Fast Protein 3D Surface Search

Sungchul Kim
POSTECH
Pohang
South Korea
+82-10-9591-1077
subright@postech.ac.kr

Lee Sael
The State University of New York Korea
Incheon
South Korea
+82-32-626-1215
sael@sunykorea.ac.kr

Hwanjo Yu
POSTECH
Pohang
South Korea
+82-10-4118-7006
hwanjoyu@postech.ac.kr

ABSTRACT

Functionally annotating protein structures of unknown function is one of the important challenges in Bioinformatics. An informatics approach to predict the function of a protein is by analyzing the functions of other structurally similar proteins. Ability to search and retrieve similar protein structures among large dataset is crucial in this approach. Here, we propose a novel approach for efficient protein structure search where protein structures are represented as vectors by 3D-Zernike Descriptor (3DZD). Surface shape of protein tertiary structure is compactly represented with 3DZD encoding. This simplified representation accelerates the structural search from daylong to matter of seconds. However, further speed up is required to address the scenarios where multiple users access the database at the same time. We address this need for further speed up in protein structural search by exploiting the fast k nearest neighbor algorithms on the 3DZDs. The results show that the proposed methods significantly improve the searching speed. In addition, we introduce an extended approach for protein structure search based on the methods that utilize the 3DZD characteristic. Experiments show that the searching time reduced 75.41% by the fast k-nearest neighbor algorithm, 88.7% by the extended fast k-nearest neighbor algorithm, 88.84% by the fast threshold-based nearest neighbor algorithm, and 91.53% by the fast extended threshold-based nearest neighbor algorithm. In a simulationed test case, the extended threshold-based algorithm which had the highest speed improvement in the initial test case, showed speed improvement up to 87.48% compared to linear scan.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Content Analysis and Indexing, Indexing methods*; H.3.1 [**Information Systems**]: Information Search and Retrieval—*Information Search and Retrieval*

General Terms

Algorithm, Theory

Keywords

protein surface shape; protein structure classification; database search; structure similarity; 3D Zernike descriptor

1. INTRODUCTION

One of the challenges in bioinformatics field is annotating the function of protein structures [1, 2, 3]. With the acceleration of semi-automated structure solving efforts such as the Protein Structural Initiative[1], there has been a steady increase in the number of protein structures that the function are not known, i.e., there are over 3200 structures of unknown functions in the Protein Data Bank (PDB). Given the protein of unknown function, computational function prediction methods are used to find similar sequential/structural patterns in known proteins. There is a long stream of researches on protein function annotation for newly determined genes and genomes by sequence database searches. However, the 3D structures of proteins are more conserved over the evolutionary changes than the sequences, thus the comparison of structural information provides more reliable similarity measures. Therefore, researches on functional studies for protein structure data are of the highest priority [4, 5, 6].

Much structure alignment based pair-wise comparison methods were introduced. These methods first find the best alignment between the two structure, then compute the Root Mean Square Deviation (RMSD) between the core atomic positions, e.g., alpha carbon coordinates, of the aligned proteins. However, most of methods based on structural alignment cannot be used for protein structure search against large database, since it is computationally expensive to compute their similarities. Sael et al. introduced a new approach for fast protein surface similarity search using new protein representations by 3DZDs [7]. 3DZD has three advantages: 1) fast nearest neighbor search, 2) rotational invariance, and 3) easy adjustment of the resolution of the structural representation resolution. Particularly, using 3DZDs, it is possible to retrieve similar proteins in seconds among 150,000 protein structures using the linear scan approach.

However, few seconds are still too slow for a real time search system since response time increases further when multiple users access the database simultaneously. To further improve the search speed, we exploit the notion of fast

[1]http://www.nigms.nih.gov/Research/FeaturedPrograms/PSI

nearest neighbor algorithms by nonlinear embedding [8, 9]. Therefore, this paper proposes variants of fast nearest neighbor algorithm for 3DZDs and show the speed improvement in protein structure search. Specifically, we explore four approaches: 1) k-nearest eighbor, 2) extended k nearest neighbor, 3) threshold-based nearest neighbor, and 4) extended threshold-based nearest neighbor approach.

The results show that all the variants of the fast nearest neighbor algorithms improve the searching speed. The searching time reduced 75.41% by the fast k-nearest neighbor algorithm (fKNN), 88.7% by the extended fast k-nearest neighbor algorithm (efKNN), 88.84% by the fast threshold-based nearest neighbor algorithm (fTNN), and 91.53% by the extended fast threshold-based nearest neighbor algorithm (efTNN). In simulation result, the extended threshold-based algorithm which is the best works faster than the linear scan by up to 87.48%. It indicates that users can take advantage of this system without delay when there are many users accessing at the same time

The paper is organized as follows. Section 2 introduces related works about protein structure search and nearest neighbor search. Section 3 describes fast nearest neighbor algorithms including the extended approaches. In Section 4, we provide experimental results to verify the efficiency of our approaches, and conclusion with future works.

2. RELATED WORK

In this section, we provide a brief introduction of protein structure search, 3D-Surfer, which is our target system to improve, and nearest neighbor search.

2.1 Protein Structure Search

A protein consists of a sequence of amino acid (AA) residues. A sequence of AA residues folds into a 3-dimensional (3D) structure in space and forms a functional protein. A 3D structure of a protein is recorded in a pdb file format as a set of Cartesian coordinates of all the atoms in the protein. The 3D structure contains rich information relating to function and evolution of the protein. Therefore, structural information provide more reliable similarity measures between proteins and protein function can be predicted by finding similar proteins to the queried protein with unknown function.

