In section 3, we describe the definition of the proposed geometric suffix tree. In sections 4 and 5, we explain algorithms for constructing the data structure and algorithms for searching queries. In section 6, we demonstrate experimental results. In section 7, we conclude our results.

2 Related Work

2.1 RMSD and URMSD

A protein is a chain of amino acids. Each amino acid has one unique carbon atom named C_α, and we often use the coordinates of the C_α atom as the representative position of the amino acid. The set of C_α atom positions of all the amino acids in a protein is called the backbone of the protein, and is often used for protein structure analysis in previous work. We also consider the backbone as the target to index in this paper.

The most popular and basic measure to determine geometric similarity between two sets of points like the positions of backbone atoms is the RMSD (root mean square deviation) [2, 9, 17], if we know which atom in one structure corresponds to which atom in the other. The measure describes the similarity of two structures when one of the point sets is rotated and translated reasonably. Let the two sets of points to be compared be $P = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_n\}$ and $Q = \{\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_n\}$, where $\vec{p}_i$ and $\vec{q}_j$ are coordinates in 3-D space, and we consider $\vec{p}_i$ corresponds to $\vec{q}_i$ for each i. The RMSD is the minimum value of $\{(\sum_{i=1}^{n} \|\vec{p}_i - (R \cdot \vec{q}_i + \vec{v})\|^2)/n\}^{1/2}$ over possible rotation matrices R and translation vectors $\vec{v}$, where $\| \cdot \|$ denotes the norm. Let $\hat{R}(P,Q)$ and $\hat{\vec{v}}(P,Q)$ be R and $\vec{v}$ that minimizes the value. We call $\sum_{i=1}^{n} \|\vec{p}_i - (\hat{R}(P,Q) \cdot \vec{q}_i + \hat{\vec{v}}(P,Q))\|^2$ the MSSD (minimum sum square distance) of P and Q. It is known that $\hat{\vec{v}}(P,Q) = \sum (\vec{p}_i - \hat{R}(P,Q) \cdot \vec{q}_i)/n$, *i.e.*, the distance is minimized when the centroids of the two point sets are translated to the same point. Hence, if both of the point sets are translated so that their centroids are located at the origin of the coordinates, the RMSD/MSSD problem is reduced to a problem of finding R that minimizes $f(R) = \sum_{i=1}^{n} \|\vec{p}_i - R \cdot \vec{q}_i\|^2$. We can find this $\hat{R}(P,Q)$ in linear time by using singular value decomposition (SVD) [2, 17] as follows. Let $H = \sum_{i=1}^{n} \vec{p}_i \cdot \vec{q}_i^{\,t}$. Then $f(R)$ can be described as $\sum_{i=1}^{n} (p_i^T p_i + q_i^T q_i) - trace(R \cdot H)$, and $trace(RH)$ is maximized when $R = VU^T$, where $U\Lambda V$ is the SVD of H. Hence $\hat{R}(P,Q)$ can be obtained in constant time from H (see [13] for SVD algorithms). Note that there are rare degenerate cases where $det(VU^T) = -1$, which means that VU^T is a reflection matrix. We ignore the degenerate cases in this paper. In this way, we can compute the RMSD/MSSD values in $O(n)$ time.

The URMSD (unit-vector root mean square deviation) [7, 15] is a variation of the RMSD. The RMSD is sometimes influenced badly by very distant pairs of points, and the URMSD is designed to avoid such influence. It is the minimum value of $\{(\sum_{i=1}^{n-1} \|\vec{p}'_i - R \cdot \vec{q}'_i\|^2)/(n-1)\}^{1/2}$ over possible rotation matrices R, where $\vec{p}'_i = (\vec{p}_{i+1} - \vec{p}_i)/\|\vec{p}_{i+1} - \vec{p}_i\|$ and $\vec{q}'_i = (\vec{q}_{i+1} - \vec{q}_i)/\|\vec{q}_{i+1} - \vec{q}_i\|$. Let $\check{R}(P,Q)$ be R that minimizes the value. We call $\sum_{i=1}^{n-1} \|\vec{p}'_i - \check{R}(P,Q) \cdot \vec{q}'_i\|^2$ the UMSSD (unit-vector minimum sum square distance). The URMSD/UMSSD can be computed with the same strategy in $O(n)$ time, *i.e.* by computing the SVD of $H' = \sum_{i=1}^{n} \vec{p}'_i \cdot (\vec{q}'_i)^t$.

