Localized Co-occurrence Model for Fast Approximate Search in 3D Structure Databases

Zi Huang, Heng Tao Shen, Xiaofang Zhou

Abstract

Similarity search for 3D structure data sets is fundamental to many database applications such as molecular biology, image registration and computer aided design. Identifying the common 3D subtructures between two objects is an important research problem. However, it is well known that computing structural similarity is very expensive due to high exponential time complexity of structure similarity measures. As the structure databases keep growing rapidly, real-time search from large structure databases becomes problematic. In this paper, we present a novel statistical model, multi-resolution *Localized Co-occurrence Model* (LCM), to approximately measure the similarity between the two point-based 3D structures in linear time complexity for fast retrieval. LCM could capture both distribution characteristics and spatial structure of 3D data by localizing the point co-occurrence relationship within a predefined neighborhood system. As a step further, a novel structure query processing method called *iBound* is also proposed to speed up the search process. iBound avoids a large amount of expensive computation at higher resolution LCMs. By superposing two LCMs, their largest common substructure can also be found quickly. Finally, our experiment results prove the effectiveness and efficiency of our methods.

Zi Huang is with School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane QLD 4072 Australia. Email: huang@itee.uq.edu.au. Tel: 61-733653476. Fax: 61-733654999.

Heng Tao Shen is with School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane QLD 4072 Australia. Email: shenht@itee.uq.edu.au. Tel: 61-733658359. Fax: 61-733654999.

Xiaofang Zhou is with School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane QLD 4072 Australia. Email: zxf@itee.uq.edu.au. Tel: 61-733653248. Fax: 61-733654999.

I. Introduction

Research in 3D structure data has a great impact on many applications, such as molecular biology, image registration and computer aided design (CAD). Structure similarity search is a fundamental operation for 3D structure databases, since data with similar (sub)structure may often have similar functions [1], [2], [3], [4]. For example, in protein studies, 3D protein structures with similar spatial relationships potentially have similar biological functions. For CAD applications, the goal is to reduce the cost of producing new spare parts by maximizing the reuse of old parts. With the fast increasing size of data set, there are strong demands for developing efficient techniques for retrieving similar (sub)structures from large structure databases.

In this paper, we focus on the point-based structure similarity search. A 3D structure P is defined as $P = \{p_i | i = 1..m\}$, which is a set of m points in the 3D Euclidian space, where m is a positive and finite integer. Different structures may have different numbers of points. For example, a protein can be viewed as a set of points in 3D space where each point represents an atom of the protein. Note that, we only concern the location information of each atom and ignore the atom's type.

For any two structures P_1 and P_2, they are identical if these two structures are *isometric*. That is, P_1 and P_2 are isometric if there is an bijection f between P_1 and P_2 such that for any two points p_i and p_j from P_1, $d(p_i, p_j) = d(f(p_i), f(p_j))$, where $d(.,.)$ denotes the Euclidean metric. Intuitively, this definition implies that two structures are identical if there exists a rigid transformation under which P_1 becomes congruent to P_2 (i.e., share same number of points and keep all their internal distances unchanged during the transformation) [5]. Figure 1 shows an example in 3D space that structure P_1 and P_2 are identical.

It is impractical to search identical structures in the real applications. The structure similarity search is being concerned widely. By following the definition above, two structures are regarded as similar if $|d(p_i, p_j) - d(f(p_i), f(p_j))| \leq \epsilon$, where ϵ is the distance threshold specified by users. We also call it as ϵ-congruence of two structures.

For any two subsets P'_1 and P'_2, where $P'_1 \subseteq P_1$, $P'_2 \subseteq P_2$ and $|P'_1| = |P'_2|$, we can say P_1 and P_2 share a $|P_1'|$(or $|P_2'|$)-sized identical/similar substructure, if P_1' is identical/similar to P_2'. Therefore, the problem to find the largest similar substructure between two structures is equal to the problem to find the largest common point set between two point sets under ϵ-congruence [6].

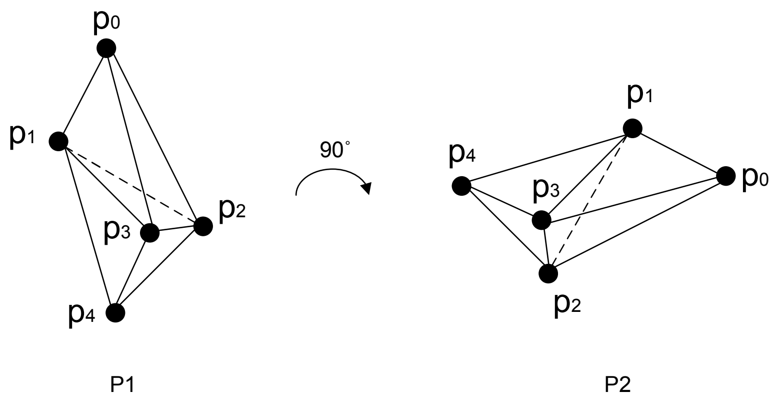


Fig. 1. An example of identical structures

Techniques of efficient search for similar 3D (sub)structures have been studied in various areas. The common requirement is that structures should be invariant to rotation and translation. This generally leads to high exponential time complexity in computing the similarity of two structures.

Graph theoretical approaches are widely used to find the maximal similar (sub)structure by modelling each structure as a graph [4], [7] and detecting similar sub-graphs among the graphs. The problem is often transformed to the clique problem. As the clique detection problem is NP-hard, many heuristic algorithms are developed. Most existing heuristic algorithms for the clique problem are partially enumerative and branch-and-bound based [8]. However, they are not scalable to large scale databases due to high exponential time complexity.

Geometric hashing [9] is also widely used to find the common point sets between two structures. It defines a coordinate system for each structure. The similarity between two 3D structures is calculated by comparing two frame systems. The time complexity of computing structure similarity is $O(n^8)$, where n is the number of points in a structure. Obviously, such an expensive measure cannot be applied for practical usages on large databases.

Based on the techniques of clique detection and geometric hashing, *k-clique hashing* has recently been developed [10]. It aims to combine the above two approaches to speed up structures similarity search. However, it can only be applied on the small sized structures since decomposing the product graph of two structures (e.g., a query structure and a database structure) into k-sized cliques is enumerative in nature. Generally, the number of points in a structure in their experiments is no more than 30. It is often that a protein structure contains hundreds of points.

Moreover, the sizes of 3D structure databases keep increasing quickly. Take Protein Data Bank (PDB) as an example. The number of total searchable structures maintained by PDB is increasing in an exponential rate, from 3,835 in 1995 to 43,339 as of May 08, 2007 [11]. Obviously, examining all structures in a large database with an expensive similarity measure is computationally prohibitive. Real-time similarity search from very large 3D structure databases faces great challenges.

In this paper, we present a novel statistical model for 3D structures and approximate the structure similarity by the corresponding statistical similarity in linear time complexity. The main contributions are summarized below.

- We develop a novel statistical model called *Localized Co-occurrence Model* (LCM) which captures the statistical characteristics of point distribution as well as the spatial structure of points in a point set by localizing the co-occurrence relationship within a predefined neighborhood system. Interestingly, LCM is invariant to rotation and translation. The time complexity of matching two LCMs is linear.
- A multi-resolution representation of LCM is also introduced. Higher resolution converges LCM to get closer to the intrinsic characteristics of point distribution and can be constrained by low resolution.
- We apply our multi-resolution LCM to model 3D structures. By the property of LCM, the structural characteristics of a 3D structure can be captured by a *Local Joint Probability Distribution* (LJPD). Since LCM is invariant to rotation and translation, the similarity of two structures can be computed based on their LJPDs efficiently in linear time complexity.
- To further improve the efficiency, a novel structure query processing method called *iBound* (incremental and Bounded search) is proposed to reduce the computational cost. By superposing two LCMs, their

largest common substructure can also be found quickly.

- We evaluate our new techniques by extensive experiments. Comparing with the recent *k-clique hashing* method, our experiment results show that our methods achieve real-time response with satisfactory accuracy.

The rest of the paper is organized as follows. Some related work is reviewed in Section II. The general LCM, multi-resolution LCM and their properties are introduced in Section III. In Section IV, we see how a 3D structure is modelled by multi-resolution LCM. iBound is presented in Section V. The experimental results are reported in Section VI. We conclude our paper in Section VII.

II. Related Work

Many new database applications, such as in the areas of molecular biology [5], biochemistry [4], 3D image registration [12] and computer aided design (CAD) [13], need to deal with a very large collection of 3D structures. A fundamental operation in such applications is to find structures from a database that are similar to a query structure, i.e., 3D structure similarity search.