Earlier structural similarity measurements were designed for pair-wise analysis where the user only needed to compare handful of protein structures [10, 11, 12]. However, as the number of known structures increased, more methods were proposed for similarity search in protein database [13, 14]. One of the most intuitive approaches is to compare the coordinates of corresponding residues or atoms of proteins after structural alignment [15, 16]. Root Mean Square Deviation (RMSD) is often used as the similarity measure. Due to its high computational complexity, structure alignment is done by using Dynamic Programming (DP) or its extensions [11, 17, 18].

There are major structure databases such as PDB, CATH [19], and SCOP [20] which provides only keyword search and browsing of pre-computed classification. Some database systems that are able to take a query structure are for the search includes Distance matrix ALIgnment (DALI) server [21], Vector Alignment Search Tool (VAST) search [22], and eF-site database [23]. Given a query protein structure, it takes around an hour to finish searching their databases. Zeyar et al. suggests an index technique called ProtDex for fast search in 3D protein structure database [24]. Although it works faster than DaliLite [13] which is one of the most popular protein structure search algorithms, ProtDex requires over a few minutes for retrieval making it not practical for online database searches.

2.2 3D-Surfer

3D-Surfer is a new and efficient protein structural search system which represents protein structures based on 3D-Zernike Descriptor (3DZD). The major advantage of 3DZD is that it allows a fast nearest neighbor search of protein structures. It has been verified that the retrieved similar proteins by 3D-Surfer have similar functional and evolutional information in terms of SCOP classification [25]. Some of the characteristics of the 3DZDs are that it is rotationally invariant, the resolution of the representation of protein structures are easily adjusted by changing the order, and the descriptors of the lower order are contained in the descriptors of the higher order.

2.3 Nearest Neighbor Search

The nearest neighbor search is one of the most important tasks in machine learning, pattern recognition, and information retrieval which are an optimization problem for finding closest points in metric spaces. The simplest solution for this problem is linear scan which computes the distance from the query point to every point in database sequentially. However, the running time of this technique is proportional to the number and dimensionality of data. To make the nearest neighbor search algorithm more efficient, tree-based data structures such as X-tree, TV-tree, and SR-trees [26, 27, 28] are often used. The advantage of the tree structures is ability to reduce the number of search candidates based on the well-structured data. However, they may not work well on high-dimensional and large-scale data since it is computationally expensive to construct data structures and requires large amount of memory. In this work, we exploit the fast nearest neighbor search introduced by Hwang et al [8] which can efficiently retrieve exact nearest neighbors on high-dimensional data. In addition, their method retrieves exact nearest neighbors according to given query.

3. MATERIALS AND METHODS

In this section, we introduce the protein structure dataset and their 3D-Zernike Descriptor (3DZD) representation. Then, we provide the descriptions of the fast nearest neighbor algorithms including its extension.

3.1 Protein Structural Dataset and 3D-Zernike Descriptor

3DZDs are compact and rotationally invariant representation of 3D structures. 3DZD has been successfully used for protein [7] and ligand structure analyses [29]. We provide brief description of 3DZD for reader's convenience. Detailed description can be found in [30, 31].

The 3DZD descriptors of 158,781 protein chain structures was obtained through 3D-Surfer database. The entire structures in PDB was collected and processed on 2009 [32]. For each of the pdb files that contain one to several protein chains, the chains were separated and surfaces of each chain were obtained through molecular surface calculation program, MSROLL version 3.9.3 [33], and then voxelized.

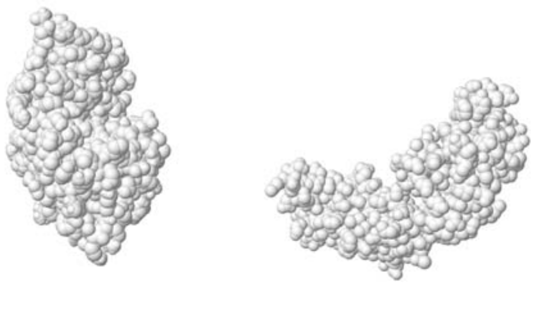

(a) 1mrg-A (b) 3DVG-A

Figure 1: Two example proteins; 1mrg-A and 3DVG-A

Each of the voxelized protein surface were used as an input to 3DZD conversion program and a vector of 121 numbers called invariants were computed.

In 3DZD construction, a given 3D function, $f(x)$, that contains a surface information of protein, is expanded into a series of Zernike-Canterakis bases defined as follows.

$$Z_{nl}^{m}(r,\vartheta,\varphi) = R_{nl}(r)Y_{l}^{m}(\vartheta,\varphi) \tag{1}$$

where $-l < m < l, 0 \leq l \leq n$, $(n-l)$ is even, $Y_l^m(\vartheta,\varphi)$ are spherical harmonics, and R_{nl} are radial functions constructed to convert $Z_{nl}^m(r,\vartheta,\varphi)$ to polynomials in the Cartesian coordinates, $Z_{nl}^m(x)$. To obtain the 3DZD of $f(x)$, 3D Zernike moments need to be computed first. They are defined by expanding the orthonormal bases as follows:

$$\Omega_{nl}^{m} = \frac{3}{4\pi}\int_{|x|\leq 1} f(x)\bar{Z}_{nl}^{m}(x)dx \tag{2}$$

Then, the 3DZD, F_{nl}, is computed by normalizing Ω_{nl}^m as follows.

$$F_{nl} = \sqrt{\sum_{m=-l}^{m=l}(\Omega_{nl}^{m})^2} \tag{3}$$

where n is the order of 3DZD determining the resolution of the descriptor. Then, the norms allow rotational invariance to the descriptor. For each pair of n and l, 3DZD has a series of invariants, the numbers in the vector of 3DZD, where n is ranged from 0 to the predefined order (20 in this case).

