Indexing Methods for Efficient Protein 3D Surface Search

Sungchul Kim
POSTECH
subright@postech.ac.kr

Lee Sael
The State University of New York Korea
sael@sunykorea.ac.kr

Hwanjo Yu
POSTECH
hwanjoyu@postech.ac.kr

ABSTRACT

This paper exploits efficient indexing techniques for protein structure search where protein structures are represented as vectors by 3D-Zernike Descriptor (3DZD). 3DZD compactly represents a surface shape of protein tertiary structure as a vector, and the simplified representation accelerates the structural search. However, further speed up is needed to address the scenarios where multiple users access the database simultaneously. We address this need for further speed up in protein structural search by exploiting two indexing techniques, i.e., iDistance and iKernel, on the 3DZDs. The results show that both iDistance and iKernel significantly enhance the searching speed. In addition, we introduce an extended approach for protein structure search based on indexing techniques that uses the 3DZD characteristic. In the extended approach, index structure is constructured using only the first few of the numbers in the 3DZDs. To find the top-k similar structures, first top-$10 \times k$ similar structure is selected using the reduced index structure, then top-k structures are selected using similarity measure of full 3DZDs of the selected structures. Using the indexing techniques, the searching time reduced 69.6% using iDistance, 77% using iKernel, 77.4% using extended iDistance, and 87.9% using extended iKernel method.

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

Algorithms, Theory

Keywords

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

DTMBIO'12, October 29, 2012, Maui, Hawaii, USA.

1. INTRODUCTION

The size of protein structure database such as the Protein Data Bank (PDB) continues to grow. PDB had around 1000 structures in 1992, but now it stores over 150,000 structures. In addition, the number of proteins with unknown functions is increasing due to efforts in structural genomics projects. Knowing the functions of proteins is crucial to many studies of biological processes. Especially, researchers need to know the key proteins that play an important role for severe deseases, and it is directly related to human life. Therefore, assigning functions to novel proteins is one of the most significant problems in proteomic study, and several methods have been developed to assign functions to an unknown protein. Basically, the function of a protein can be identified by searching amino acid sequence database for similar sequences that the functions are already known. However, the 3D structures of proteins are more conserved than the sequences and using the structural informaion provide more reliable similarity measures.

Many methods have been introduced for pair-wise protein structural search. They align the two structures and 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 to search structures against large database, since it has high computational complexity. Sael et al. introduced a new approach for fast protein surface similarity search using 3DZDs [18]. This approach does not consider individual residue/atom positions, or the arrangement of the secondary structure segments. 3DZD has three advantages: 1) fast k-nearest neighbor search, 2) rotational invariance, and 3) easy adjustment of the resolution of the structural representation resolution. In particular, using 3DZDs, it is possible to retrieve similar proteins in seconds among 150k protein structures. However, few seconds are still too long for a real time search system. Response time increases further when multiple protein structure search requests are processed simultaneously. To enhance the searching speed, using indexing technique could be a good solution [7, 27]. Therefore, this paper exploits indexing techniques on 3DZDs in order to speed up protein structure search. Specifically, we apply two indexing techniques, iDistance and iKernel, on 3D-Surfer data set. We extend top-k protein structure search algorithm based on the indexes and 3DZD characteristic. The experimental results show that the indexing techniques both decrease the protein structure search speed (iKernel-based indexing works better than iDistance, which is the state of the art method in a typical database

indexing problems), and our top-k search algorithm further speed up the protein structure search.

This paper is organized as follows. We briefly introduce related works about protein structure search and top-k query search. Then, we explain iDistance, iKernel and the extended top-k query search method in combination with iDistance and iKernel. Finally, we provide experimental results to verify the efficiency of our approaches, and conclusion with future works.

2. RELATED WORK

Protein structure: 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.

Protein Structure Search: Earlier structural similarity measurements were designed for pair-wise analysis where the user only needed to compare handful of protein structures [9, 28, 29]. However, as the number of known structures increased more methods are proposed for similarity search in protein database [11, 21]. One of the most intuitive approaches is to compare the coordinates of corresponding residues or atoms of proteins after structural alignment [22, 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 [28, 14, 8].

There are major structure databases such as PDB, CATH [26], and SCOP [23] 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 [10], Vector Alignment Search Tool (VAST) search [20], and eF-site database [15]. Given a query protein structure, they need around an hour to finish searching their databases. Zeyar et al. suggests an indexing method called ProtDex for fast search in 3D protein structure database [1]. Although it performs faster than DaliLite [11], one of the most popular protein structure search algorithms, the search time of ProtDex takes over a few minutes and it is not practical for online database searches.