A. Structure Representation

A variety of features have been proposed to represent a 3D structure which are discussed in [14] for different applications. Typically, features contain the basic properties of a 3D structure such as the number of vertices, surface area, volume and so on [15], [16], [17], [18], [19]. The volume-based features are captured from volumetric representations obtained by dividing object surface into voxel grids [20], [19], [21], [13], [22]. On the other hand, the features of surface geometry focus on characteristics derived from 3D structure model surfaces [23]. Some other features (e.g., extension-based one [22]) have also been studied in different applications. Depending on the different applications and models, various features can be used to capture most important geometric information of 3D structures.

In this paper, we focus on the *point set representation*, which implies that a 3D structure is represented as a set of 3D points. We take protein structure as an representative example.

As we know, a protein is constructed by a set of atoms. A simple way to represent a 3D protein structure is to arrange protein on an imaginary Cartesian coordinate frame and assign (x, y, z) coordinates to each atom. By considering each atom as a 3D points, a protein structure is a set of 3D points. The Protein Data Bank (PDB) is a representative of this kind of representation.

Protein structure similarity search is not concerned about the exact positions of points (atoms) in the 3D space, but the relative spatial relationship among the points. We consider a structure as rigid, which will not be affected by Euclidean transformation. This property can be illustrated by Fig.1, where P_1 shows a 3D structure and P_2 shows the same structure after $90°$ rotation. The similarity of a structure to the query is established by taking into account all the possible Euclidean transformations (i.e., rotations and translations) of the structure.

Derived from the point set representation, another widely used way of representing a structure to keep its rigidity is a matrix of all inter-point distances [3], [24], [5]. Obviously the dimensionality of such a data space is $D = n^2$, where n is the number of points in an object. However, such representation carries a lot of redundant information, as a structure may still be rigid even when some of the distances are removed.

B. Structure Similarity Search

The problem of finding similar 3D structures from a database is gaining its importance [1], [2], [3], [4]. Beside global structure matching, one may also show interests on how many points are superposed between two structures, which can be generalized as the problem of finding largest common substructure. It has been extensively studied in computational geometry as the *largest common point set* problem (LCP) [6], [25], [26] which finds a subset of one point set with the maximum cardinality congruent to some subset of the other point set. However, as it is unreasonable to expect an exact match between positions, two points are considered to be matched if their distance is less than a predefined tolerance threshold ϵ. The above problem is then relaxed to find LCP of two points sets by ϵ-congruences [6], [25], [26].

There have been extensive effort to develop computational tools to speed up the similar (sub)structure search. There is a large amount of literature on computer vision, computational chemistry and Patter recognition, which addresses (sub)structure identification problem. Most of them are based on graph theories or/and geometric hashing.

Graph theoretical approaches are widely used, such as in ASSAM [4] and VAST [7]. They model each structure as a graph and often solve the problem by applying the clique detection techniques. The basic idea goes as follows. Given two structures P_1 and P_2, all the internal distances among any two points are first calculated. Then, a product graph G can be generated whose nodes are pairs of points selected from P_1 and P_2 respectively. The edge between any two nodes exists if and only if two nodes are in the same relative position as their points. In other words, the distance between two nodes is equal to the distance between their points. Now, the problem of finding maximal common substructure is transformed to finding maximal completed subgraphs (cliques) in graph G. It is indeed the clique detection problem. For example, ASSAM detects cliques using a maximal common subgraph isomorphism algorithm from graph theory [27], [4]. Since the clique detection problem is NP-hard, many heuristic algorithms that are partially enumerative and branch-and-bound based have been proposed to speed up the process [8]. However, most of them have high exponential time complexity and are not scalable to large scale databases.

Geometric hashing is another popular technique [9]. It is initially developed in computer vision and now widely used in structure analysis to find the similar (sub)structures. Geometric features of protein structure are calculated and stored in a hash table. The basic idea is to define a coordinate system, called *reference frames*, for each structure. Consider 3D structures as an example. Any three non-collinear points belonging to the same structure can be used to define an $x-y$ plane and the z-axis is orthogonal to the plane. The coordinates of all other points of the structure are then mapped into the reference frames. The similarity between two 3D structure data is calculated by comparing two frame systems. Since any three non-collinear points could be used as the base for every structure and pairwise comparison is required to get an accurate solution, geometric hashing leads to the time complexity of $O(n^8)$, where n is the number of points in a structure. To improve the performance, some approximate techniques have been introduced [26]. However, there is no clear performance result reported.

Integrating the techniques of clique detection and geometric hashing, *k-clique hashing* has recently been proposed [10]. The key idea of *k-clique hashing* is to eliminate false-positive matches between two structures. It first follows the graph theoretical approach to make the product graph for two input structures and then decomposes the product graph into k-sized cliques which are considered as local matches of two structures. R*-tree is applied to index the k-cliques. Given a query structure, a similarity search is first performed on the R*-tree to find similar k-cliques (i.e.,local matches) and the local matches are then assembled into a global match. However, it is not applicable to very large sized structures.

There are also other techniques in protein bioinformatics for sequence similarity search [28]. However, their assumption is that the sequence order is pre-defined. The problem is different from ours to be studied. In this paper, we aim at developing an effective structure representation model which is invariant to rotation and translation and an highly efficient query processing technique to achieve real-time similarity search for very large structure databases.

III. Multi-resolution Localized Co-occurrence Model

In this section, we introduce the LCM model in a general setting, followed by its multi-resolution representation. Their properties are also explored.

A. Localized Co-occurrence Model (LCM)

We first define the neighborhood system and Local Joint Probability which form the base of LCM. For easy reference, Table 1 lists notations used in this paper.

Definition 1 (Neighborhood System): Let $S = \{s_i | i = 0, 1, 2, ..., n-1\}$ be a set of *n* sites. For any site s_i from S, it is associated with a state value $x_i \in \Lambda$, where $\Lambda = \{0, 1, ..., l-1\}$ is an application-specific *state space* containing complete state values which are discrete and finite. Each site s_i has a set

Notation	**Description**
P, p_i	a point set (i.e., a structure), a point
S, s_i	a site set, a site
N_i	neighborhood of site s_i
Λ	state space
l	number of state values in Λ
F	frequency of co-occurrence
Pr	local joint probability
L	resolution level
N	number of structures in database
R	highest resolution maintained
K	K most similar structures
U_{p_i}	unit structure for point p_i
$V(U_{p_i})$	volume of unit structure

TABLE I

LIST OF NOTATIONS

of neighboring sites, denoted as N_i ($\subset S$), according to an application-specific distance measure function. The *neighborhood system* is the set of neighborhoods of all sites $\mathbb{N} = \{\mathsf{N}_i | i = 0, 1, 2, ..., n-1\}$.

The above is a generic definition and can be applied to different applications. Generally speaking, the sites are used to refer to the objects of users' interests (i.e., descriptors of the objects) and state values are used to describe the objects (i.e., properties of the descriptors). Different applications may have different constraints in determining the neighborhood relation which is often asymmetric. Given a site s_i, those sites whose distances to s_i are within a predefined range or s_i's K nearest sites are often used as the neighboring sites of s_i. The modelling process is application-specific and can be better understood with real applications. For example, for image texture synthesis, we can model a digital image by considering each pixel in the image as a site and the grey value associated with the pixel as its state value which is contained in the state space $\Lambda = \{0, 1, ..., l-1\}$, where l is the number of grey levels in the image. However, in wireless networking, we may model a wireless network by considering an access point in the network as a site and the number of users covered by the access point as its state value which is contained in the state space $\Lambda = \{0, 1, ..., l-1\}$, where l is the maximal number of users simultaneously supported by a single access point. Since state values are discrete, for applications whose sites are associated with real values, it is necessary to transform each site's real value into a discrete state value. The transformation should be application-specific.

Next, based on the *Neighborhood System*, we define *Local Joint Probability*.

Definition 2 (Local Joint Probability (LJP)): Construct a 2D histogram H with domain $\Lambda \times \Lambda$. With respect to the neighborhood system $\mathbb{N}$, each cell value of H, $F(a, b)$ ($a, b \in \Lambda$), is the frequency of co-occurrence for state values a and b in the neighborhood system $\mathbb{N}$, which means a site with state value a is a neighbor of a site with state value b. The *Local Joint Probability (LJP)* of a and b is defined as:

$$Pr(a \cap b) = \frac{F(a,b)}{\sum F(i,j)}$$

Now we are ready to define the Localized Co-occurrence Model.

Definition 3 (Localized Co-occurrence Model (LCM)): The LCM is defined by the *Local Joint Probability Distribution (LJPD)* as:

$$LCM = \bigcup Pr(i \cap j)$$

Given a state space Λ, LCM defines the probability distribution of the neighbors of sites with state value i to be associated with state value j. LCM is based on the density estimation of local neighborhoods, which also captures coarse spatial relationship. Now we show an example to illustrate the above concept.