Fig. 1 is an example of 3DZD of two proteins, α-momorcharin complexed with adenine (PDB code: 1mrg-A) and ribosomal protein S27a (PDB code: 3DVG-A). As you can see, two proteins have different structures overall and their descriptor also shows visible difference (Fig. 2).

3.2 Fast Nearest Neighbor Search

Hwang et al. proposed a simple and novel technique for fast nearest neighbor search. In this section, we provide a brief description of the key idea and how this method is applied to 3DZD data set. More details can be found in [8]. To eliminate non-nearest neighbors, the method embeds the data points onto a very low dimensional space, and compares the distances between data points in the embedded space.

Given two d-dimensional vectors $\mathbf{x} = (x_1, x_2, \cdots, x_d)^T$ and $\mathbf{y} = (y_1, y_2, \cdots, y_d)^T$, the fast nearest neighbor algorithm is based on newly defined lower-bound of Euclidean

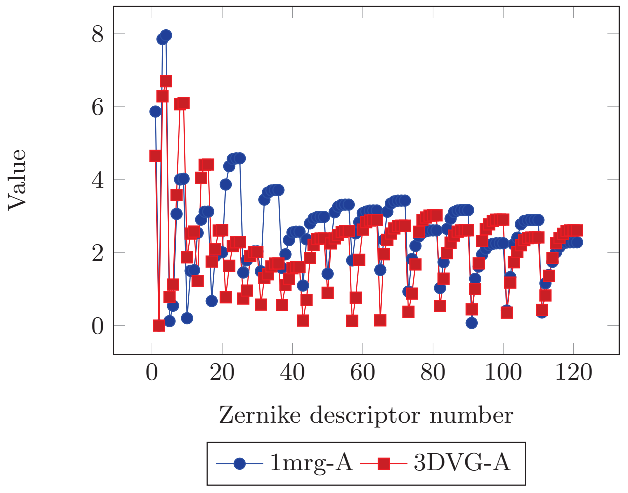


Figure 2: 3D Zernike descriptors of two example proteins

Algorithm 1 Fast K Nearest Neighbor Search (fKNN)

```
Select k vectors as a seed set of $X_k$    ▷ $X_k$ is maintained using priority queue
$\rho \leftarrow dist(\mathbf{x}_k, \mathbf{q})^2$    ▷ $\mathbf{x}_k$ is k-th nearest neighbor in $X_k$
for all $\mathbf{x} \in X$ do
    if $LB(\mathbf{x}, \mathbf{q}) \geq \rho$ then continue
    end if
    if $dist(\mathbf{x}, \mathbf{q})^2 \leq \rho$ then
        $X_k \leftarrow \mathbf{x}$
        $\rho \leftarrow dist(\mathbf{x}_k, \mathbf{q})^2$
    end if
end for
return $X_k$
```

distance between them computed as follows:

$$dist(\mathbf{x},\mathbf{y})^2 \geq d((\mu_x - \mu_y)^2 + (\rho_x - \rho_y)^2) \tag{4}$$

where the mean and standard deviation of the values in $x \in R^d$ are given by $\mu_x = \frac{1}{d}\sum_i^d x_i$ and $\rho_x^2 = \frac{1}{d}\sum_i^d (x_i - \mu_x)^2$. The theoretical proof is in [8]. Note that the lower bound of $dist(\mathbf{x},\mathbf{y})^2$ is represented by using (μ_x, μ_y) and (ρ_x, ρ_y) which is only 2-dimensional vector which is very small compared to the vector of original dimension d. Using this, the nearest neighbor search can be efficiently implemented.

Given a set of data points X, for efficient nearest neighbor search, the lower bound of all data points, $LB(\mathbf{x},\mathbf{y})$, is computed in advance. It can be done in $O(nd)$ time using $O(n)$ space. The mean and standard deviation of a query can also be computed in $O(d)$ time, then the lower bound between query and each data point is computed in $O(1)$ time. Then, given a query q, the fast k nearest neighbor search can be done according to Algo. 1.

To retrieve k nearest neighbors, we need to first select k data instances and set ρ as distance between a k-th nearest neighbor, $\mathbf{x}_k$, and query, $\mathbf{q}$. Then, at each iteration, if $dist(\mathbf{x},\mathbf{q})^2$ is less than ρ, then x is added to X_k and ρ is updated as the distance between new k-th nearest neighbor and query. It can be simply done since X_k is handled based on priority queue. Note that the filtering is done in

$O(1)$ time. In contrast, the computing the exact Euclidean distance takes $O(d)$ time.

Algorithm 2 Extended Fast K Nearest Neighbor Search (efKNN)

```
Select $k$ vectors as a seed set of $X_k$    $\triangleright$ $X_k$ is maintained using priority queue
$\rho \leftarrow dist(\mathbf{x}_k, \mathbf{y})^2$    $\triangleright$ $\mathbf{x}_k$ is k-th nearest neighbor in $X_k$
for all $\mathbf{x} \in X$ do
    if $LB(\mathbf{x}, \mathbf{q}) \geq \rho$ then continue
    end if
    if $LB_2(\mathbf{x}, \mathbf{q}) \geq \rho$ then continue
    end if
    $\sigma \leftarrow LB_2(\mathbf{x}, \mathbf{q})$
    for $i \leftarrow 2$ to 1 do
        $\sigma \leftarrow \sigma + dist(\mathbf{x}_{i,2}, \mathbf{q}_{i,2}) - LB(\mathbf{x}_{i,2}, \mathbf{q}_{i,2})$
        if $\sigma \geq \rho$ then break
        end if
    end for
    if $\sigma \leq \rho$ then
        $X_k \leftarrow \mathbf{x}$
        $\rho \leftarrow dist(\mathbf{x}_k, \mathbf{q})^2$
    end if
end for
return $X_k$
```