2.2 Suffix Trees

The suffix tree [11, 14, 16, 18, 19] of a string $S \in \Sigma^n$ is the compacted trie of all the suffixes of $S^+ = S\$$ where $\$$ is a character such that $\$ \notin \Sigma$. This data structure can be stored in $O(n)$ space and moreover is known to be buildable in $O(n)$ time. Each leaf represents a suffix of the string S^+, and each node represents some substring. Figure 1 shows an example. This data structure is very useful for various problems in sequence pattern matching. Using it, we can query a substring of length m in $O(m)$ time, we can find frequently appearing substrings in a given sequence in linear time, we can find a common substring of many sequences in linear time, and so on [14].

Not much work has been done for applying this data structure to protein structures. The PSIST [12] is the only index data structure for protein structures based on the suffix trees as far as we know. It converts local features of the amino acid chain into some alphabets and constructs suffix trees over the converted alphabet sequences, without considering global measures like the RMSD at all.

3 Geometric Suffix Tree Data Structure

In this section, we describe the definition of the geometric suffix tree. We propose two versions of the data structure. One is for queries based on the RMSD, which we call the RMSD geometric suffix tree (RGST for short). The other is for queries based on the URMSD, which we call the URMSD geometric suffix tree (UGST for short). We suppose a protein structure is represented by its C_α atom coordinates sequence $P = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_n\}$. Let $P[i..j]$ denote $\{\vec{p}_i, \vec{p}_{i+1}, \ldots, \vec{p}_j\}$, which we call a substructure of P. Note that there are $O(n^2)$ substructures for $P = P[1..n]$. We call a substructure $P[i..n]$ a suffix substructure of P. Conversely, we call a substructure $P[1..i]$ a prefix substructure of P. The main aim of the geometric suffix tree is to find efficiently a substructure $P[i..(i+k-1)]$ whose RMSD or URMSD from a given query structure $Q[1..k]$ is not larger than some given bound.

The geometric suffix tree for $P[1..n]$ is a rooted tree data structure that has following

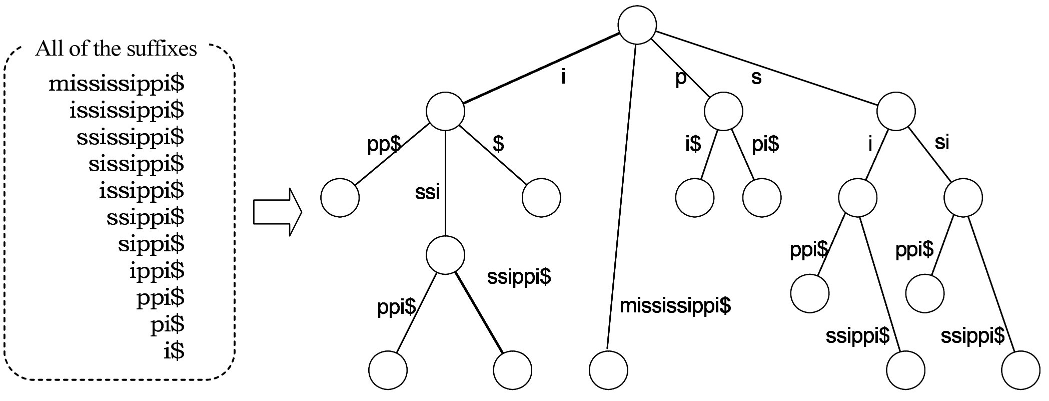


Figure 1: The suffix tree of a string 'mississippi'.

features.

