Approximation of Protein Structure for Fast Similarity Measures

Itay Lotan[1]

Department of Computer Science

Stanford University

Stanford, CA 94305

itayl@stanford.edu

Fabian Schwarzer

Department of Computer Science

Stanford University

Stanford, CA 94305

schwarzf@robotics.stanford.edu

December 5, 2003

[1]To whom correspondences should be addressed

Abstract

The structural comparison of two proteins comes up in many applications in structural biology, where it is often necessary to find similarities in very large conformation sets. This work describes techniques to achieve significant speedup in the computation of structural similarity between two given conformations, at the expense of introducing a small error in the similarity measure. Furthermore, the proposed computational scheme allows for a tradeoff between speedup and error. This scheme exploits the fact that the Cα representation of a protein conformation contains redundant information, due to the chain topology and limited compactness of proteins. This redundancy can be reduced by approximating sub-chains of a protein by their centers of mass, resulting in a smaller number of points to describe a conformation. A Haar wavelet analysis of random chains and proteins is used to justify this approximated representation. Similarity measures computed with this representation are highly correlated to the measures computed with the original Cα representation. Therefore, they can be used in applications where small similarity errors can be tolerated or as fast filters in applications that require exact measures. Computational tests have been conducted on two applications, nearest neighbor search and automatic structural classification.

Key words: Protein structure, similarity measures, structural alignment, random chains, nearest-neighbor search

1 Introduction

Automatic protein structure comparison is an important problem in computational structural biology, e.g., in structural databases and classification [12, 16, 29, 31], structure prediction [8, 9, 23, 33, 37, 38], analysis of trajectories through conformational space generated by molecular dynamics and Monte Carlo simulations [26, 36, 43], graph-based methods for evaluating ensemble properties [2, 3, 39], etc.

In contrast to sequence matching, structural matching requires a similarity measure that is based on spatial atom coordinates. Nevertheless, it is still important whether the structures that are compared have the same amino acid sequence or not. For conformational samples from the same sequence there are no ambiguities about correspondences. In this case, similarity measures such as *cRMS* or *dRMS* are commonly used [22]. These measures are defined as the root mean square (RMS) of either distances between corresponding atoms in the two compared structures (*cRMS*) or their corresponding intra-molecular distance matrix entries (*dRMS*). Comparing protein structures that derive from different sequences is more difficult because it is generally not obvious which features (atoms) of one structure should be matched with which features from the other structure. Moreover there is a tradeoff between the length of the correspondence that is chosen and the quality of the similarity that results. Numerous methods for finding a set of correspondences have been proposed, e.g. [10, 11, 13, 15, 22, 25, 35, 42].

Many similarity measures are based on a rather fine granularity feature selection. Typically, the coordinates of all Cα atoms and sometimes even those of additional atom centers are considered. For large proteins, the number of considered features greatly affects the efficiency of the structural comparison. For example, the size of the intra-molecular distance matrices and the complexity of the dynamic programming

algorithm in [11] are quadratic in the number of residues. While this is not a real problem when comparing a single pair of structures (most proteins have less than 1000 residues), it becomes an important issue when querying a database for similar structures or clustering a large set of structures.

In this paper, we show that the description size of a protein conformation can be reduced while introducing only a small error in the computed similarity measure. We uniformly sub-divide the backbone into contiguous sub-chains of fixed length and represent each of the sub-chains by the average coordinates of its atom centers. Similarity measures can then be computed on these "averaged conformations". Although this simplification introduces some error, we provide theoretical and experimental evidence that enough information about relative similarity is retained to discriminate structures in practical applications. In particular, the derived similarity measures using the averaged protein representation are highly correlated to their original full-atomic counterparts. If high accuracy is a concern, approximate similarity measures are still useful as a fast filter to considerably reduce the number of pairs that need to be passed to the exact similarity measure.