3.3 Extended Fast Nearest Neighbor Search

Hwang et al. introduced the tightened lower bound based on their previous work [9]. Given d-dimensional vector $\mathbf{x}$ divided into two disjoint sub-vectors with half sizes, $\mathbf{x}_{1,2} = (x_1, \cdots, x_{\lceil d/2 \rceil})^T$ and $\mathbf{x}_{2,2} = (x_{\lceil d/2 \rceil + 1}, \cdots, x_d)^T$, and $\mathbf{y}_{1,2}$ and $\mathbf{y}_{2,2}$ which is the first $\lceil d/2 \rceil$-dimensional vector and the second $\lfloor d/2 \rfloor$-dimensional vector of $\mathbf{y}$, respectively, a new lower bound of $dist(\mathbf{x}, \mathbf{y})^2$ can be obtained as follows:

$$LB_2(\mathbf{x}, \mathbf{y}) = LB(\mathbf{x}_{1,2}, \mathbf{y}_{1,2}) + LB(\mathbf{x}_{2,2}, \mathbf{y}_{2,2}) \qquad (5)$$

In addition, $LB_2(\mathbf{x}, \mathbf{y})$ is always larger than $LB(\mathbf{x}, \mathbf{y})$. More details in [9]. Based on this fact, they proposed the improved fast nearest neighbor search algorithm which exploits this tight lower bound. Based on this notion, they divided the original vector and computed tighter lower bound. However, the dimension of our dataset is not that large compared to image dataset they used. We need to tune this approach for our dataset. To do this, we simplified the improved fast nearest neighbor algorithm (Algo. 2).

Contrary to the algorithm proposed by Hwang et al., we divide once and proceed with the nearest neighbor search using LB_2 to retrieve top-k results. One more difference is that we update σ which is a candidate minimum distance using second half vectors first, since $\mathbf{x}_{2,2}$ has lower variance than $\mathbf{x}_{1,2}$. Similar to Algo. 1, to retrieve k nearest neighbors, we first select k data instances and set ρ as the distance between the k-th nearest neighbor, $\mathbf{x}_k$, and the query, $\mathbf{q}$. Then, at each iteration, if the σ is less than the ρ, $\mathbf{x}$ is inserted to X_k, and the σ is updated as the distance between the new k-th nearest neighbor and the query.

3.4 Fast threshold-based nearest neighbor search

While analyzing the results, we found that when we exploit the fast nearest neighbor algorithms, the processing time or the evaluation ratio is very different depending on

Algorithm 3 Fast Threshold-based Nearest Neighbor Search (fTNN)

```
Given $X_k = \emptyset$ and $\rho = \theta$    $\triangleright$ $\theta$ is given, and $X_k$ is maintained using priority queue
for all $x \in X$ do
    if $LB(\mathbf{x}, \mathbf{q}) \geq \rho$ then continue
    end if
    if $dist(\mathbf{x}, \mathbf{q})^2 \leq \rho$ then
        $X_k \leftarrow \mathbf{x}$
    end if
end for
return $X_k$    $\triangleright$ $|X_k|$ can vary
```

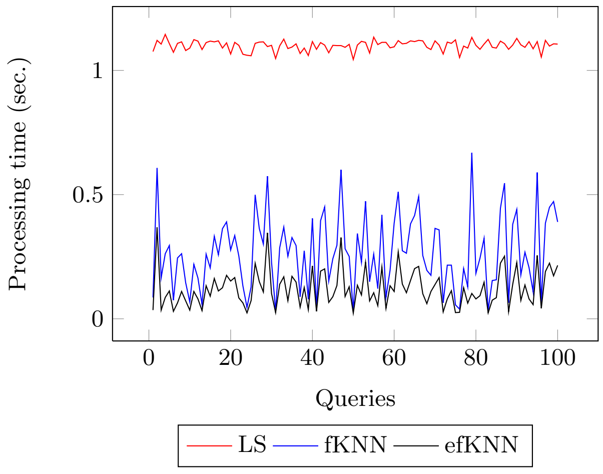
Algorithm 4 Extended Fast Threshold-based Nearest Neighbor Search (efTNN)

```
Given $X_k = \emptyset$ and $\rho = \theta$    $\triangleright$ $\theta$ is given, and $X_k$ is maintained using priority queue
for all $x \in X$ do
    if $LB(\mathbf{x}, \mathbf{q}) \geq \rho$ then continue
    end if
    if $LB_2(\mathbf{x}, \mathbf{q}) \geq \rho$ then continue
    end if
    $\sigma \leftarrow LB_2(\mathbf{x}, \mathbf{q})$
    for $i \leftarrow n$ to 1 do
        $\sigma \leftarrow \sigma + dist(\mathbf{x}_{i,2}, \mathbf{q}_{i,2}) - LB(\mathbf{x}_{i,2}, \mathbf{q}_{i,2})$
        if $\sigma \geq \rho$ then break
        end if
    end for
    if $\sigma \leq \rho$ then
        $X_k \leftarrow \mathbf{x}$
    end if
end for
return $X_k$    $\triangleright$ $|X_k|$ can vary
```