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 k-nearest neighbor (k-nn) search of protein structures. It has been verified that the retrieved k-nn proteins by 3D-Surfer have similar functional and evolutional information in terms of SCOP classification [6]. Some of the characteristics of the 3DZDs is that it is rotational invariant, and the resolution of the representation of protein structures are easily adjusted by changing the order, and descriptors of the lower order are contained in the descriptors of the higher order.

K-nn search algorithm: There is a long stream of researches on finding top-k nearest neighbor (k-nn) search problem which is an optimization problem for finding closest points in metric spaces. The simplest method is to compute the distance from the query point to every other point in the database. It has $O(Nd)$ complexity where N is the number of data points and d is the dimensionality of the data, and 3D-Surfer also used this approach. For efficient top-k search, there have been various methods via space partitioning including X-tree, TV-tree, and SR-trees [4, 13, 2]. iDistance that we used here is also space partitioning method. There are other methods such as iKernel which is an indexing technique and designed for efficient calculation of support vector machine (SVM). The details of iDistance and iKernel are described in the methods. Note that those methods cannot be directly used for protein structure data, thus in this work we exploit the 3DZD of protein structures and apply indexing techniques on the 3DZDs.

3. METHODS

In this section, we first introduce the protein structure dataset and their 3D-Zernike Descriptor (3DZD). Then, the descriptions of iDistance and iKernel methods and the proposed efficient top-k query search method based on the characteristics of 3DZD are provided.

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 [18] and ligand structure analyses [30] as well. We provide brief description of 3DZD for reader's convenience. Detailed description can be found in [3, 25].

The 3DZD descriptors for protein structural dataset of 158781 number of protein chain structures was obtained through 3D-Surfer database. The entire structures in PDB was collected and processed on 2009 [17]. 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 [5], and then voxelized. Each of the voxelized protein surface were used as a 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:

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

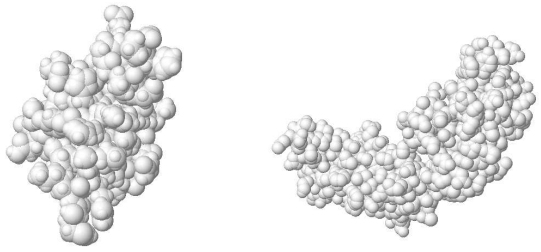
where $-l < m < l, 0 \leq l \leq n$, $(n-l)$ is even, $Y_l^m(\vartheta,\varphi)$ are spherical harmonics, and R_{nl} are radial functions constructed to convert $Z_{nl}^m(r,\vartheta,\varphi)$ to polynomials in the Cartesian coordinates, $Z_{nl}^m(x)$. To obtain the 3DZD of $f(x)$, 3D Zernike moments 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 ratational invariance to the desriptor. 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).



(a) 2kr1-A (b) 3DVG-A

Figure 1: Two example proteins; 2kr1-A and 3DVG-A

Fig. 1 is an example of 3DZD of two proteins, triosephosphate isomerase (PDB code: 2kr1-A) and interleukin-4 receptor α-chain (PDB code: 3DVG-A). As you can see, two proteins have different structures overall and their descriptor also shows visible difference (Fig. 2).