Example 1 (An LCM Example): In Figure 2(a), there are nine sites in the site set $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}$, each of them is associated with a real value 0.8, 0.6, 0.1, 0.3, 0.7, 0.4, 0.15, 0.20 and 0.90

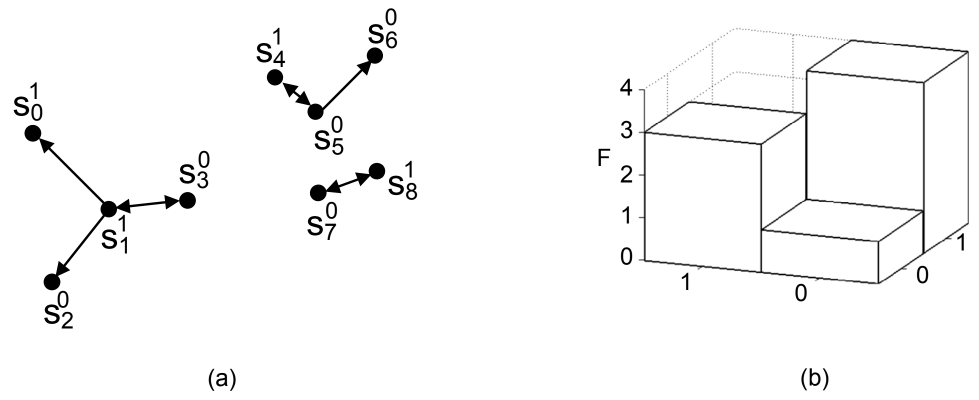


Fig. 2. $\Lambda = \{0, 1\}$ neighborhood system and its frequency histogram

respectively in the range of [0,1]. Assume the state space $\Lambda = \{0, 1\}$. Each site's real value need to be transformed into a state value. To do so, we can divide the range of [0,1] into two subintervals [0,0.5) and [0.5,1]. Sites whose values lie in the first subinterval are assigned 0 and sites whose values lie in the second subinterval are assigned 1. For example, s_0^1 represents site s_0 which is assigned 1 as its state value. Their neighboring relationships are represented by $\longrightarrow$, where $s_i \longrightarrow s_j$ indicates that s_i is a neighbor site of s_j. For example, s_1 is the neighboring site of s_0 and $\mathsf{N}_0 = \{s_1\}$. We call Figure 2(a) as the neighborhood system of S.

Based on the neighborhood system (in Figure 2(a)), a 2D histogram can be built by the frequency of co-occurrence of the state values (in Figure 2(b)). For example, the frequency for co-occurrence of $(0, 1)$ is 3 ($F(0, 1) = 3$) which is contributed by $s_3^0 \longrightarrow s_1^1, s_7^0 \longrightarrow s_8^1$ and $s_5^0 \longrightarrow s_4^1$. The local joint probability that state 0 is a neighbor of state 1 is $\frac{3}{9}$ ($Pr(0, 1) = \frac{3}{9}$), where 9 is the frequency summation of the histogram. Then the whole sites could be statistically represented by LCM as

$$\{Pr(0, 0), Pr(1, 0), Pr(0, 1), Pr(1, 1)\} = \{\frac{1}{9}, \frac{4}{9}, \frac{3}{9}, \frac{1}{9}\}.$$

□

As we can see, the *domain space* of LCM is $\Lambda \times \Lambda$. That is, LCM is described by l^2 local joint probability functions, where l is the scale of Λ. Hence LCM can be viewed as a l^2-dimensional vector. Such a statistical representation is invariant to scaling, rotation and translation.

From the 2D frequency histogram, we can get the following properties of LCM in a straightforward way.

Property 1: Total Probability[1]

$$\sum Pr(i \cap j) = 1$$

Property 2: Marginal Probability[2]

$$Pr(i) = \sum Pr(i \cap j), \quad i \in \Lambda$$

B. Multi-resolution LCM

Given a sites set and the predefined neighborhood system, the total number of co-occurrences between the state values assigned to the sites and their neighbors (i.e., the total number of neighbors for all sites) is fixed. In another word, $\sum\limits_{\forall i \in \Lambda, \forall j \in \Lambda} F(i,j)$ remains constant irrespective of the state space Λ, which is application-specific. By associating sites with multi-scale Λ, we can construct a multi-resolution representation of LCM. More specifically, LCM can be defined as a multi-resolution representation created from a *multinomial decomposition* on the 2D frequency histogram by changing the scale of Λ - *l*.

To illustrate the idea, let us start with a simple scenario by using the simplest binomial decomposition. Assume each site has a real value in [0,1]. At resolution $L = 1$, i.e., *l*=2, [0,1] is split into two subintervals [0,1/2) and [1/2,1]. The sites with real values in [0,1/2) and [1/2,1] are associates with state values 0 and 1 respectively. At resolution $L = 2$, i.e., $l = 4$, [0,1/2) and [1/2,1] are further split into [0,1/4), [1/4,1/2) and [1/2,3/4), [3/4,1] respectively. Sites with the real value in the above intervals are then associates with state values 0, 1, 2, and 3 correspondingly. At each resolution, a 2D frequency histogram is then constructed with respect to the new state space. Iterating this process L times generates a L-resolution LCM representation with $l = 2^L$. This defines the process of *binomial decomposition*. One important

[1] $\sum Pr(i \cap j) = \sum \frac{F(i,j)}{\sum\limits_{a,b \in \Lambda} F(a,b)} = \frac{\sum\limits_{i,j \in \Lambda} F(i,j)}{\sum\limits_{a,b \in \Lambda} F(a,b)} = 1$

[2] $Pr(i) = \frac{\sum\limits_{j \in \Lambda} F(i,j)}{\sum\limits_{a,b \in \Lambda} F(a,b)} = \sum\limits_{j \in \Lambda} \frac{F(i,j)}{\sum\limits_{a,b \in \Lambda} F(a,b)} = \sum\limits_{j \in \Lambda} Pr(i \cap j), \quad i \in \Lambda$

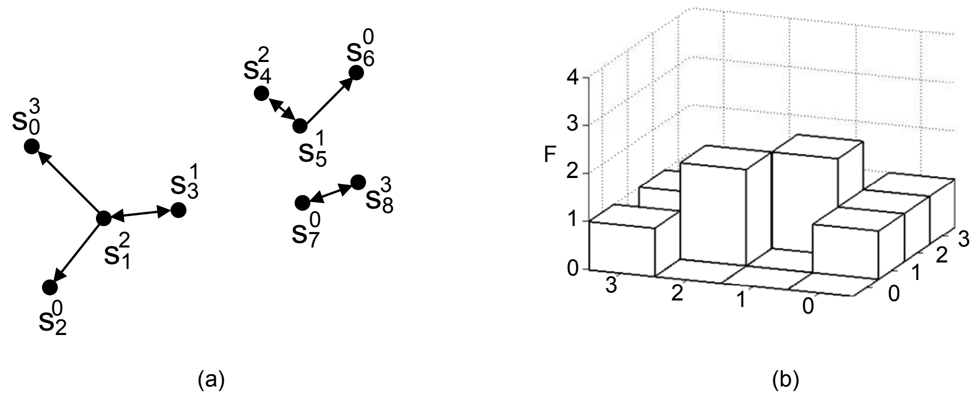


Fig. 3. $\Lambda = \{0, 1, 2, 3\}$ neighborhood system and its frequency histogram

property of this process is that *the sites associated with i^{th} state value at $(L-1)$-resolution will be diverted to be associated with either $(2i-1)^{th}$ or $(2i)^{th}$ state value at L-resolution* .

The construction of multi-resolution LCM by binomial decomposition can be further extended by splitting an interval into a larger number of subintervals. This method is referred as *multinomial decomposition*.

More generally, we have the following property for multi-resolution LCM representation. For easy illustration, in this paper, we use binomial decomposition.

Property 3 (Probability Decomposition): Denote i^{th} and j^{th} state values at $(L-1)$-resolution as l_i^{L-1} and l_j^{L-1}, and their corresponding decomposed ones at L-resolution as l_{2i-1}^L, l_{2i}^L and l_{2j-1}^L, l_{2j}^L respectively, we have

$$Pr(l_i^{L-1} \cap l_j^{L-1}) = Pr(l_{2i-1}^L \cap l_{2j-1}^L) + Pr(l_{2i}^L \cap l_{2j-1}^L) + Pr(l_{2i-1}^L \cap l_{2j}^L) + Pr(l_{2i}^L \cap l_{2j}^L)$$

where $Pr(l_i^{L-1} \cap l_j^{L-1})$ is LJP computed from the 2D frequency histogram.

Proof: In binomial decomposition, the sites associated with i^{th} state value at $(L-1)$-resolution can only be associated with either $(2i-1)^{th}$ or $(2i)^{th}$ state value at L-resolution, i.e., $Pr(l_{2i-1}^L \cap l_{2j-1}^L) + Pr(l_{2i}^L \cap l_{2j-1}^L) = Pr(l_i^{L-1} \cap l_{2j-1}^L)$ and $Pr(l_{2i-1}^L \cap l_{2j}^L) + Pr(l_{2i}^L \cap l_{2j}^L) = Pr(l_i^{L-1} \cap l_{2j}^L)$, and similarly for j. Thus, exhausting these possibilities, the right hand side of the above is immediate. $\square$

As decomposition process goes on from resolution $(L-1)$ to L, the frequency indicated by each grid coordinated by (l_i^{L-1}, l_j^{L-1}) in the 2D frequency histogram is further decomposed into four grids coordinated by $(l^L_{2i-1}, l^L_{2j-1}), (l^L_{2i}, l^L_{2j-1}), (l^L_{2i-1}, l^L_{2j}), (l^L_{2i}, l^L_{2j})$ respectively.