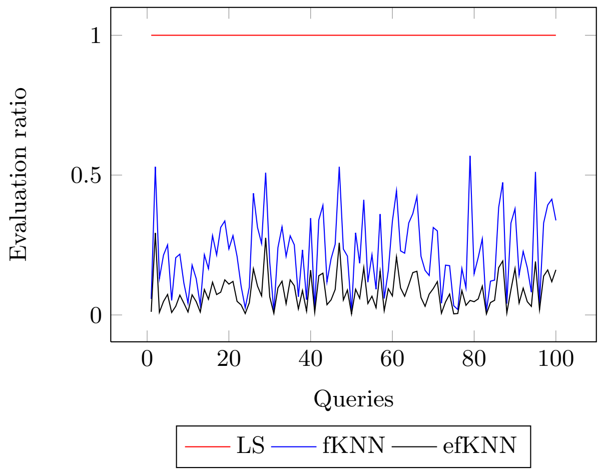
the queries. This is different from the linear search which shows stability as q varies (Fig. 3). The distribution of evaluation ratio shows almost same distribution where the evaluation ratio of linear scan is always one. It indicates that the processing time highly depends on the evaluation ratio. Intuitively, we can see that if the distance between k-th nearest neighbor and query is large, k-th nearest neighbor might be frequently changed during the search procedure. In addition, if the distance between two proteins is large, it indicates that they are not similar in terms of their structural information. Therefore, we consider the nearest neighbor search task as threshold-based nearest search. That is, the nearest neighbor search can be solved based on two different user parameters of either the number of nearest neighbor, k, or the threshold of distance between data instance and query, θ. Based on this fact, we modify the fast nearest neighbor algorithm to exploit user parameter, θ as follows: Initial θ is selected without selection of the seeds. Notice that update on ρ is not necessary, since we do not need nearest neighbors with distance larger than θ. Accordingly, Algo. 1 and Algo. 2 is modified to Algo. 3 and Algo. 4, respectively.

4. RESULTS

In this section, we verify the effectiveness of the proposed algorithms on nearest neighbor search of protein structures. Sael et al. showed that 3DZD works well on finding similar



(a) Processing time (sec.)



(b) Evaluation ratio

Figure 3: The distribution of processing time of fKNNs

proteins in terms of functional and evolutionary characteristics based on SCOP classification [7]. The SCOP provides the ordering of all proteins of known structure according to their evolutionary and structural relationships. In addition, the fast nearest neighbor search algorithms are exact method that can retrieve exact nearest neighbors from database according to the structural similarity described by 3DZD. We assume that search is correct if a protein is considered nearest neighbor by both the linear scan and our proposed method. Therefore, we only measure the efficiency in terms of processing time and evaluation ratio with small modification. The processing time could be affected by the various factors including performance of machine, the number of users, and network environment. In contrast, the evaluation ratio shows a consistent measure. The evaluation ratio is an important measure in this task which is computed as the fraction of accessed data points over the number of data points in database (1 for linear scan since it access all data points in the data set). In this work, the proposed methods access all data instances for rejecting data points based on LB, therefore we modified the evaluation ratio as the number of computation of Euclidean distance

Table 1: Comparison between LS and fKNNs

	LS	fKNN	efKNN
Proc.	1.097	0.2698	**0.124**
Eval.	1.0	0.2268	**0.0843**

Table 2: Rejected data instances by fKNNs; the number of data is 158786

	LS	fKNN	efKNN
Step 1	-	122761 (77.31%)	121965 (76.81%)
Step 2	-	-	16045 (10.1%)

over the number of database by counting it only when the computations is done for entire dimensions.

Hwang et al. verified that their methods outperforms many previous works for efficient nearest neighbor search in various dimensions using the benchmark dataset from UCI repository [9]. Taking it as granted, we only used linear scan algorithm which is currently used in 3D-Surfer system as bases of performance comparison. In overall experiment, we used 100 data points that are randomly selected from data set, and averaged the entire processing time and the evaluation ratio. The experiments were conducted on the machine, Intel Core(TM) i7 CPU (3.40GHz), and 16 GB memory.

4.1 Comparison of the proposed methods

First, we compare Linear Scan (LS) and the fast k nearest neighbor algorithm (fKNNs). The number of nearest neighbors, k, is determined as 25 since 3D-Surfer provides 25 similar proteins in 5×5 grid as default option. Table. 1 shows that fKNNs work much faster than the linear search and the extended fast k nearest neighbor algorithm (efKNN) works faster than fKNN in terms of both metrics.

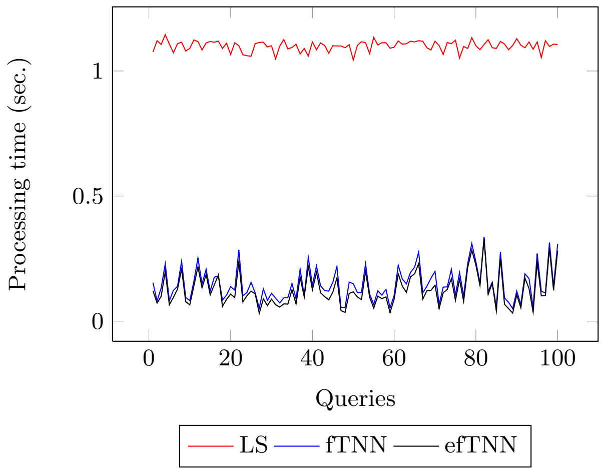
To further compare the fKNNs, we observe the number of eliminated data instances during nearest neighbor search procedure. In Algo.1, a data point is rejected if its lower bound distance is less than the minimum distance (Line 4), ρ. Similarly, in Algo.2, x is eliminated if its lower bound distance is less than σ (Line 4, 6). Table. 2 shows that fKNNs reject large amount of data instance during the searching procedure. fKNN reject 77.31% of database at step 1. efKNN reject around 76.81% at step 1 which is slightly less than fKNN. However, it rejects more than 10% of data at step 2 using the tightened lower bound, so that it rejects more data instances than fKNN in overall.