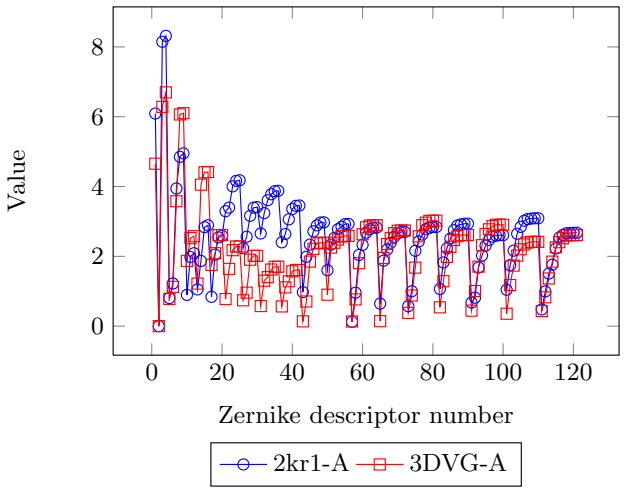


Figure 2: 3D Zernike descriptors of two example proteins

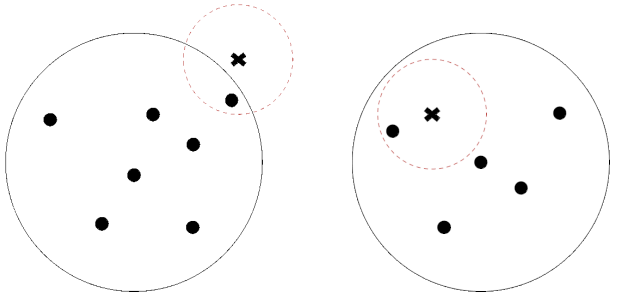
3.2 Indexing Techniques

In this work, we exploit two indexing techniques: iDistance and iKernel. Two indexing techniques partition given data points into clusters and using the clusters to find k-nn. Note that both techniques exactly retrieve k nearest neighbor results given a query. The details of the two indexing methods follows.

3.2.1 *iDistance*

iDistance is an efficient indexing technique for k-nearest neighbor search in a high-dimensional metric space [12]. It depends on how data are partitioned and how reference points for each partition are defined (we will henceforth mention partition as cluster for terminology consistency between iDistance and iKernel). After clustering and reference point selection, each data point is indexed according to the distance between its reference points.

To build an index, the reference points are selected by data clustering. Although various clustering techniques can be used to select reference points, we have used k-means clustering. And then, data points are assigned its closest reference point. During the assignment process, the data point is recorded with the distance which is called iDistance and is used as a key for top-k search. The iDistance is computed as follows:



(a) Query is out of the cluster (b) Query is in the cluster

Figure 3: Top-k search using iDistance (x mark is query; circle mark is data point; circle with solid line is cluster; circle with dashed line is query region)

$$y = i \times C + dist(p, O_i) \tag{4}$$

where y is iDistance of point p in i-th cluster, O_i, and C is a constant used to stretch the data ranges of indexes.

To retrieve top-k results, we visit the clusters to check whether the cluster can have nn or not. The radius r that indicates query region defined as the range from the query, and r increases by Δr to form a larger query region after iterations. When the query region is overlapped with certain cluster, we notice that the cluster will have nearest points. Therefore, at each iteration, we first check whether the target cluster C_i can have nn of query q by comparing the distance from q to the reference point of C_i, and the farthest distance in C_i . If the area of C_i overlaps with the query region (Fig. 3-(a)), it indicates that C_i can have nn. Therefore, we check the data points in the cluster to find the nearest points from the outermost position of the cluster. If q is located in C_i (Fig. 3-(b)), it also indicates that C_i have nn. In this case, we need to search the cluster inward and outward from the position of q.

3.2.2 *iKernel*

iKernel is originally designed for the efficient learning of support vector machine (SVM) [31]. However, it is also applicable for top-k search for Euclidean distance. Similar to iDistance, it first divide given data points to clusters where the clusters have set of rings which are the data structure defined for iKernel. Given a query, it searches k-nn by visiting each cluster and its rings.

To build an index, given data points are clustered into m clusters and centroids of each clusters are computed in the feature space. Various clustering techniques can be applied. Then, based on those centroids and clusters, we can build an index by assigning data points into a set of rings in clusters

as follows.

$$
\begin{aligned}
C_1 \quad & : \{C_{1,1} : x_{1,1}^{(1)}, x_{1,1}^{(2)}, \cdots, x_{1.1}^{(g)}\}, \\
& \quad \{C_{1,2} : x_{1,2}^{(1)}, x_{1,2}^{(2)}, \cdots, x_{1,2}^{(g)}\}, \cdots \\
C_2 \quad & : \{C_{2,1} : x_{2,1}^{(1)}, x_{2,1}^{(2)}, \cdots, x_{2.1}^{(g)}\}, \\
& \quad \{C_{2,2} : x_{2,2}^{(1)}, x_{2,2}^{(2)}, \cdots, x_{2,2}^{(g)}\}, \cdots \\
& \cdots
\end{aligned} \tag{5}
$$

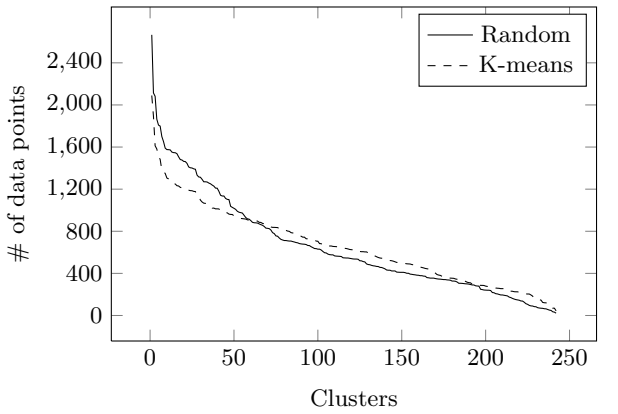


Figure 4: The number of data instances assigned to clusters by random and k-means clustering

Note that each ring have g number of data points. The paramter g is user adjustable and need to be determined prior to index construction.