1. All the internal nodes have more than one child, whereas the root has one or more children. (We call nodes other than the root and the leaves internal nodes.) The number of leaves is n. Each leaf corresponds to one suffix substructure of P, and no two leaves correspond to the same suffix substructure. Let $leaf(i)$ denote the leaf that corresponds to $P[i..n]$.
2. Each edge that does not end at a leaf is given a positive length. Other edges are given non-negative lengths. The path length from the root to a node is called the depth of the node. Let $l(e)$ denote the length of the edge e and let $depth(v)$ denote the depth of the node v. For any leaf $u = leaf(i)$, $depth(u) = n - i + 1$ *i.e.* the length of $P[i..n]$.
3. Each edge e with positive length has information of P's substructure $P(e)$ whose length is $l(e)$, a rotation matrix $R(e)$, and a translation vector $\vec{v}(e)$. For the edge e, let $S(e)$ be a structure that can be obtained by rotating $P(e)$ with the rotation matrix $R(e)$ and translating it with the translation vector $\vec{v}(e)$ after that. We call $S(e)$ the edge structure of e. Edges with zero lengths do not have the information.
4. The 'node structure' $S(x)$ for a node x is a structure that can be obtained by concatenating edge structures of the edges on the path from the root to the node x. For any leaf $v = leaf(i)$ and its arbitrary ancestor w (w can be v itself), the MSSD (in case of RGSTs, or UMSSD in case of UGSTs) between $S(w)$ and $P[i..(i+depth(w)-1)]$ must not be larger than some given fixed bound b.
5. For an edge $e = (v, w)$ with a positive length (where v is the parent of w), the 'branching structure' $str(e)$ is a structure that is obtained by concatenating $S(v)$ and the first coordinates of $S(e)$. For any internal node v with more than two outgoing edges with positive lengths, the MSSD (in case of RGSTs, or UMSSD in case of UGSTs) between $str(e_1)$ and $str(e_2)$ must be larger than b, where e_1 and e_2 are arbitrary two of the edges.

We can store the substructure information for each edge by just remembering the indices, which means we need only $O(1)$ memory for each edge. Thus the total memory space for storing the data structure is only $O(n)$, as there are at most $O(n)$ edges and nodes in the tree. Note that we can easily extend this data structure for multiple protein structures, as we extend the ordinary suffix tree to the generalized suffix trees for strings of alphabets [14]. We call the extended data structure the generalized geometric suffix tree.

4 Constructing Geometric Suffix Trees

Given a coordinates list P of a protein backbone structure and some given MSSD (for

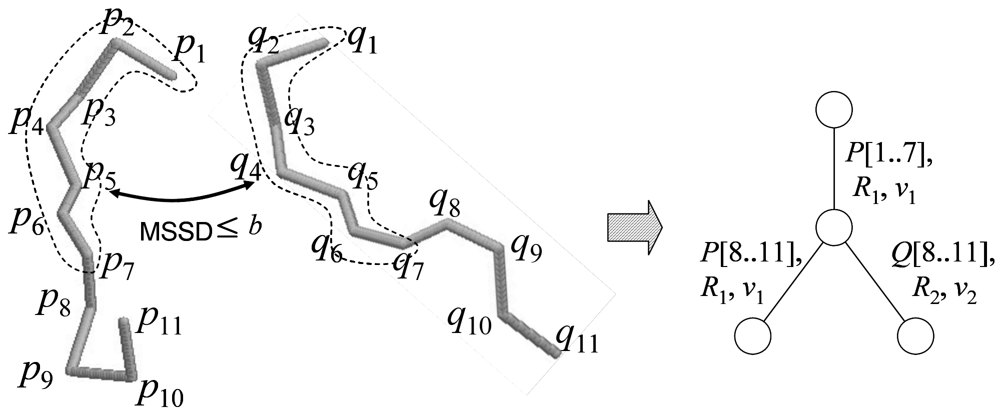
RGSTs or UMSSD for UGSTs) bound b, we can naively construct the geometric suffix tree by adding suffix substructures one by one as follows.

Algorithm 1 *At first, construct a tree with only the root node. For each suffix substructure $P[i..n]$ $(1 \leq i \leq n)$ of the protein, set the root to v and do the following.*