Next, we compare LS and the threshold-based methods: the fast threshold-based nearest neighbor algorithm (fTNN) and the extended fast threshold-based nearest neighbor algorithm (efTNN). The threshold, θ, that we used is 3 which is approximated from the closest value (2.93) to the average of distances between k-th nearest neighbor and queries where k is 25. We assume that the approximated value can retrieve similar amount of nearest neighbors according to queries. Table. 3 shows that fTNNs work extremely faster than LS in terms of both metrics. Note that, in this case, 31.4% of queries cannot have nearest neighbors since there is no data instance that has smaller distance from query then θ. Data points that have higher distance than *theta* can be removed from further evaluation. It is also comprehensive since some proteins may not have similar proteins. If users want to find nearest neighbors of certain query proteins that

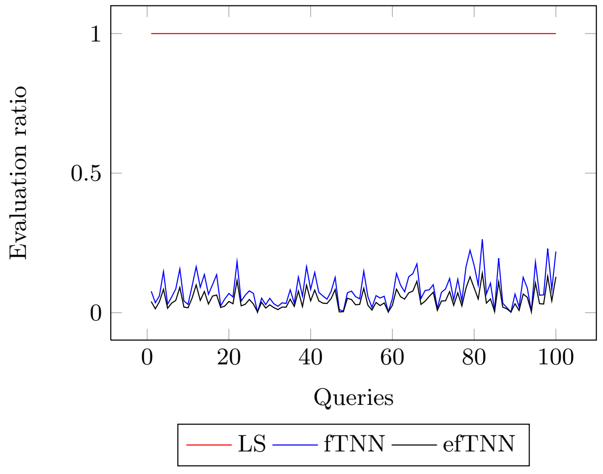
Table 3: Comparison between LS and fTNNs

	LS	fTNN	efTNN
Proc.	1.082	0.1084	**0.034**
Eval.	1.0	0.0789	**0.0101**

did not retrieve any nearest neighbors using the previous θ, they can use smaller θ to find the nearest neighbors again. Note that the proposed methods work much faster than the existing system. For fTNNs, we skip the result of the rejected data instances since it shows almost same result (Table. 2). Instead of it, we provide the processing time and evaluation ratio distribution of fTNNs to compare that of fKNNs (Fig. 4). It shows that although it is not as stable as linear search, fTNNs are more stable than fKNNs. It indicates that using fTNNs we can provide more stable service to users since its processing time relies less on the queries.



(a) Processing time (sec.)

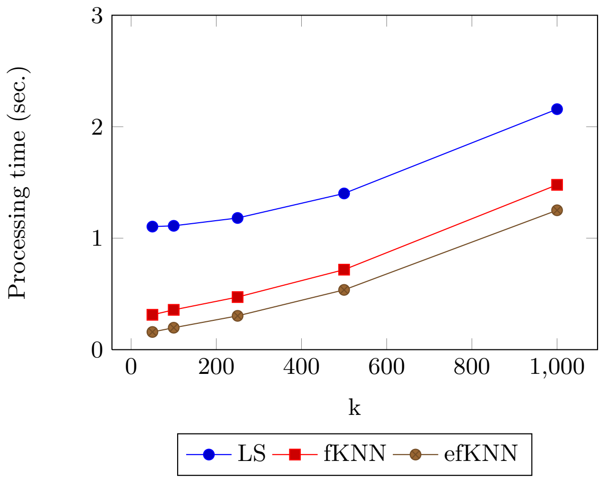


(b) Evaluation ratio

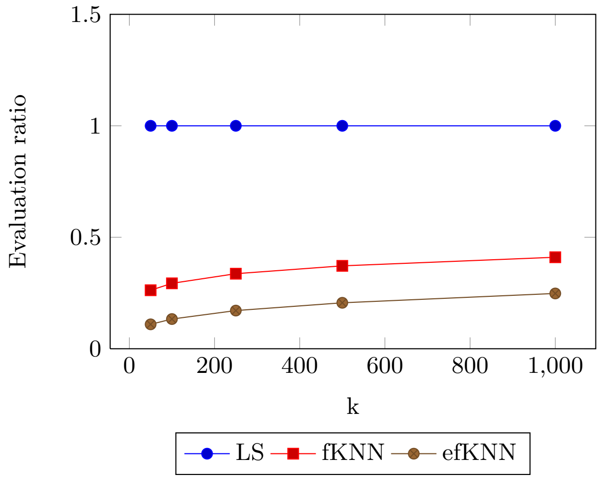
Figure 4: The distribution of processing time of fTNNs

4.2 The number of nearest neighbor, k

Although we have fixed the k to 25 in the previous experiments, we explore the effect of k and on the performance of fKNNs. For k, we used numbers adjustable k in 3D-Surfer service ($k = \{50, 100, 250, 500, 1000\}$). As we have expected, though the processing time of all methods almost linearly increases, fKNN and efKNN is always much faster than LS and fKNN works slightly slower than efKNN (Fig. 5). In terms of the evaluation ratio, the result is almost same. However, the evaluation ratio of fKNN and efKNN seems to converge to certain point. It indicates that even though the effectiveness of fKNNs maintains as k varies, the cost of recording the nearest neighbors in priority queue is more expensive.



(a) Processing time (sec.)