Recall Figure 2 that shows a 1-resolution LCM representation for Example 1. In Figure 3, its 2-resolution LCM representation is depicted based on binomial decomposition, where sites are associated with new state values. There is no change on the neighborhood. However, we have to refine the state space as $\Lambda = \{0, 1, 2, \cdots, 2^L - 1\}$ for L-resolution. More information is infused into next higher resolution since the frequency indicated by each grid is further disaggregated. As we can see, low resolution LCM can be used to constrain its next higher resolution representation.

Given that the total number of co-occurrences is constant when the neighborhood system remains unchanged during the decomposition process, LJPD becomes more sparsely dispersed over its domain space at higher resolution. With a larger L, the number of sites associated with the same state value is expected to be less. When L=0, all sites have same value 0. As L increases, sites have less probability to have same state values. Clearly, the probability converges to 0 for sure as L increases to infinity, so as to the probability of localized co-occurrence of any two state values. The change from Figure 2b to Figure 3b depicts this phenomenon. Clearly, *higher resolution converges LCM to get closer to the structure/distrubition of original sites*. However, the dimensionality of LCM representation, i.e., $l^2 = (2^L)^2 = 4^L$, increases too. With a larger l, computing the statistical similarity between two LJPDs turns to be more expensive.

In short, LCM captures the statistical characteristics of a set of sites. The spatial structure can be captured by localizing the co-occurrence relationship. Multi-resolution LCM representation provides the flexibility to examine different levels of distribution/structure detail. More interestingly, by the property of Probability Decomposition (Property 3), low resolution LCM is an aggregation of higher resolution one which in turn can be constrained by low resolution one. Next, we look at how LCM can be used to model a 3D structure.

IV. Structure Modelling

In this section, we look at how multi-resolution LCM can be applied to model a 3D structure. Obviously, a single point does not carry any structural information. Inspired by that *a line can be determined by 2 points in 1D space, and a triangle can be determined by 3 points in 2D space,* we can make use of *a triangle pyramid determined by 4 points in 3D space to represent a simple 3D structure*. Meanwhile, structures having local interactions are also more meaningful, particularly in structural biology. Motivated by this, given a 3D structure $P = \{p_i | i = 1..m\}$, we transform P from a set of points into a set of *unit structures* to take the surrounding structural information into consideration.

Definition 4 (Unit Structure): Given a 3D structure $P = \{p_i | i = 1..m\}$, for each point p_i, it is associated with a unit structure U_{p_i} which is a triangle pyramid determined by p_i and its 3 nearest points in P.

Clearly, the unit structure for each point p_i is constructed by using p_i's nearest neighbors which reflects the local spatial information surrounding p_i. The Euclidean distance is used to measure the closeness of two points. Since a unit structure is determined by 4 points, the same unit structure can be shared by at most 4 points, when every point is among another's 3 nearest neighbors. Figure 4 shows an example of a structure of twelve points forming four unit structures. As we can see, p_0 and p_1 share the same unit structure formed by $\{p_0, p_1, p_2, p_3\}$, i.e., $U_{p_0} = U_{p_1}$=$\{p_0, p_1, p_2, p_3\}$. Similarly, $U_{p_2} = U_{p_3} = U_{p_4}$=$\{p_1, p_2, p_3, p_4\}$, $U_{p_5} = U_{p_6} = U_{p_7}$=$\{p_5, p_6, p_7, p_8\}$, and $U_{p_8} = U_{p_9} = U_{p_{10}} = U_{p_{11}}$=$\{p_8, p_9, p_{10}, p_{11}\}$.

Intuitively, a unit structure shared by more points indicates higher local density, i.e., points are closer to each other. Each point corresponds to a unit structure. After a unit structure for every point is constructed, the structure P is then represented by m unit structures.

Next, we use the $volume$ of a unit structure $V(U_{p_i})$ as its descriptor. In the special case when a point has multiple choices to construct its unit structures, the unit structure with the minimal volume is selected. $\forall V(U_{p_i}),\ 0 \leq i < m$, we normalize them into the range of [0,1]. Given the resolution level L (i.e., the state values in Λ are discrete from 0 to $2^L - 1$), the range of [0,1] is then evenly divided into 2^L subintervals with IDs from 0 to $2^L - 1$.

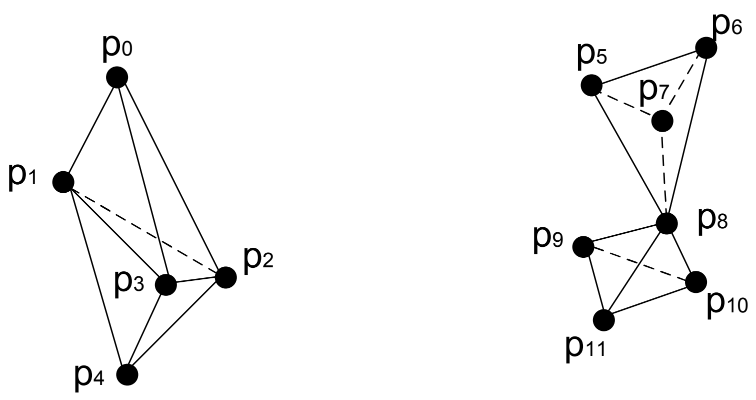


Fig. 4. Construction of unit structures.

Now we can apply LCM to model a 3D structure by considering each unit structure as a site and associating each site a subinterval ID where the unit structure's volume lies. That is, if a normalized value $V(U_{p_i})$ falls in the subinterval [0,1/2^L), we assign site p_i a state value 0. If a normalized value $V(U_{p_i})$ falls in the interval [1/2^L,2/2^L), we assign site p_i a state value 1, and so on. Binomial decomposition is applied to generate a multi-resolution LCM representation. As for the neighborhood system, we define the neighborhood of a site (i.e., a unit structure U_{p_i}) as its K most closest unit structures or close unit structures whose distances to U_{p_i} is not greater than a predefined distance threshold, where *the distance of two unit structures is measured by inter-distance of their gravity centers*.

Finally, by applying LCM, a 3D structure is modelled as a multi-resolution LCM described by LJPD which is invariant to rotation and translation. Formally, given a 3D structure P, its L-resolution LCM is denoted as:

$$LCM_P^L = \{Pr_P(l_i^L \cap l_j^L) : 0 \leq i, j < 2^L\}$$

where $Pr(l_i^L \cap l_j^L)$ is LJP at L-resolution.

The similarity between two 3D structures P and Q is then approximated by the statistical similarity of their LCMs at L-resolution as defined below:

Definition 5 (LCM Similarity Measure):

$$d(P,Q) \approx d(LCM_P^L, LCM_Q^L) = \sum_{0 \le i,j < 2^L} ||Pr_P(l_i^L \cap l_j^L) - Pr_Q(l_i^L \cap l_j^L)||$$

The distance between two LCMs is measured by city-block distance whose complexity is linear to the dimensionality of LCM, i.e., $(2^L)^2 = 4^L$

During the construction of multi-resolution LCM, the following factors may affect the quality of the model:

- *Neighborhood System:* The predefined neighborhood system will affect the construction of 2D frequency histogram determined by localized co-occurrences. Obviously, different applications may define different neighborhood systems which may affect the accuracy of LCM. In this paper, we will examine the neighborhood system defined by U_{p_i}'s top K nearest neighbors and its close neighbors whose distances are less than or equal to a predefined distance range.
- *Resolution L:* Resolution level L determines the scale of state space $\Lambda = \{0, 1, 2, ..., 2^L - 1\}$. Since a higher resolution LCM captures more details and is a more accurate approximation, a larger L is preferred for accuracy reason. However, a larger L causes more expensive similarity computation. More importantly, a too large L leads to sparse distribution of LJPD in domain space. Given that the number of unit structures for a 3D structure is fixed, when L becomes larger, there is an increasing danger to associate similar unit structures with different state values which are discrete. When L reaches large enough, every unit structure having different volume will have a different state value. This is obviously against the perception of similarity search. Hence a reasonably large L value is needed to avoid positive drops and more expensive similarity computation. Our experimental results (in Section VI) suggest that *the dimensionality of LCM (i.e., 4^L) should be about the average number of points per structure in the database*.

The above factors will be further analyzed together with experiment results later.