To process a k-nn of a query structure, we exploits the index and Minimal Possible Distance (MPD) [31]. MPD is the minimal possible distance between a query q and ring structure $C_{i,j}$. With this new notion of the MPD, k-nn search works as follows. Given a query point q, we first initialize a priority queue Q with a set of pair $<C_{i,j}$, MPD$>$ of each cluster in the ascending order of their MPDs between the q where only the outermost ring is considered first. Then, at each iteration, the top entry of Q is popped. If a ring is popped, the data points in the ring are inserted to Q with the distance from q, and if the popped item is a data point, it is simply added into top-k result since the priority queue ensures that all instances in the queue have larger distances from q and also all rings have larger MPDs between q.

3.2.3 *Partitioning*

For both indexing techniques, we require division of data set into clusters by assigning similar number of data points to each cluster. First method randomly selects reference points and assign remaining data points to their closest reference point. K-means clustering, which is one of the most widely used clustering methods, is also tested. The goal of K-means clustering algorithm is to divide a set of points into k clusters so that the within-cluster sum of squares is minimized [19]. K-means algorithm is easily applicable to problems and performance is often shown to be satisfying. However, it also has some disadvantages as the K-means algorithms is a local search procedure and it suffers from the serious drawback that its performance depends on the initial starting conditions [24]. Therefore, in this work, we repeatedly cluster data points and conduct experiments, and select the best result.

Fig. 4 shows that the number of data points assigned to clusters. As you can see, using K-means generates more stable clusters with similar amount of data points. If some clusters have larger number of data points than other clusters, we need to search their region more than other clusters. Therefore, balancing the number of data points in clusters is required for efficient k-nn search. In experiment, we compare those partitioning strategies in terms of search speed.

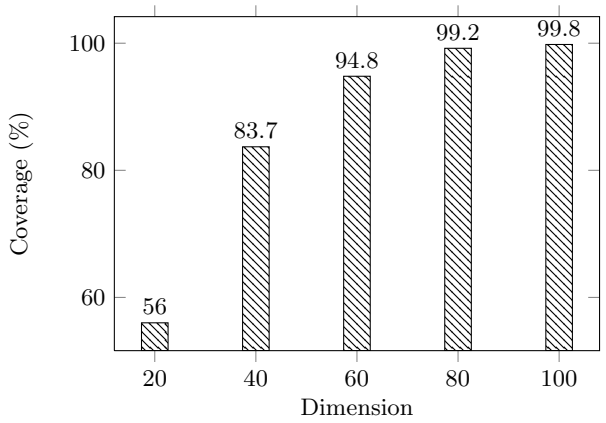


Figure 5: The coverage of top-25 in top-250 as the dimension used for index increases (M is 866, $< \Delta r, C >$ is <0.2, 4>, and g is 50)

3.3 Extended top-k search based on 3DZD

According to 3DZD, the descriptor vector has one notable characteristic. The prior dimensions in the descriptor vector indicate global shape and the posterior dimensions include more specific shape information. Based on this fact, we propose an extended top-k search approach for protein structure based on 3DZD. In this approach, the index is constructed based on the data set using only the prior half of original vectors. Retrieval result of top-250 using first 60 invariants in the descriptor vectors covered 94.8% of the top-25 retrieval result using the full descriptors as shows in Fig. 5. In addition, finding top-$k \times 10$ result using half dimension takes less time than using basic indexing techniques (Fig. 6) where iD-s and iK-s is the result of top-$k \times 10$ for iDistance and iKernel using small dimensions, respectively, and iD-121 and iK-121 is the result of top-k for iDistance and iKernel using original 121 dimensions. This shows that using half of the descriptor for indexing allows a fast and accurate approximation of using the full descriptor.

Based on this observation, we introduce a new approach for top-k search as follows.

1 Given a query protein Q, search top-$k \times 10$ result using the indexing structure with 60-dim (the half of entire dimension).

2 Using the top-$k \times 10$ results, find exact top-k result.

Note that $k \times 10$ is very small number compared to the size of the database (around 1.6 million).