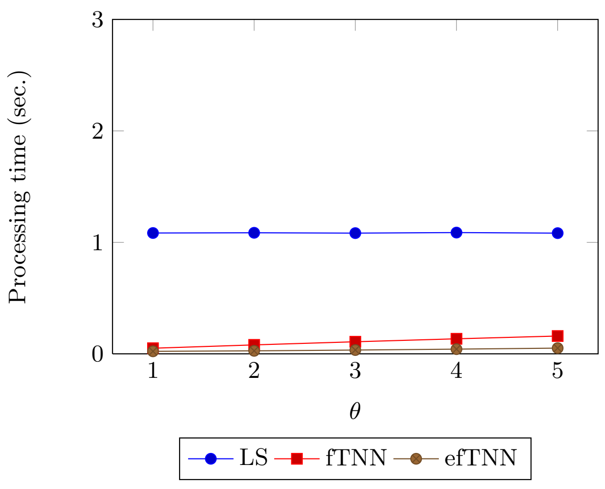
(b) Evaluation ratio

Figure 5: The result of fKNNs as k increases

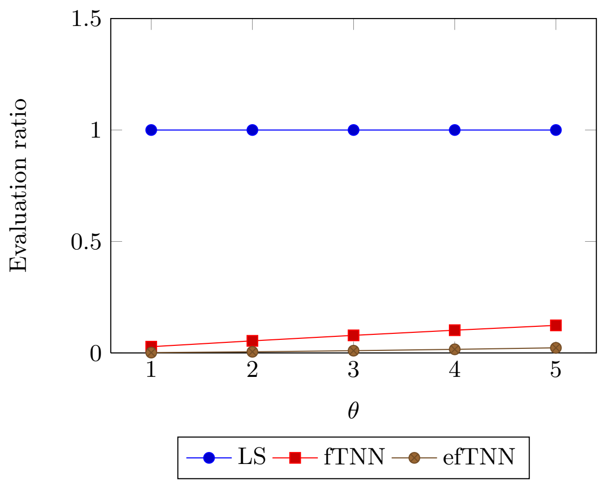
4.3 The threshold, θ

Similar to the experiment above, we conducted the experiments to observe the effect of θ and the performance of fTNNs. We set θ from 1 to 5 (for comparison, the average of distances between {50, 100, 200, 500, 1000}-th nearest neighbor and query based on the previous experiment are 3.15, 3.5, 3.94, 4.25, 4.59, respectively). As shown in Fig. 6, the processing time of LS does not change much as θ varies, since large θ does not affect much on the computational cost of LS. The rest of results are also almost same to the result above. fTNNs work faster than LS, and among them efTNN shows the best result overall. The result is intuitive since if

θ is very small most of data instances are rejected during the search procedure.



(a) Processing time (sec.)



(b) Evaluation ratio

Figure 6: The result of fTNNs as θ increases

4.4 Simulation result

To better support our statement, we simulate the senario in which multiple users enter queries at the same time via multi-threading. According to the result, the fast nearest neighbor algorithms always work faster than the linear search. Specifically, the extended approaches always work faster than the basic approaches, and θ-based approaches always work faster than the fast k nearest neighbor search algorithms. Though the differences are not significant when the number of simultaneous users is small, the proposed methods work much faster than the linear search as the number increases (Fig. 7, 8).

5. CONCLUSIONS

In this paper, we exploit the fast nearest neighbor algorithms for protein structure search where protein structures are represented as vectors by 3D-Zernike Descriptor (3DZD). Using the fast nearest neighbor algorithms with the number of nearest neighbor k as the user parameter, the searching time reduced 75.41% by the fast k-nearest neighbor algorithm, and 88.7% by the extended fast k-nearest neighbor algorithm. Using same algorithm using threshold θ for rejecting non-nearest neighbors, the searching time reduced 88.84% by the fast threshold-based nearest neighbor algorithm, and 91.53% by the extended fast threshold-based nearest neighbor algorithm, and it shows more stable result than the k nearest neighbor algorithms. The experimental result shows that users can take advantage of this system without delay even there are many user access at the same time. For future work, we will apply the fast nearest neighbor algorithms for protein binding site similarity search or other data set represented based on 3DZD.

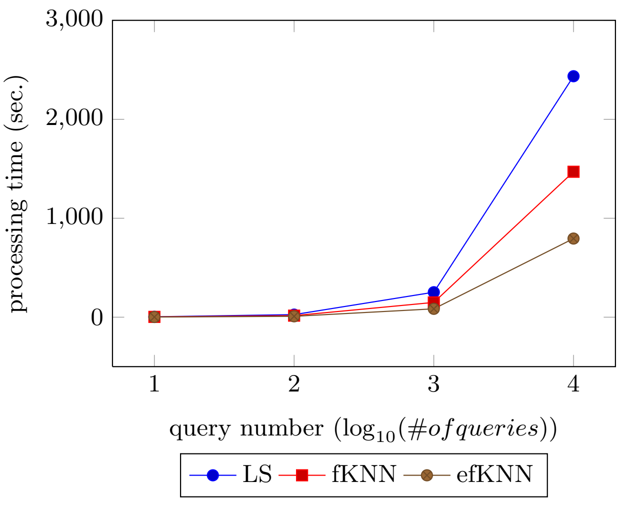


Figure 7: The simulation result of fKNNs as the number of query increases

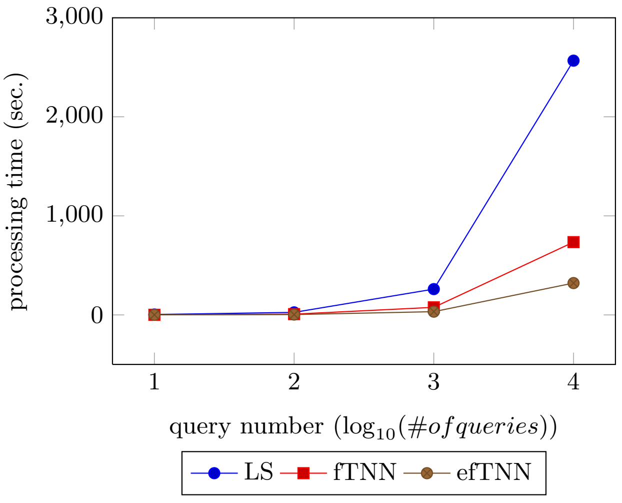


Figure 8: The simulation result of fTNNs as the number of query increases

6. REFERENCES

[1] T. Hawkins and D. Kihara, "Function prediction of uncharacterized proteins," *Bioinform. Comput. Biol.*, pp. 1–30, 2007.