1. *From a set of v's outgoing edges with positive lengths, find an edge e such that the MSSD (for RGSTs, or UMSSD for UGSTs) between $P[i..(i + depth(v))]$ and the branching structure $str(e)$ is smaller than b. If no such edge exists, go to step 2. Otherwise go to step 3.*
2. *Add a new outgoing edge $e' = (v, w)$ to v, where the new leaf w corresponds to the suffix $P[i..n]$. Let $P(e')$ be $P[(i + depth(v))..n]$. If v is the root, let $R(e')$ be the identity matrix and let $\vec{v}(e')$ be a zero vector. Otherwise, in case of RGSTs, let $R(e')$ be $\hat{R}(S(v), P[i..(i + depth(v) - 1)])$, and let $\vec{v}(e')$ be $\hat{\vec{v}}(S(v), P[i..(i+depth(v)-1)])$. In case of UGSTs, let $R(e')$ be $\check{R}(S(v), P[i..(i + depth(v) - 1)])$, and let $\vec{v}(e')$ be $(S(v)[depth(v)] - R(e') \cdot P[i + depth(v) - 1])$. Then stop.*
3. *Let w be the node where the edge e ends. Find the longest prefix substructures of $S(w)$ and $P[i..n]$ whose MSSD (for RGSTs, or UMSSD for UGSTs) is not larger than b, and let the length be l. If $l < depth(w)$ go to step 4. If $l = depth(w)$ and $l < n - i + 1$, set w to v and go to step 1. Otherwise, add a new outgoing edge (w, u) whose length is 0, where u corresponds to the suffix substructure $P[i..n]$. Then stop.*
4. *Insert a new node u between v and w. Let $e_1 = (v, u)$ and let $e_2 = (u, w)$. Let $P(e_1)$ be the prefix substructure of $P(e)$ of length $(l - depth(v))$, and $P(e_2)$ be the suffix substructure of $P(e)$ of length $(depth(w) - l)$. Let $R(e_1)$ and $R(e_2)$ be the same matrix as $R(e)$, and $\vec{v}(e_1)$ and $\vec{v}(e_2)$ be the same vector as $\vec{v}(e)$. Add a new outgoing edge $e'' = (u, x)$ to u, where the new leaf x corresponds to the suffix substructure $P[i..n]$. If $l = n - i + 1$, Let the length of e'' be 0. Otherwise, let $P(e'')$ be $P[(i + l)..n]$. In case of RGSTs, let $R(e'')$ be $\hat{R}(S(u), P[i..(i + l - 1)])$ and let $\vec{v}(e'')$ be $\hat{\vec{v}}(S(u), P[i..(i + l - 1)])$. In case of UGSTs, let $R(e'')$ be $\check{R}(S(u), P[i..(i + l - 1)])$ and let $\vec{v}(e'')$ be $(S(u)[l] - R(e'') \cdot P[i + l - 1])$. Then stop.*

Figure 2 shows an example of the tree data structure just after adding two different suffix substructures (of different structures) $P = \{\vec{p}_1, \ldots, \vec{p}_{11}\}$ and $Q = \{\vec{q}_1, \ldots, \vec{q}_{11}\}$ in this order to the root, to construct some generalized RGST. In the figure, we consider the MSSD of $P[1..7]$ and $Q[1..7]$ is not larger than b, while the MSSD between $P[1..8]$ and $Q[1..8]$ is larger than b.

As the MSSD or the UMSSD between two structures of size m can be computed in $O(m)$ time, it takes $O(n^2)$ time to add one suffix to the tree if we compute it naively, which means the above naive algorithm requires $O(n^3)$ time in total. But we can reduce it to $O(n^2)$ time as follows. Recall that the MSSD of two protein structures $P[1..j]$ and $Q[1..j]$ can be obtained by computing the SVD of $H = \sum_{i=1}^{j} (\vec{p}_i - \vec{c}_p) \cdot (\vec{q}_i - \vec{c}_q)^t$ where $\vec{c}_p$ and $\vec{c}_q$ are the centroids of P and Q. H can be computed in constant time if we are given followings: $f_P(j) = \sum_i^j \vec{p}_i$, $f_Q(j) = \sum_i^n \vec{q}_i$, and $g(j) = \sum_i^n \vec{p}_i \cdot \vec{q}_i^t$. Add to these values, we need $h_P(j) = \sum_{i=1}^n \vec{p}_i^t \vec{p}_i$ and $h_Q(j) = \sum_{i=1}^2 \vec{q}_i^t \vec{q}_i$ to compute the MSSD or RMSD values in constant time. Notice that all of these can be computed incrementally in constant time from $f_P(j-1)$, $f_Q(j-1)$, $g(j-1)$, $h_P(j-1)$, and $h_Q(j-1)$. It means that we can add a suffix to the tree in $O(n)$ time, and therefore we can construct the RGST in $O(n^2)$ time. Similarly, we can compute the UMSSD of two protein structures $P[1..j]$ and $Q[1..j]$ in constant time if we have the following values: $g'(j) = \sum_{i=1}^j \vec{p}'_i \cdot (\vec{q}'_i)^t$, $h'_P(j) = \sum_{i=1}^j (\vec{p}'_i)^t \vec{p}'_i$, and $h'_Q(j) = \sum_{i=1}^j (\vec{q}'_i)^t \vec{q}'_i$. We can easily see that these can also be computed from $g'(j-1)$, $h'_P(j)$ and $h'_Q(j)$ in constant time, and therefore we conclude that the UGST



p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11}
q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8 q_9 q_{10} q_{11}
MSSD$\leq b$
P[1..7], R_1, v_1
P[8..11], R_1, v_1
Q[8..11], R_2, v_2

Figure 2: Constructing a geometric suffix tree.

can also be constructed in $O(n^2)$ time.