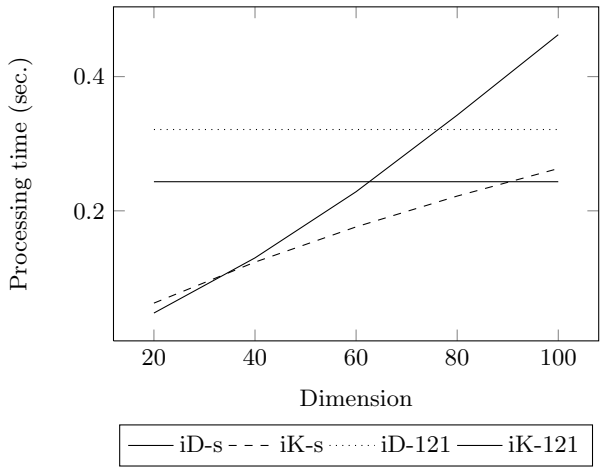


Figure 6: The change of processing time as the dimension used for index increases (M is 866, $< \Delta r, C >$ is <0.2, 4>, and g is 50)

4. RESULTS

In this section, we verify the effectiveness of indexing techniques on top-k search of protein structures. Sael et al. showed that 3DZD works well on finding similar proteins in terms of functional and evolutionary characteristics based on SCOP classification [18]. The SCOP provides the ordering of all proteins of known structure according to their evolutionary and structural relationships. In addition, both of iDistance and iKernel are not approximate techniques and find exact top-k nn from database according to the structural similarity described by 3DZD. Therefore, we only measure the efficiency in terms of processing time and evaluation ratio. The evaluation ratio is computed as the fraction of accessed data points over the number of database (1 for linear scan since it access all data points in the data set). 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 consistent measure.

The experiments were conducted on the machine, Intel Core(TM) i7 CPU (3.40GHz), and 16 GB memory. In overall experiment, we used 100 data points that are randomly selected from data set, and averaged entire processing time and evaluation ratio.

4.1 The user parameters

There are a few parameters that are needed to be optimized in the iDistance and iKernel methods. In this section, we observe how the result varies as the user parameters varies to select the best. We also observe how the cluster number affects the top-k search. We vary the partition data points using different number of clusters: 121, 242, 498, and 866. 121 is the dimensionality of data set, and 242 is the two times the dimensionality ([12] refers that this way works well on iDistance). And the others are according to SCOP classification hierarchy. 498 is the number of families, and 866 is the number of protein domains [23]. We assume that the numbers defined by domain experts could have good evidence of cluster number. The followings include the explanation of user parameters and their experimental result.

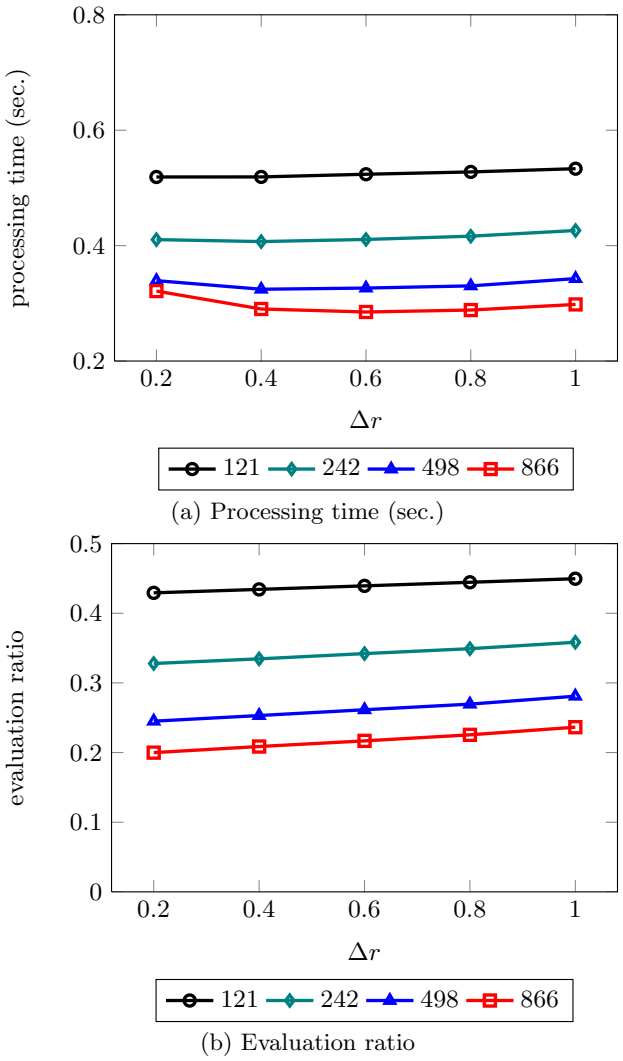
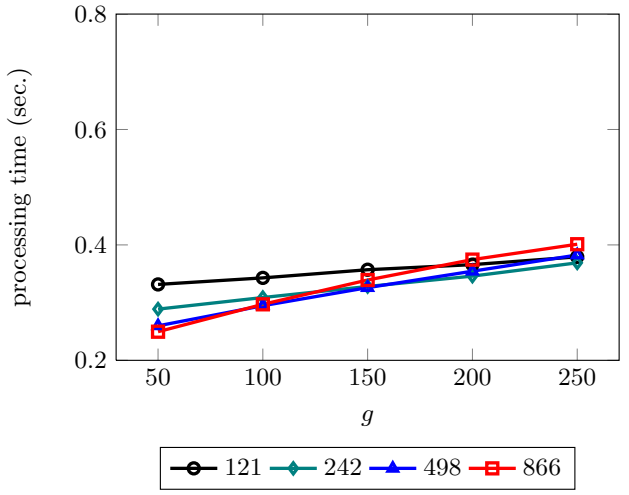