As we can see, by modelling a 3D structure using LCM, structural similarity can be approximated by statistical similarity in linear time complexity. Note that LCM is a generic model to capture the local distribution by using localized data co-occurrence information. Different applications may have different data types. To apply LCM, it's necessary to transform other data types of values into numerical values so that the discrete state values can be assigned in LCM. For example, to model a 3D structure using LCM in this paper, there is a normalization step to divide real values (i.e., volume values) into intervals so that numerical values can be assigned. Low resolution representations will be used to further speed up query processing as we shall see next.

V. Structure Query Processing

In this section, we first define the problems of structure similarity search and largest common substructure identification, followed by our efficient solution iBound which explores the properties of multi-resolution LCM.

A. Query Types

The general problems of finding similar structures and the largest common substructures arise in many areas from computer vision, pattern recognition, to structural biology.

Definition 6 (Most Similar Structure): Given a 3D structure database and a query structure Q, the most similar structure of Q in the database is the structure which minimizes the distance to Q, measured by a predefined distance function.

The above definition can be easily relaxed for top-K most similar structures, or similar structures whose distances to the query are within a distance range. Commonly, *root mean square deviation* (RMSD) is used [5]. Smaller RMSD is better, and zero indicates exact match of two structures. Given two structures $P = \{p_i | i = 1..m\}$ and $Q = \{q_i | i = 1..m\}$, their RMSD is computed as

$$RMSD(P, Q) = \min_T \sqrt{\sum_{i=1}^{m} (||T(q_i) - p_i||^2)}$$

where T is a transformation of P. A transformation can be performed by a translation (three distances) and a rotation (three angles). All transformations have to be computed to find the best match. More seriously, when two structures are not in the same size, all the same sized substructures as the small structure from the large structure have to be generated and compared with the small structure. Obviously, this distance measure is extremely expensive. In this paper, we refer to the process of finding similar structures as *structure similarity search*.

Definition 7 (Largest Common Substructure): Given two 3D structures $P = \{p_i | i = 0, 1, 2, ..., m_1\}$ and $Q = \{q_i | i = 0, 1, 2, ..., m_2\}$, and a tolerance threshold $\epsilon \geq 0$, we want to find a transformation T that maximizes the cardinality of a subset $I_Q \subseteq Q$, such that $||T(I_Q) - I_P|| \leq \epsilon$ where $I_P \subseteq P$ with the same size of I_Q. I_Q (or I_P) is the largest common substructure shared by P and Q.

Several distance measures can be used, such as Hausdorff distance [25] and bottleneck distance [29]. We refer the process of identifying the largest common substructure as *substructure identification*.

In this paper, we consider the problem of finding the top K most similar structures from a large structure database and identifying the largest common substructure between the query and its similar structure.

B. iBound

Given a 3D structure database represented by multi-resolution LCM as $\Phi = \{LCM_i^L | 0 \leq i < N, 1 \leq L \leq R\}$, where N is the number of structures and R is the highest resolution maintained in the database, we are interested to find the top K most similar structures based on their R-resolution LCMs, since R-resolution is more sophisticated in representing structures. LCM can be regarded as a high-dimensional point and can be indexed by existing high-dimensional data structures. However, the dimensionality of R-resolution LCM, i.e., $(2^R)^2 = 4^R$, reaches up to hundreds when $R = 4$, or much more when R becomes larger[3]. Due to known "dimensionality curse", existing methods are not able to effectively index very high dimensional space [30], [31]. Approximate search becomes a more practical solution [32], [33]. From previous study on hyper-dimensional indexing, distance computational cost dominates the overall

[3]As we will see later in experiments, a practical R value is around 4

cost as dimensionality reaches hundreds or more [32]. In this paper, we focus on computational reduction on the query processing.

The complexity of distance computation in R-resolution is 4^R. Based on Property 3, a lower resolution LCM is an aggregation from its higher resolution representation. It is worth to note that *the total computational cost of calculating the similarities between two LCMs from 1 to $(R-1)$-resolution is less than the computational cost at R-resolution*. By maintaining multi-resolution LCM in the database, low resolution LCM can be used to prune dissimilar structures quickly to avoid more expensive computation at next higher resolution. Before we move to the detailed algorithm, we first present a lemma on multi-resolution LCM that paves the way to our algorithm.

Theorem 1: Lower Bound

$$d(LCM_P^{L-1}, LCM_Q^{L-1}) \leq d(LCM_P^L, LCM_Q^L)$$

Proof: Denote i^{th} and j^{th} state values at resolution $(L-1)$ as l_i^{L-1} and l_j^{L-1}, and their corresponding decomposed ones at resolution L as l_{2i-1}^L, l_{2i}^L and l_{2j-1}^L, l_{2j}^L respectively. For LJPD, from Property 3, we have:

$$\begin{aligned}
&||Pr_P(l_i^{L-1} \cap l_j^{L-1}) \quad - Pr_Q(l_i^{L-1} \cap l_j^{L-1})|| \\
&= ||(Pr_P(l_{2i-1}^L \cap l_{2j-1}^L) - Pr_Q(l_{2i-1}^L \cap l_{2j-1}^L)) + (Pr_P(l_{2i}^L \cap l_{2j-1}^L) - Pr_Q(l_{2i}^L \cap l_{2j-1}^L)) + \\
&\quad (Pr_P(l_{2i-1}^L \cap l_{2j}^L) - Pr_Q(l_{2i-1}^L \cap l_{2j}^L)) + (Pr_P(l_{2i}^L \cap l_{2j}^L) - Pr_Q(l_{2i}^L \cap l_{2j}^L))|| \\
&\leq ||(Pr_P(l_{2i-1}^L \cap l_{2j-1}^L) - Pr_Q(l_{2i-1}^L \cap l_{2j-1}^L)|| + ||Pr_P(l_{2i}^L \cap l_{2j-1}^L) - Pr_Q(l_{2i}^L \cap l_{2j-1}^L)|| + \\
&\quad ||Pr_P(l_{2i-1}^L \cap l_{2j}^L) - Pr_Q(l_{2i-1}^L \cap l_{2j}^L)|| + ||Pr_P(l_{2i}^L \cap l_{2j}^L) - Pr_Q(l_{2i}^L \cap l_{2j}^L)||, \quad 0 \leq i, j < 2^{L-1}
\end{aligned}$$

Based on the LCM similarity measure, the total distance at $(L-1)$-resolution is less than or equal to that at L-resolution. $\square$

Given two LCMs and their distance at $(L-1)$-resolution, Theorem 1 provides a lower bound on their distance at L-resolution. Based on Theorem 1, we propose a novel similarity query processing method for multi-resolution LCM called iBound (incremental and Bounded search). The key idea of iBound is to compute distances at low resolutions which lower bounds the distances at higher resolutions to avoid more expensive computation. Figure 5 outlines the algorithm.

```
Algorithm: iBound
Input: $\Phi$, $LCM_Q^L : 1 \leq L \leq R$
Output: topK most similar $LCM^R$
1. for i = 0 to N-1
2.     buffer[i].id = i;
3.     buffer[i].resolution = 1;
4.     buffer[i].distance = $d(LCM_i^1, LCM_Q^1)$;
5. quick_sort(buff);
6. while !all_highest_resolution(buffer,R,K)
7.     for i = 0 to K
8.         if buffer[i].resolution $<$ R
9.             j=buffer[i].id;
10.            L=++buffer[i].resolution;
11.            buffer[i].distance=$d(LCM_j^L, LCM_Q^L)$;
12.    topK_sort(buffer);
13. return buffer[i]: $0 \leq i < K$;
```

Fig. 5. Incremental and bounded search.

Given a query's multi-resolution LCM, iBound starts with an initial loop to compute all LCM^1s' distances to LCM_Q^1 at 1-resolution[4] (lines 1-4). The structures are then sorted based on their distances to the query and maintained in a buffer (line 5). For the top K results in the buffer, iBound iteratively checks their resolution levels, until all top K structures reach highest resolution R (line 6-12). During each iteration, for each current top K result, if its resolution is less than R, its distance to the query is

[4]Note that LCM is the same for all structures when L=0

id	resolution	distance
2	1	0.10
3	1	0.15
5	1	0.25
1	1	0.40
4	1	0.50
7	1	0.56
6	1	0.60

Fig. 6. After initial iteration.

id	resolution	distance
3	**2**	**0.20**
5	1	0.25
2	**2**	**0.30**
1	1	0.40
4	1	0.50
7	1	0.56
6	1	0.60

Fig. 7. After 1st **while** iteration.

id	resolution	distance
3	2	0.20
2	2	0.30
5	**2**	**0.35**
1	*1*	*0.40*
4	*1*	*0.50*
7	*1*	*0.56*
6	*1*	*0.60*

Fig. 8. After 2nd **while** iteration.

then computed at next higher resolution (lines 7-11). The buffer is then re-sorted by calling $topK\_sort$ which inserts the updated at most K distances into the right position in the buffer. The iteration continues until all top K results in the buffer are at R-resolution. Theorem 1 ensures the correctness of iBounds to find top K most similar structures. The advantage of iBound is to save expensive distance computations at high resolutions. To have a clear run on iBound, an example is shown in Example 2.