While we cannot give bounds on the error that is introduced, we show through wavelet analysis of protein structures and random chains that averaging is a reasonable method for reducing the dimensionality of structure descriptors. Our analysis reveals two properties of proteins, which makes averaging work. The first property is the chain topology of proteins, which forces atoms that are nearby in the chain to also be spatially close in any conformation the protein chain assumes. The second property is the fact that van der Waals forces limit the compactness of the protein structures. Since atoms can overlap only by a little, on average the positions of atoms, which are far apart along the chain, will also be spatially distant.

Reducing the computational complexity of similarity measures significantly accelerates many tasks that involve structural matching. In our experiments we observed decreases in running times by large factors, typically from days to hours or even minutes. For very large sets of proteins, both the efficiency of structure comparison of a single pair and the number of such pairs that are actually evaluated are important. Many

approaches require evaluation of all pairs ("brute-force approach") even if the task is to identify only a small constant number of nearest neighbors for each conformation in the set. Using our averaged representation we show that we can avoid this quadratic cost of examining all pairs in a k nearest-neighbor application. In automatic structural classification, the complexity of a previous algorithm [11, 41] that matches pairs of structures grows quadratically with the number of residues. In this case as well, a small reduction in the number of features results in substantial savings.

In Section 2, we describe in detail and analyze the simplified representation of proteins used to approximate similarity measures. In Section 3, we demonstrate our approach in a k nearest-neighbor application on a large set of conformations of the same protein. In Section 4, we use our approach to significantly speed up the STRUCTAL algorithm [11, 41] for classification of structures with different sequences.

2 Shape similarity and approximation of chains

Given two sequences of points in 3-space $P = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ and $Q = (\mathbf{q}_1, \ldots, \mathbf{q}_n)$, their coordinate root mean square deviation (*cRMS*) is a common measure of similarity. It is defined as

$$cRMS(P,Q) = min_T \sqrt{\frac{1}{n}\sum_{i=1}^{n} \parallel \mathbf{p}_i - T\mathbf{q}_i \parallel^2} \tag{1}$$

where $\parallel \cdot \parallel$ is the Euclidean L_2-norm and T is a rigid body transform (rotation and translation). A closed form solution for T yields the optimal transform [17, 19].

Another common RMS shape similarity measure, *dRMS*, is based on comparing intra-set point distance matrices, i.e. the matrix of distances between all points within each structure. For a point set P, this matrix is defined as

$$\left(d_{ij}^P\right) = \|\mathbf{p}_i - \mathbf{p}_j\|\,. \tag{2}$$

The distance matrix RMS deviation (*dRMS*) of P and Q is then defined as

$$dRMS(P,Q) = \sqrt{\frac{2}{n(n-1)}\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d_{ij}^P - d_{ij}^Q)^2}. \tag{3}$$

When applying these similarity measures to proteins, it is common practice to use the Cα atom centers, ordered along the backbone, as defining points. (Sometimes, additional atoms are included or C$_\beta$ atoms are used instead.) Due to the special geometry of proteins the positions of these atoms completely determine the shape of the backbone. However, in the case of *dRMS*, the intra-molecular distance matrices grow quadratically with the length of the protein, which significantly slows down the *dRMS* computation for large proteins.

2.1 Approximate similarity measures

We reduce the number n of sample points in P and Q as follows. In each sequence, we replace contiguous subsequences of points by their centroids. That is, we uniformly partition the sequence P of length n into contiguous subsequences of length m each. (If n/m is not an integer, some subsequences will be chosen to be longer by one.) For each subsequence, we then replace its points by their centroid, which we denote by $\overline{\mathbf{p}}_j$ for subsequence j. For example, if subsequence j spans points $(\mathbf{p}_a, \ldots, \mathbf{p}_b)$ where $b - a + 1 = m$ then

$$\overline{\mathbf{p}}_j = \frac{1}{m}\sum_{i=a}^{b}\mathbf{p}_i. \tag{4}$$

Based on these averaged subsequences we define the m-averaged representation $\overline{P}_m$ of P as the sequence of $r = \lfloor n/m \rfloor$ points $(\overline{\mathbf{p}}_1, \ldots, \overline{\mathbf{p}}_r)$. (For Q, we proceed in the same way.) We can now define

the simplified RMS measures for $\overline{P}_m$ and $\overline{Q}_m$ analogously to the above RMS measures on the original sequences. That is, in the defining formulas (Equations 1, 2 and 3), we replace $\mathbf{p}_i$ ($\mathbf{q}_i$) by $\overline{\mathbf{p}}_i$ ($\overline{\mathbf{q}}_i$) and n by r. We call these measures m-*averaged measures* and denote them by $\overline{c}_m RMS$ and $\overline{d}_m RMS$.