Figure 7: The efficiency of top-k search using iDistance with various Δr; the legend indicates the cluster number.

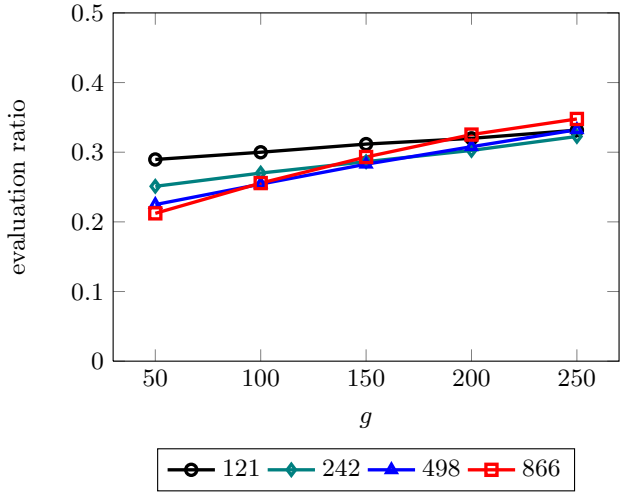
There are two user paramters: Δr and C in iDistance. r is the distance radius of query region that indicates an area that we need to search, and Δr is the amount of value added to r after each iteration, and C is used to obtain key value for index construction. Although we have conducted some experiments to tune C as well, it seems not affect much on top-k search. Therefore, in this work, we set C as 4 by maintaining few of data instances are overlapped between clusters as [12] did. Fig. 7 shows the changes of processing time and evaluation ratio as Δr increases. The results show that smaller Δr and larger number of clusters generally performs better. It indicates that iDistance depends highly on the number of clusters. In contrast to statement made by Jagadish et al. that using two times the dimension of data as the number of clusters often works well, the result show that when the data size is very large, large number of clusters is needed as well. Therefore, different from Jagadish's work [12], it is likely that when the data size is very large, we have to use large cluster number as well.

For the optimized value, we decided to use 0.2 as Δr and

866 as the number of clusters, since it shows the best result in terms of the evaluation ratio. Though it does not result out the best result in terms of processing time, the difference among Δr is not that large compared to the difference among the dimensions.



(a) Processing time (sec.)



(b) Evaluation ratio

Figure 8: The efficiency of top-k search using iKernel with various ring size; the legend indicates the cluster number.

There is one parameter, g, in iKernel. The parameter g is the number of data points in rings of the clusters. As mentioned before, for top-k search, we visit the rings according to its MPD and add all data instances of the ring into the priority queue, which is sorted at the end of every iteration. Fig. 8 shows that the best performance is obtained using the 50 when the number of cluster is 866. The result also seems that when the number of cluster is small, the amount of changes as g varies becomes small. When the number of clusters is small, the number of rings decreases as well, so that the result affects by g less than when the number of clusters is large.

For the optimized value, we decided to use 50 as g and 866 as the number of clusters, since it shows the best result in terms of the evaluation ratio as well as the processing time.

Table 1: The effectiveness of clustering

	iDistance	iKernel
Random	0.34124	0.26173
K-means	**0.3237**	**0.246**

(a) Processing time (sec.)

	iDistance	iKernel
Random	0.2132	0.2251
K-means	**0.2**	**0.2122**

(b) Evaluation ratio

4.2 The comparison of clustering techniques

First, we compare the performance of the two clustering approaches: Random clustering and K-means clustering algorithm. In addition to normal clustering requirement of high inter-distance and low intra-distance between the clusters, for efficient indexing purposes, we require that the size of clusters are balanced, that is clusters should have similar number of data points. Intuitively, if some clusters have more data points than others, search time for those clusters will be high. Table 1 shows that indexing techniques with k-means clustering, although slightly slower, have better evaluation ratio than random clustering. Therefore, the following results in later sections use the result by the indexing techniques with k-means algorithm.