Example 2 (An iBound Example): Given a set of multi-resolution LCMs with N=7, R=2 and K=2, Figure 6 shows the sorted buffer after the initial loop when $L = 1$. In the first iteration of while loop, iBound checks the top 2 results' resolutions and compute their distances at 2-resolution. Meanwhile, the buffer is re-sorted by inserting the newly computed distances to the right position. The updated buffer is shown in Figure 7. In the next iteration, iBound computes the second most similar structure's distance at 2-resolution and updates the buffer again. The updated buffer is shown in Figure 8, whose top K results all are at 2-resolution. iBound stops and returns the results. As we can see from above tables, distance computation at 2-resolution does not occur for structures with ID 1, 4, 7 and 6.

□

The major cost of iBound comes from the while loop. For each iteration, there are at most K distance computations and insertions. However, in the same iteration, different structures may be at different resolution, i.e., different complexity. In the worst case, all structures reach R-resolution. While in the best case, only K structures reach R-resolution. For a structure which is at L-resolution after iBound stops, the following percentage of computational cost can be saved:

$$\frac{4^R - \frac{4(4^L-1)}{3}}{4^R}$$

For instance, given R=4 and L=2, about $\frac{256-20}{256} \approx 92.2\%$ computational cost can be saved. The efficiency of iBound will be further studied in the experiments.

C. Substructure Identification

It is interesting to further identify the largest common substructure between two structures which usually has important implication. Next, we look at how the largest common substructure between a query and its similar structure can be identified. Figure 9 outlines the algorithm called *ϵ-congruence Superposition*. Note that the identification happens at R-resolution.

The basic idea of ϵ-congruence Superposition is to superpose two LCMs generated from their 2D frequency histograms. Local joint probability $Pr(l_i^R \cap l_j^R)$ indicates the probability of the localized co-occurrence in a neighborhood system. We here borrow the ϵ-congruence concept. $||Pr_P(l_i^R \cap l_j^R) - Pr_Q(l_i^R \cap l_j^R)|| \leq \epsilon$ suggests that the local spatial relationship between points associated with l_j and their neighbors (i.e., points associated with l_i) are superposed between structure P and query Q, where ϵ is a user-defined distance tolerance threshold. Since LCM does not carry exact location information, ϵ-congruence Superposition is also approximate. Our experiment shows the effectiveness of our approach.

VI. PERFORMANCE STUDY

In this section, we conduct an extensive performance study to test the effectiveness of LCM and efficiency of iBound.

```
Algorithm: $\epsilon$-congruence Superposition
Input: $LCM_S^R, LCM_Q^R$
Output: $I_Q$
1. for i = 0 to $2^R$-1
2.     for j = 0 to $2^R$-1
3.         if $||P_S(l_i^R \cap l_j^R) - P_Q(l_i^R \cap l_j^R)|| \leq \epsilon$
4.             add points mapped into unit structures
               associated with $l_i^R$ or $l_j^R$ into $I_Q$;
5. return $I_Q$;
```

Fig. 9. Largest common substructure identification.

A. Set Up

LCM approximates the structure similarity whose computation is NP-hard by the localized statistical similarity whose time complexity is linear to the dimensionality of LCM. Its effectiveness has to be tested. To have a sound and valid ground-truth for proper evaluation, we use the Structural Classification of Proteins (SCOP) database [34] which is a comprehensive classification of all proteins of known structure, according to their evolutionary and structural relationships. The SCOP database, created by visual inspection and comparison of structures, but with the assistance of tools to make the task manageable, has many levels in the classification tree. But the principal levels are family, superfamily and fold. At higher levels of the tree, proteins are clustered with structural similarity. In this paper, we use its highest level, where structurally similar proteins are clustered into a single $family$. SCOP has been updated to include all PDB 25,973 entries released up to 1 October 2004 with total number of 2,845 families [35]. 8300 protein structures are used in our experiment to assess the performance of our method. The average size of proteins are 300, which implies that each structure consists 300 3D points averagely.

In this paper, the following measures are used to indicate the effectiveness of our LCM and efficiency of iBound.

- **Precision-Recall**: Given a query from a family, its ground-truth are those structures in the same family. Denote the returned results as $return$, and the ground-truth as $truth$, the precision and recall are then defined as:

$$Precision = \frac{|return \cap truth|}{|return|}$$

and

$$Recall = \frac{|return \cap truth|}{|truth|}$$

where $| \cdot |$ is the set size. By default, binomial decomposition uses equal-space interval division.

- **Computational Improvement (CI)**: To test the efficiency of iBound, we use the Computational Improvement as defined below:

$$\frac{N * 4^R}{\sum_{j=1}^{N} \sum_{i=1}^{L_j} 4^i}$$

where N is the number of structures, L_j is the resolution that the j^{th} structure reaches after iBound terminates. CI indicates how many times iBound can improve the method of computing all distances at R-resolution.

Our methods are implemented in C++ and run on Windows XP operating system with Pentium 4 CPU (2.8GHZ with 1GB RAM). 100 structures are randomly selected from the database as queries. All the experimental results reported will be averaged over those queries. We also tune the parameters used in our methods to see how they affect the performance and how their good values can be determined.

B. *Effect of Neighborhood System*

The neighborhood system in LCM determines the degree of locality. We first look at how the neighborhood system affects the effectiveness of LCM. In this experiment, we set resolution $L = 4$. Two types of neighborhood are defined: *range neighborhood* and *knn neighborhood*. Given a unit structure U_{p_i}, range neighborhood contains those unit structures whose distances to U_{p_i} is less than or equal to a predefined distance range - γ, and knn neighborhood contains the top K most nearest neighbors of U_{p_i}, where the distance between two unit structures is measured by the Euclidean distance of their gravity centers.

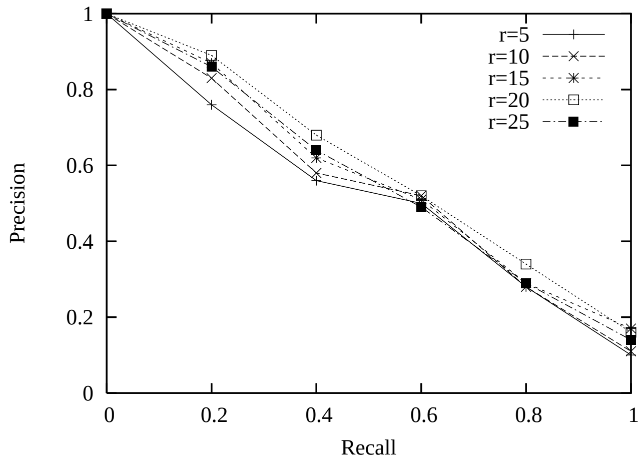


Fig. 10. Effect of $range$ neighborhood.

Figure 10 shows the Precision-Recall for LCM when range neighborhood is defined, where the range threshold γ varies from 5Å to 25Å. We have the following observations. First, for all γ values, precision descends as recall increases, and LCM achieves satisfactory performance. Especially when γ=20Å, LCM achieves precision of about 90%, 70% and 60% for recall of 20%, 40%, and 60% respectively. This is acceptable from information retrieval point of view. It is very important to note that LCM tackles the NP-hard problem of 3D structure comparison by linear statistical similarity measure. Second, as γ increases, the quality of LCM becomes better. However, when γ reaches 25Å, the effectiveness of LCM drops and becomes slightly worse than that of γ=20Å. Since γ determines the range of neighborhoods, a larger γ corresponds to larger neighborhoods which are expected to contain more points. The frequency of co-occurrence in LCM is computed based on the locality defined by neighborhoods. A too small neighborhood does not carry enough local information (consider the extreme case when a point has no neighbor), while a too large neighborhood ignores the detailed local information (consider the extreme case when the neighborhood contains all points). Hence a neighborhood system should be properly defined in order to be have a balanced degree of locality for LCM. From Figure 10, we set γ=20Å for the rest experiments.

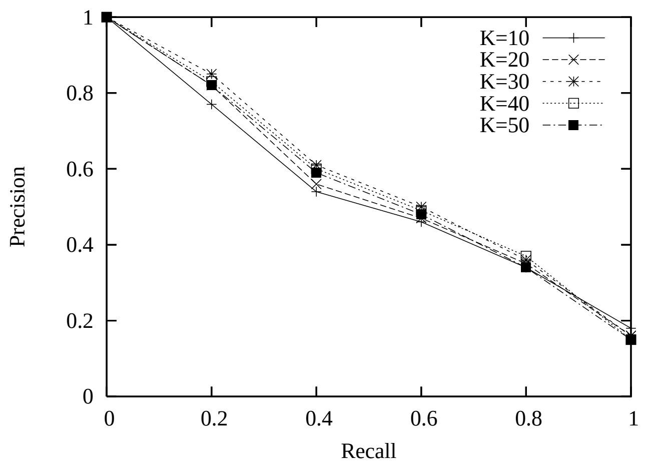


Fig. 11. Effect of knn neighborhood.