5 Geometric Suffix Tree Applications

Consider an edge $e = (u, v)$ in a geometric suffix tree. We call the node structure of v a representative structure on e. Furthermore we also call a structure that is obtained by concatenating node structure of u and some prefix substructure of edge structure of e a representative structure on e.

Let S and T be two representative structures where S is a prefix substructure of T. Then it is easy to see that the MSSD (or UMSSD) of T and $Q[1..|T|]$ is always larger or equal to that of S and $Q[1..|S|]$. Using this feature, all maximal substructures whose RMSD (or URMSD) to a query $Q[1..m]$ is within some bound d can be computed efficiently as follows. First we find all the maximal representative substructures whose RMSD (or URMSD) to the query Q is within $\sqrt{b/n}+d$, where b is the MSSD (or UMSSD) bound used for constructing the geometric suffix tree. Let E be the set of edges to which the collected representative substructures correspond. After that, find all the leaves that are descendants of the edges in E. Then the suffixes that correspond to the collected leaves are candidates of the answer substructures, whose RMSDs (or URMSDs) must be checked one by one.

Ordinary suffix trees have tremendous number of applications in string pattern matching [14]. Like them, applications of geometric suffix trees are not limited to the database search. A long representative structures whose corresponding edge has many descendants is a repeated structure in a protein structure, which could have some meaning. By constructing the generalized geometric suffix tree of two or more functionally-related protein structures, we could find structural motifs. We could further use this fundamental data structure for designing more complicated combinatorial matching algorithms on protein structures, such as clustering algorithms and classifier algorithms.

6 Experimental Results

In this section, we demonstrate the performance of the generalized geometric suffix trees through experiments on a Sun Fire 15K super computer with 288 GB memory and 96 UltraSPARC III Cu CPUs running at 1.2GHz. Note that we used only one CPU for each experiment. As a data for experiments, we used a set of 228 myoglobin or myoglobin-related PDB data files containing 275 protein structures. The total number of amino acids in the protein set is 41,719.

Table 1 shows the computation time for constructing generalized RGST against databases of different sizes, setting 400Å^2 to the MSSD bound. In the experiment (1), we used all the 275 proteins to index. In

Table 1: Computation time for constructing generalized GSTs. ($b = 400\text{Å}^2$)

Database	#sequence	(#a.a.)	Time (sec)	GST Size
(1) Entire database	275	(41,719)	53.15	57,241
(2) Subset A	111	(16,983)	17.68	25,942
(3) Subset B	54	(8,267)	7.91	13,050

the experiments (2) and (3), we used different subsets of them. The '#sequence(#a.a.)' column shows the numbers of sequences and amino acids contained in the protein sets. The 'Time' column shows the computation time, while the 'GST Size' column shows the numbers of nodes in the constructed geometric suffix trees. According to the table, The computation time is almost linear to the size of the databases, though the theoretical time bound is $O(n^2)$. We consider that it is because there is a bound on protein lengths.

Next, we examined the query speed on the RGSTs with different MSSD bounds. Table 2 shows the results, where '$b = \ldots$' denotes the MSSD bound in Å^2. We used two protein substructures of same length as queries: In experiment (a), we used as a query a substructure from the 20th amino acid to the 69th amino acid of a myoglobin's structure obtained from the PDB entry named 103M. In experiment (b), we used a protein that is unrelated to myoglobins: A substructure from the 20th amino acid to the 69th amino acid of a rhodopsin's structure obtained from the PDB entry named 1F88. In both experiments, we examined query time by setting two different RMSD bounds: $d = 1.0\text{Å}$ and $d = 5.0\text{Å}$. In the table, the '#found' column shows the numbers of found substructures similar to the query. According to the experiments, the query is very fast when the RMSD bound for the query is small. The best setting for the MSSD bound of the geometric suffix tree is different for the two queries. If a lot of similar substructures to the query exist, it is better to use a little smaller MSSD bounds, which is observed also in other experiments not shown on this paper.

7 Concluding Remarks

We proposed a new data structure called the geometric suffix tree for indexing the protein 3-D structures. The data structure can be stored in $O(n)$ space where n is the database size, and we presented an $O(n^2)$ construction algorithm for it. Moreover, we showed through experiments that we can build the data structure in quasi-linear time in practice. We also showed that we can search for queries very efficiently with the geometric suffix tree.