4.3 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 the performance. As expected, the processing time increases (Fig. 9). However, the increase processing time is less for iKernel than iDistance as k increases, and iKernel shows works faster than overall. In terms of the evaluation ratio, the result is different. Though the difference is small, iDistance shows better result than iKernel. It indicates that when k becomes large, iKernel needs to search more data points than iDistance since it adds all of data points in visited rings computing the distance between the data point and query. However, the overall processing cost is less for iKernel.

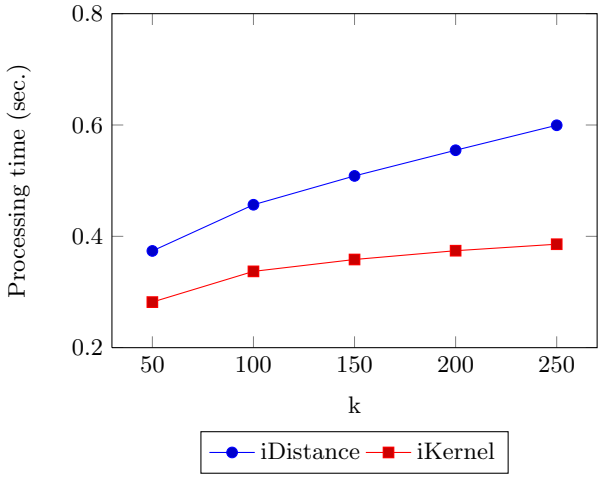
4.4 The enhancement with the extended top-k search based on 3DZDs

In this section, the result of the extended top-k search based on 3DZDs is shown compared to the best results of the basic approach discussed in the previous section. Since we use different number of k for the extended approach, we need to tune the parameters again. In the table, the number in bracket is the ratio of actual top-25 result in top-25 result which are approximately obtained by the extended approach (which is same to the preliminary result, Fig 5). As you can see, even the result of the extended approach extract around 95% of actual top-25 result where the standard deviation is 0.01.

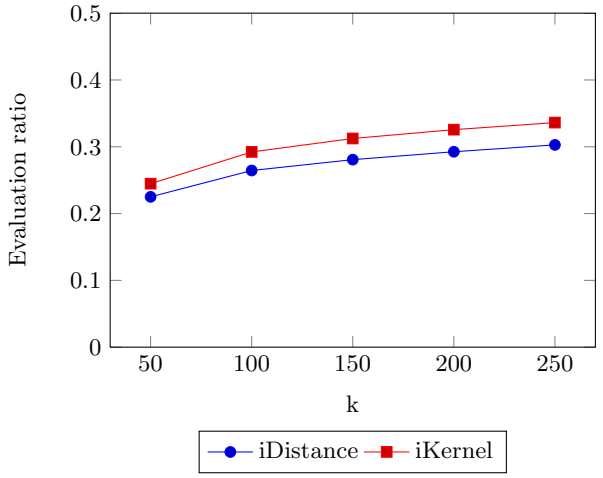
Although, the enhancement of iDistance and iKernel with basic top-k search is not that large, the extended approaches work much faster than basic approaches. It shows that the extended approach further speed up top-k query search on the protein structure data set.

Table 2: The comparison of top-k search using the extemded top-k search (Proc. is processing time measured in second and Eval. is evaluation ratio)

	LS	iDistance		iKernel	
		basic	ext.	basic	ext.
Proc.	1.0567	0.3212	0.2387 (95.2)	0.2434	**0.128 (95.2)**
Eval.	1	0.3	0.202 (95.2)	**0.1761**	0.1993 (95.2)



(a) Processing time (sec.)



(b) Evaluation ratio

Figure 9: The results as k increases

4.5 Simulation result

To support our statement more, we simulate the senario that a number of users enter queries at the same time via multi-threading. According to the result, the extended approach always works faster than the basic approach. Specifically, iKernel shows the better result than iDistance in terms of processing time, but not evaluation ratio. Note that when they access to data instance to compute inner products of vectors, in the case of basic approach, there are 121-dimensional vectors. However, in the case of the extended approach, the inner product takes 60-dimensional vectors. It indicates that if the difference is small between two approaches, the extended approach may work better than the basic approach in real. In addition, when the number of query is small, the quality is comparable. However, when the number of query becomes large, the difference of processing time becomes larger as well.