Figure 11 depicts the Precision-Recall for LCM when knn neighborhood is defined, by varying K from 10 to 50. It shows similar trend as in Figure 10. However, comparing with the effect of range neighborhood in Figure 11, LCM with knn neighborhood is less sensitive to K and it performs marginally worse. One possible reason is that top K nearest neighbors do not explicitly carry distance information. For different points, the distances to their respective nearest neighbors may vary greatly. However, range neighborhood explicitly specifies a distance threshold which bounds the neighborhoods. That is, range neighborhood may better defines the degree of locality in LCM.

C. *Effect of Resolution* L

Resolution level L is another important factor which affects the quality of LCM in representing 3D structures. Higher resolution LCM corresponds to more detailed representation, which also causes more expensive distance computation.

Figure 12 shows the Precision-Recall for LCM in different resolutions from L=1 to L=5. As expected, LCM performs better when LCM resolution gets higher. However, when $L = 5$, LCM turns out to get worse. This is interesting. As we mentioned, although higher resolution converges LCM to get closer to intrinsic data distribution, a too large L leads to sparse distribution of LJPD in domain space. As

L becomes larger, there is an increasing danger to associate similar unit structures with different state values. Based on binomial decomposition, the volume intervals in very high resolution are divided into very small range. As a result, many similar unit structures will be assigned to different state values. Hence LCM resolution should be set at reasonable levels. Given that the number of points in a protein structure is typically hundreds and LCM performs best when its dimensionality is $4^4 = 256$, a reasonable way to determine the highest LCM resolution R is to set 4^R be approximately equal to the average number of points per structure in the database.

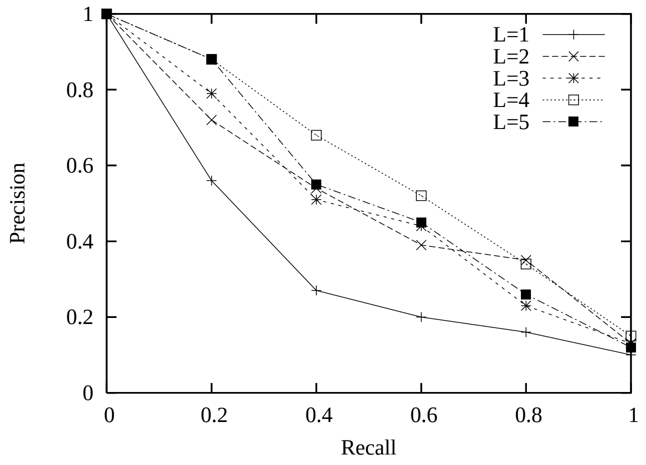
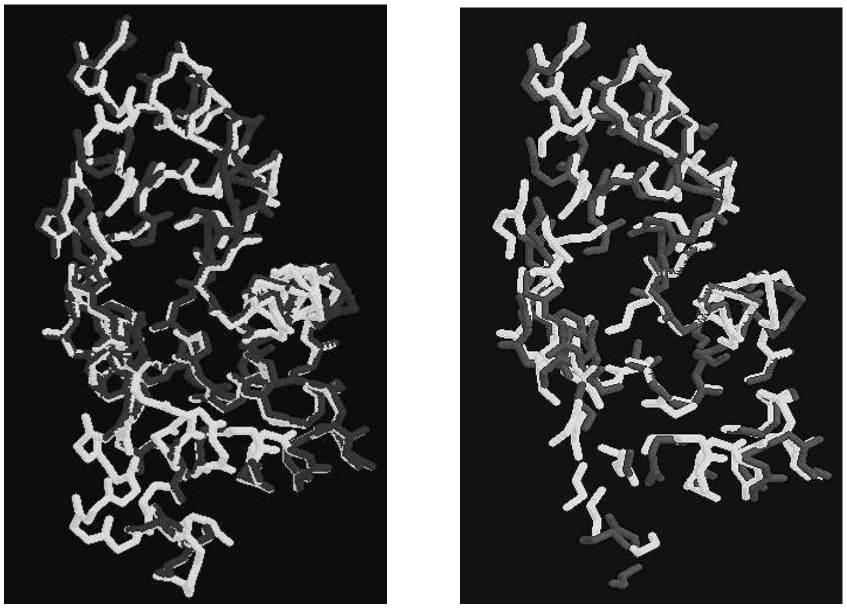


Fig. 12. Effect of resolution L.

D. *Effectiveness of Substructure Identification*

In this experiment, we look at the largest common substructure identified by ϵ-congruence Superposition method on LCM. In this experiment we set $R = 5$ for a higher resolution identification. Obviously, ϵ affects the number of points being superposed between two structures. The smaller the ϵ is, the less points are superposed, and the more rigid the superposition is. When ϵ=0, an exact superposition is required. Users can adjust the value of ϵ to have different levels of superpositions. To have a clearer picture, Figure 13(a) and Figure 13(b) show two sample largest common substructures for a query identified by ϵ-congruence Superposition for ϵ=0.0005 and 0.0001 respectively. As we can see, our method can identify the largest common substructure with very few mismatching points, and a smaller ϵ results in a more rigid superposition. After all, LCM is an effective approximation of the intrinsic structure.



(a) ϵ=0.0005 (b) ϵ=0.0001

Fig. 13. Visualization of common substructure.

E. Efficiency

In this section, we test the performance of our methods. The recent proposal *k-clique hashing* [10] is selected for comparison. Traditional graph theoretical approaches and geometric hashing approaches cannot be compared since they do not have practical running time complexities and results are typically reported from very small numbers of structures with small sizes [4], [36]. Note that our database is very large, containing 8300 structures with average size of about 300 points.

From our experiments, the total runtime for *k-clique hashing* is impractically high. Each query takes thousands of seconds or more to complete. This is caused by two main factors. Firstly, the average number of points in our protein structures is very large (about 300). Therefore, huge number of small sized cliques are generated and the clique database size becomes massive. Secondly, given a query structure, a large number of small sized query cliques are also generated and each of them has to be searched in the clique database before the global match can be finalized. It is expected that this method cannot be applied on very large structure databases for practical usages. However, our method is able to complete the search within one second, which improves the performance of *k-clique hashing* by several orders of magnitude (the comparison cannot be well visualized in a figure). This is reasonable since we model a structure as a single LCM, which is invariant to rotation and translation. Meanwhile, its time complexity for similarity measure is linear.

Existing approaches are not comparable to our method that achieves real-time response. Therefore, we focus on the self comparison of iBound.

iBound is proposed to speed up the similarity search by using multi-resolution LCM. We test the performance of iBound by searching top K most similar structures of a query. Figure 14 depicts the computational improvement of iBound when the highest resolution R varies from 1 to 5, comparing with directly computing the distances at the highest resolution. There are two obvious observations. First, the improvement achieved by iBound goes upward rapidly as R increases. This proves that there are only a small percentage of LCMs necessary for distance computation at the highest resolution. iBound is able to speed up the similarity search by tens of times since it prevents a large portion of LCMs to be computed at high resolutions. Second, iBound performs better for smaller K values. Especially, the improvement increases exponentially for K=10. This is expected. For smaller K values, iBound computes less distances at higher resolution in each iteration. This experiment confirms the novelty of iBound.

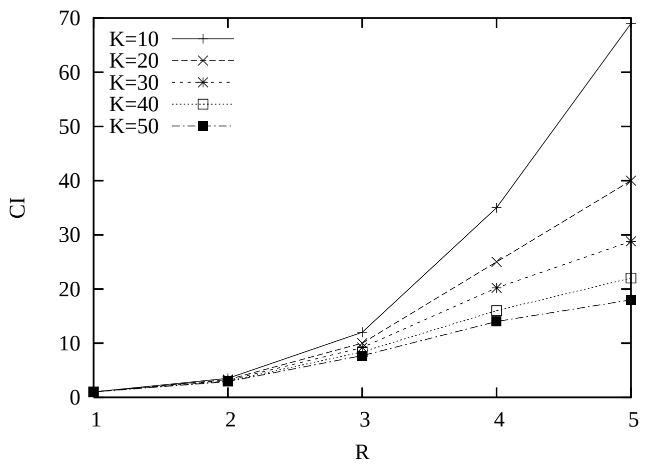


Fig. 14. Computational improvement by iBound.

VII. Conclusion and Future Work

In this paper, we present a novel statistical model called Localized Co-occurrence Model (LCM) to model 3D structures. LCM captures distribution characteristics and spatial structure of data by localizing the point co-occurrence relationship within a predefined neighborhood system. The statistical characteristics of a 3D structure are defined by Local Joint Probability Distribution (LJPD). The statistical similarity of two LCMs is then computed based on their LJPDs efficiently. We further develop Multi-resolution LCM for fast pruning. Higher resolution converges LCM to get closer to the intrinsic characteristics of data distribution/structure and can be constrained by low resolution. To further improve the performance, a novel structure query processing method called iBound is proposed to reduce the computational cost. By maintaining low resolution LCM whose distance lower bounds the distance in higher resolution, iBound avoids a large amount of more expensive computation at higher resolution. By enlarging the local interactive neighborhood, the largest common substructure can also be found. Finally, our experiment results prove the effectiveness and efficiency of our methods.