[2] T. Hawkins, M. Chitale, and D. Kihara, "New paradigm in protein function prediction for large scale omics analysis," *Mol. Biosyst.*, pp. 223–31, 2008.

[3] J. Watson, R. Laskowski, and J. Thornton, "Predicting protein function from sequence and structural data," *Curr. Opi. Struct. Biol.*, pp. 377–385, 1996.

[4] R. Roberts, "Identifying protein function–a call for community action," *PLoS. Biol.*, 2004.

[5] D. L. Wild and M. A. S. Saqi, "Structural proteomics: Inferring function from protein structure," *Curr. Proteomics*, vol. 1, pp. 59–65, 2004.

[6] D. Zhi, M. Shatsky, and S. E. Brenner, "Alignment-free local structural search by writhe decomposition," *Bioinform.*, 2010.

[7] S. Lee, B. Li, D. La, Y. Fang, K. Ramani, R. Rustamov, and D. Kihara, "Fast protein tertiary structure retrieval based on global surface shape similarity," *Curr Opin Struct Biol*, pp. 393–398, 2006.

[8] Y. Hwang and H.-K. Ahn, "Convergent bounds on the euclidean distance," in *NIPS*, 2011.

[9] Y. Hwang, B. Han, and H.-K. Ahn, "A fast nearest neighbor search algorithm by nonlinear embedding," in *CVPR*, 2012.

[10] J.-F. Gibrat, T. Madej, and S. H. Bryant, "Surprising similarities in structure comparison," *Curr. Opi. Struct. Biol.*, pp. 377–385, 1996.

[11] I. N. Shindyalov and P. E. Bourne, "Protein structure alighment by incremental combinatorial extension (ce) of the optimal path," *Protein Eng.*, pp. 739–747, 1997.

[12] A. P. Singh and D. L. Brutlag, "Hierarchical protein structure superposition using both secondary structure and atomic representations," in *Intl. Syst. for Mol. Biol. (ISMB)*, pp. 1013–1022, 2008.

[13] L. Holm and C. Sander, "Protein structure comparison by alighment of distance matrices," *Mol. Biol.*, pp. 123–138, 1993.

[14] A. Martin, "The ups and downs of protein topology: rapid comparison of protein structure," *Protein Eng.*, pp. 829–837, 2000.

[15] K. Mizuguchi and N. Go, "Seeking significance in three-dimensional rotein structure comparisons," *Curr. Opin. Struct. Biol.*, pp. 377–382, 1995.

[16] R. Kolodny, D. Petrey, and B. Honig, "Protein structure comparison: implications for the nature of 'fold space', and structure and function prediction," *Curr. Opin. Struct. Biol.*, pp. 393–398, 2006.

[17] D. Kihara and J. Skolnick, "The pdb is a covering set of amall protein structures," *Mol. Biol.*, pp. 793–802, 2003.

[18] M. Gerstein and M. Levitt, "Using iterative dynamic programming to obtain accurate pairwise and multiple alignments of ptotein structures," in *Intl. Conf. Intl. Syst. Mol. Biol.*, pp. 393–398, 2006.

[19] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton, "CATH–a hierarchic classification of protein domain structures.," *Structure (London, England : 1993)*, vol. 5, no. 8, pp. 1093–1108, 1997.

[20] 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," *Mol. Biol.*, vol. 247, pp. 536–540, 1995.

[21] L. Holm and C. S, "Touring protein fold space with dali/fssp," *Nucleic Acids Res*, vol. 26, pp. 316–319, 1998.

[22] T. Madej, J. F. Gibrat, and S. H. Bryant, "Threading a database of protein cores.," *Proteins*, vol. 23, no. 3, pp. 356–369, 1995.

[23] K. Kinoshita and H. Nakamura, "Identification of protein biochemical functions by similarity search using the molecular surface database ef-site," *Protein Sci.*, 2003.

[24] Z. Aung, W. Fu, and K. lee Tan, "An efficient index-based protein structure database searching method," in *Intl. Conf. on Database Systems for Advanced Applications (DASFAA)*, pp. 311–318, 2003.

[25] L. L. Conte, S. E. Brenner, T. J. Hubbard, C. Chothia, and A. G. Murzin, "Scop database in 2002: refinements accommodate structural genomics," *Nucleic. Acids Res.*, pp. 316–319, 2002.

[26] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: an efficient access method for similarity search in metric spaces," in *Proc. Int. Conf. Very Large Databases (VLDB)*, 1997.

[27] D. Keim, "Tutorial on high-dimensional index structures: Database support for next decades applications," in *Proc. Int. Conf. Data Engineering (ICDE)*, 2000.

[28] N. Bruno, L. Gravano, and A. Marian, "Evaluating top-k queries over web-accessible databases," in *Proc. Int. Conf. Data Engineering (ICDE)*, 2002.

[29] V. Venkatraman, P. R. R. Chakravarthy, and D. Kihara, "Application of 3D Zernike descriptors to shape-based ligand similarity searching.," *cheminformatics*, vol. 1, 2009.

[30] N. Canterakis, "3d zernike moments and zernike affine invariants for 3d image analysis and recognition," in *11th Scandinavian Conf. on Image Analysis*, pp. 85–93, 1999.

[31] M. Novotni and R. Klein, "3d zernike descriptors for content based shape retrieval," in *8th ACM symp. on Solid modeling and applications*, SM '03, pp. 216–225, 2003.

[32] D. La, J. Esquivel-Rodríguez, V. Venkatraman, B. Li, L. Sael, S. Ueng, S. Ahrendt, and D. Kihara, "3d-surfer: software for high-throughput protein surface comparison and analysis," *Bioinformatics*, vol. 25, no. 21, pp. 2843–2844, 2009.

[33] M. L. Connolly, "The molecular surface package," *Mol. Graphics*, vol. 11, pp. 139–141, June 1993.