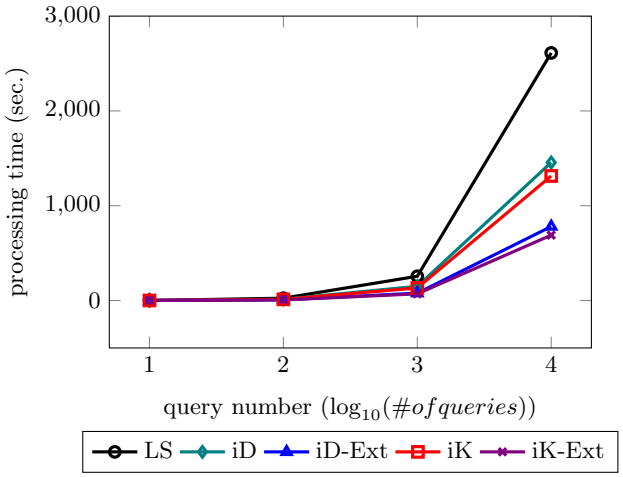


Figure 10: The simulation result as the number of query increases

5. CONCLUSION

In this paper, we introduce an efficient indexing for protein structure search where protein structures are represented as vectors by 3D-Zernike Descriptor (3DZD). Using indexing techniques alone, we were able to make the search speed 77% faster compared to the prevoius version of 3D-Surfer that uses linear Euclidian distance scan between the 3DZDs in the database. We also proposed an extended version of the protein structure search based on the key observation that the prior dimension of the descriptor indicates global shape of the protein structure. Using the extended techniques it is improved up to 87.9%. It indicates that users can take advantage of this system without delay even there are many simultaneous user access. For future work, we will improve the top-k search with indexing techniques by utilizing the characteristics of the query prior to searching. In addition, we will apply indexing techniques for protein binding site similarity search with other data set represented based on 3DZD as well.

6. ACKNOWLEDGMENTS

I wish to express thank to Ilhwan Ko who is the author of [31] for helping me a lot to exploit iKernel approach for this work. This work was also supported by IT Consilience Creative Program of MKE and NIPA (C1515-1121-0003), and Next-Generation Information Computing Development Program through the National Research Foundation of Ko-

rea(NRF) funded by the Ministry of Education, Science and Technology (2012M3C4A7033344)

7. REFERENCES

[1] 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)*, pages 311–318, 2003.

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

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

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

[5] M. L. Connolly. The molecular surface package. *Journal of Molecular Graphics*, 11(2):139–141, June 1993.

[6] 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.*, pages 316–319, 2002.

[7] K. Deng, X. Zhou, H. T. Shen, Q. Liu, K. Xu, and X. Lin. A multi-resolution surface distance model for k-nn query processing. *The VLDB Journal*, 17(5):1101–1119, 2008.

[8] M. Gerstein and M. Levitt. Using iterative dynamic programming to obtain accurate pairwise and multiple alignments of ptotein structures. In *Proc. Int. Conf. Intl. Syst. Mol. Biol.*, pages 393–398, 2006.

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

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

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

[12] H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang. idistance: An adaptive b-tree based indexing method for nearest neighbor search. *ACM Trans. Database Syst.*, pages 364–397, 2005.

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

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

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

[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.*, pages 393–398, 2006.

[17] 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*, 25(21):2843–2844, 2009.

[18] 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*, pages 393–398, 2006.

[19] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. on Math. statist. and Prob., pages 281–297, 1967.

[20] T. Madej, J. F. Gibrat, and S. H. Bryant. Threading a database of protein cores. *Proteins*, 23(3):356–369, 1995.

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

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

[23] 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.*, 247:536–540, 1995.

[24] J. P. nad J. Lozano and P. Larranaga. An empirical comparison of four initializatoin methods for the k-means algorithm. pages 393–398, 1999.

[25] M. Novotni and R. Klein. 3d zernike descriptors for content based shape retrieval. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, SM '03, pages 216–225, 2003.

[26] 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)*, 5(8):1093–1108, 1997.

[27] H. T. Shen, Z. Huang, J. Cao, and X. Zhou. High-dimensional indexing with oriented cluster representation for multimedia databases. ICME'09, pages 1628–1631, 2009.

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

[29] 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)*, pages 1013–1022, 2008.

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

[31] H. Yu, I. Ko, Y. Kim, S. Hwang, and W.-S. Han. Exact indexing for support vector machines. In *Proc. int. conf. on Management of data*.