There are several directions that can be further explored in future. First, different definition on the neighborhood system can be further studied. Second, a weighted LJPD considering the neighbors' distances will be further investigated. Third, effective indexing structures are also useful to organize multi-resolution LCMs such that query response can be further reduced. Forth, internal structure of LJPD that could facilitate search will be also studied.

References

[1] P. Artymiuk, R. Spriggs, and P. Willett, "Graph theoretic methods for the analysis of structural relationships in biological macromolecules," *Journal of the American Society for Information Science and Technology*, vol. 56, no. 5, pp. 518–528, 2005.

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

[3] L. Holm and C. Sander, "Searching protein structure database has come of age," *Proteins*, vol. 19, pp. 165–173, 1994.

[4] R. Spriggs, P. Artymiuk, and P. Willett, "Searching for patterns of amino acids in 3D protein structures," *J. Chem. Inf. Comput. Sci.*, vol. 43, no. 2, pp. 412–21, 2003.

[5] E. Ingvar, J. Inge, and W. R. Taylor, *Protein Bioinformatics: An Algorithmic Approach to Sequence and Structure Analysis*. WILEY, 2004.

[6] T. Akutsu, H. Tamaki, and T. Tokuyama, “Distribution of distances and triangles in a point set and algorithms for computing the largest common point sets,” in *Discrete and Computational Geometry*, 1998, pp. 20:307–331.

[7] J.-F. Gibrat, T. Madej, and S. Bryant, “Surprising similarities in structure comparison,” *Curr. Opin. Struct. Biol.*, vol. 6, pp. 377–385, 1996.

[8] E. Gardiner, P. Artymiuk, and P. Willett, “Clique-dection algorithms for matching three-dimensional molecular structures.” *Journal of Molecular Graphics and Modeling*, vol. 15, pp. 245–253, 1997.

[9] H. Wolfson, “Geometric hashing: an overview.” *IEEE Comp. Science and Eng.*, vol. 4, no. 4, pp. 10–21, 1997.

[10] N. Weskamp, D. Kuhn, E. H$\ddot{u}$llermeier, and G. Klebe, “Efficient similarity search in protein structure databases by k-clique hashing,” *Bioinformatics*, vol. 20, no. 10, pp. 1522–1526, 2004.

[11] “Protein data bank,” http://www.rcsb.org/pdb/, May, 2007.

[12] Z. Xie and G. E. Farin, “Image registration using hierarchical b-splines,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 1, pp. 85–94, 2004.

[13] H.-P. Kriegel, S. Brecheisen, P. Krger, M. Pfeifle, and M. Schubert, “Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects,” in *SIGMOD*, 2003, pp. 587–598.

[14] B. Bustos, D. A. Keim, D. Saupe, T. Schreck, and D. V. Vrani$\acute{c}$, “Feature-based similarity search in 3D object databases,” *ACM Comput. Surv.*, vol. 37, no. 4, pp. 345–387, 2005.

[15] H. S. Ip and Y. F. Wong, “3D head models retrieval based on hierarchical facial region similarity.” in *International Conference on Vision Interface*, 2002, pp. 314–319.

[16] R. Ohbuchi, T. Otagiri, M. Ibato, and T. Takei, “Shape-similarity search of three-dimensional models using parameterized statistics,” in *Pacific Conference on Computer Graphics and Applications*, 2002, pp. 265–273.

[17] R. Ohbuchi, T. Minamitani, and T. Takei, “Shape-similarity search of 3D models by using enhanced shape functions,” in *Theory and Practice of Computer Graphics*, 2003, p. 97.

[18] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, “Shape distributions,” *ACM Trans. Graph.*, vol. 21, no. 4, pp. 807–832, 2002.

[19] E. Paquet, M. Rioux, A. Murching, T. Naveen, and A. Batabai, “Description of shape information for 2-D and 3-D objects,” *Signal processing: Image communication*, vol. 16, pp. 103–122, 2000.

[20] M. Ankerst, G. Kastenm$\ddot{u}$ller, H.-P. Kriegel, and T. Seidl, “3D shape histograms for similarity search and classification in spatial databases,” in *International Symposium on Advances in Spatial Databases*, 1999, pp. 207–226.

[21] D. A. Keim, “Efficient geometry-based similarity search of 3D spatial databases,” in *SIGMOD*, 1999, pp. 419–430.

[22] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs, “A search engine for 3D models,” *ACM Trans. Graph.*, vol. 22, no. 1, pp. 83–105, 2003.

[23] T. Zaharia and F. Pr$\hat{e}$teux, “Shape-based retrieval of 3D mesh models,” in *ICME*, 2002.

[24] L. Holm and C. Sander, “Mapping the protein universe,” *Science*, vol. 273, pp. 595–603, 1996.

[25] S. Chakraborty and S. Biswas, “Approximation Algorithms for 3-D Common Substructure Identification in Drug and Protein Molecules,” in *Workshop on Algorithms and Data Structures*, 1999, pp. 253–264.

[26] V. Choi and N. Goyal, “An Efficient Approximation Algorithm for Point Pattern. Matching Under Noise,” in *LATIN*, 2006, pp. 298–310.

[27] C. Bron and J. Kerbosch, “Algorithm 457 - finding all cliques of an undirected graph,” *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[28] D. Mount, *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, 2001.

[29] A. Efrat and A. Itai, “Improvements on bottleneck matching and related problems using geometry,” in *Twelfth Annual Symposium on Computational Geometry*, 1996, pp. 301–310.

[30] 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.” *TODS*, vol. 30, no. 2, pp. 364–397, 2005.

[31] R. Weber, H. Schek, and S. Blott, “A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces.” in *VLDB*, 1998, pp. 194–205.

[32] N. Koudas, B. C. Ooi, H. T. Shen, and A. Tung, “LDC: Enabling Search By Partial Distance In A Hyper-Dimensional Space,” in *ICDE*, 2004, pp. 6–17.

[33] M. E. Houle and J. Sakuma, “Fast approximate similarity search in extremely high-dimensional data sets,” in *ICDE 2005*, 2005.

[34] A. Andreeva, D. Howorth, C. Brenner, T. J. Hubbard, C. Chothia, and A. G. Murzin, “Scop database in 2004: refinements integrate structure and sequence family data,” *Nucleic Acids Res.*, vol. 32, pp. 226–229, 2004.

[35] “Scop: Structural classification of proteins,” in *http://scop.mrc-lmb.cam.ac.uk/scop/*, 2006.

[36] X. Wang and J. T. L. Wang, “Fast similarity search in three-dimensional structure databases,” *J. Chem. Inf. Comput. Sci.*, vol. 2, pp. 442–451, 40.

PLACE PHOTO HERE

Zi Huang Zi Huang is a Research Fellow in School of Information Technology and Electrical Engineering, University of Queensland. She obtained his BSc from Tsinghua University, China and PhD from School of Information Technology and Electrical Engineering, University of Queensland in 2007. Her research interests include bioinformatics, multimedia search, spatial database, knowledge discovery, and natural language processing.

PLACE PHOTO HERE

Heng Tao Shen Heng Tao Shen is currently a Senior Lecturer in School of Information Technology and Electrical Engineering, University of Queensland. He received MOE (Ministry of Education) Scholarship, Singapore in 1995 and began his undergraduate study in NUS (National University of Singapore) in 1996. Heng Tao obtained his BSc (with 1st class Honors) and PhD from Department of Computer Science, NUS in 2000 and 2004 respectively, and then joined University of Queensland in June 2004. His research interests include Database, Multimedia/Web search, Mobile/Web information system, Peer-to-Peer computing, etc. Heng Tao has served as a PC member for many international conferences including VLDB, ICDE, EDBT, SAC, DASFFA, publication chair for APWEB.

PLACE PHOTO HERE

Xiaofang Zhou Xiaofang Zhou is a Professor of Computer Science at the University of Queensland. He is the Head of the Data and Knowledge Engineering Research Group (DKE). He is also the Convenor of ARC Research Network in Enterprise Information Infrastructure (EII), and a Chief Investigator of ARC Centre of Excellence in Bioinformatics. Professor Zhou received his BSc and MSc degrees in Computer Science from Nanjing University in 1984 and 1987 respectively, and PhD in Computer Science from the University of Queensland in 1994. From 1994 to 1999, he worked as a Senior Research Scientist in CSIRO, leading its Spatial Information Systems group. His research focuses on finding effective and efficient solutions to managing, integrating and analyzing very large amount of complex data for business and scientific applications. His research interests include spatial and multimedia databases, data quality, high performance query processing, Web information systems and bioinformatics.