It is an open problem whether we can improve the theoretical time bound for building the geometric suffix tree. As future work, we are now working on utilizing this data structure for further combinatorial matching problems and machine learning problems on protein structures.

Acknowledgement

The author would like to thank Prof. Tatsuya Akutsu for fruitful discussions on protein comparison algorithms. All the computational experiments in this research was done on the Super Computer System, Human Genome Center, Institute of Medical Science, University of Tokyo.

References

[1] T. Akutsu, K. Onizuka, and M. Ishikawa. New hashing techniques and their application to a protein database system, *Proc. Hawaii Int. Conf. System Sciences (HICSS-28)*, Vol. 5, pp. 197-206, 1995.

[2] K. S. Arun, T. S. Huang, and S. D. Blostein, Least-squares fitting of two 3-D point sets. *IEEE Trans Pattern Anal.*

Table 2: Query time (sec) using generalized GSTs with various MSSD bounds.

Queries		$b = 1$	$b = 100$	$b = 400$	$b = 900$	$b = 1600$	$b = 2500$	#found
(a)	$d = 1.0$Å	1.63	0.56	0.39	0.43	0.60	0.87	19
	$d = 5.0$Å	11.70	5.08	5.66	6.55	6.63	6.63	217
(b)	$d = 1.0$Å	1.63	0.73	0.48	0.33	0.19	0.21	0
	$d = 5.0$Å	16.13	7.83	7.93	8.00	7.58	7.20	0

Machine Intell., Vol. 9, pp. 698-700. 1987.

[3] Z. Aung, W. Fu and K. Tan. An efficient index-based protein structure database searching method. *Proc. Intl. Conf. on Database Systems for Advanced Applications*, pp. 311-318, 2003.

[4] H. M. Berman, J. Westbrook, Z. Feng, et al. The protein data bank. *Nucl. Acids Res.*, Vol. 28, pp. 235-242, 2000.

[5] O. Çamoğlu, T. Kahveci and A. Singh. Towards index-based similarity search for protein structure databases. *IEEE Computer Society Bioinformatics Conference*, pp. 148-158, 2003.

[6] T. Can and Y. Wang. CTSS: a robust and efficient method for protein structure alignment based on local geometrical and biological features. *IEEE Computer Society Bioinformatics Conference*, pp. 169-179, 2003.

[7] L. P. Chew, D. Huttenlocher, K. Kedem and J. Kleinberg. Fast detection of common geometric substructure in proteins. *J. Comput. Biol.*, Vol. 6, No. 3, pp. 313-325. 1999.

[8] I. Choi, J. Kwon and S. Kim. Local feature frequency profile: A method to measure structural similarity in proteins. *Proc. Natl. Acad. Sci*, Vol. 101, No. 11, pp. 3797-3802, 2004.

[9] D. W. Eggert, A. Lorusso and R. B. Fisher. Estimating 3-D rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, Vol. 9, pp. 272-290. 1997.

[10] I. Eidhammer, I. Jonassen, and W. R. Taylor. Structure Comparison and Structure Patterns. *J. Computational Biology*, Vol. 7, No. 5, pp. 685-716. 2000.

[11] M. Farach. Optimal suffix tree construction with large alphabets. *Proc. 38th IEEE Symp. Foundations of Computer Science,* pp. 137-143. 1997.

[12] F. Gao and M. J. Zaki. PSIST: Indexing Protein Structures using Suffix Trees. *Proc. IEEE Computational Systems Bioinformatics Conference (CSB)*, pp. 212-222. 2005.

[13] G. H. Golub and C. F. Van Loan. *Matrix Computation.* 3rd eds. John Hopkins University Press. 1996.

[14] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology,* Cambridge University Press. 1997.

[15] K. Kedem, P. Chew and R. Elber. Unit-vector RMS (URMS) as a tool to analyze molecular dynamics trajectories. *Proteins: Struct. Funct. Genet.*, Vol. 38, pp. 1-12. 1999.

[16] E. M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM.*, Vol. 23, pp. 262-272. 1976.

[17] J. T. Schwartz and M. Sharir. Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. *Intl. J. of Robotics Res.*, Vol. 6, pp. 29-44. 1987.

[18] E. Ukkonen. On-line construction of suffix-trees. *Algorithmica,* Vol. 14, pp. 249-260. 1995.

[19] P. Weiner. Linear pattern matching algorithms. *Proc. 14th Symposium on Switching and Automata Theory*, pp. 1-11. 1973.