Obviously, the error of these simplified similarity measures continuously approaches zero as the two compared point sets P and Q become more similar, i.e., $\lim_{Q \to P} |\overline{c}_m RMS(P,Q) - cRMS(P,Q)| = 0$ and the same holds for $\overline{d}_m RMS$. For general point sets, the error introduced by this approximation can be quite substantial. However, for proteins the error is small because of their chain topology and the limited compactness of their conformations(due to van der Waals forces). In the following section we give an intuition as to why this is the case using random chains and wavelets analysis.

2.2 Random chains and Haar wavelets

We will use random chains and the Haar wavelet transform to argue that averaging is a reasonable method for reducing the size of the representation of a protein for computing structural similarities.

A random chain $C = (\mathbf{c}_0, \ldots, \mathbf{c}_{n-1})$ in 3-D is an ordered set of points in space defined as follows:

$$\begin{aligned} \mathbf{c}_0 &= \mathbf{0}, \\ \mathbf{c}_{i+1} &= \mathbf{c}_i + \mathcal{S}_2 \cdot l \quad i = 0, \ldots, n-2 \end{aligned} \tag{5}$$

where $\mathcal{S}_2$ is a random 3-D vector uniformly distributed on the unit 3-D sphere and l is the fixed Euclidean distance between two consecutive points of the chain. $\mathcal{S}_2$ is sampled as follows:

$$\mathcal{S}_2 = \begin{bmatrix} \sin\phi\cos\theta \\ \sin\phi\sin\theta \\ \cos\phi \end{bmatrix} \tag{6}$$

where $\theta \sim U[0, 2\pi]$ and $\cos\phi \sim U[-1, 1]$.

Computing the covariance matrix of $\mathcal{S}_2$ reveals that the off-diagonal elements are identically 0, and as a result the three dimensions of each random step are uncorrelated. Since each step is independent of all other steps the distributions of the three dimensions of any point on the chain are uncorrelated. Consequently, the random chain as described above can be approximated well by replacing $\mathcal{S}_2$ with a 3-vector sampled from the normal distribution $\mathcal{N}(\mathbf{0}, \frac{1}{3} \cdot \mathcal{I})$, where $\mathcal{I}$ is the 3×3 identity matrix, for sufficiently large values of n (typically greater than 10). Moreover, the fact that the three dimensions are uncorrelated enables us to perform three independent one-dimensional Haar wavelet transforms as described below instead of the more complicated three-dimensional transform.

The Haar wavelet transform of a chain is a recursive averaging and differencing of the coordinates of the points. The transform recursively smoothes the chain while keeping the *detail* coefficients needed to reconstruct the full chain from the smoothed out version. We define the full resolution chain to be of level 0: $C = C^0$. We recursively create smoothed versions of the chain by averaging pairs of consecutive points:

$$\mathbf{c}_i^j = \frac{1}{\sqrt{2}} \left(\mathbf{c}_{2i}^{j-1} + \mathbf{c}_{2i+1}^{j-1} \right) \quad \begin{cases} j = 1, \ldots, \log n \\ i = 0, \ldots, \frac{n}{2^j} - 1 \end{cases} . \tag{7}$$

As each level of resolution is created we also compute the *details* that are smoothed out by the averaging:

$$\mathbf{d}_i^j = \frac{1}{\sqrt{2}} \left(\mathbf{c}_{2i}^{j-1} - \mathbf{c}_{2i+1}^{j-1} \right) \quad \begin{cases} j = 1, \ldots, \log n \\ i = 0, \ldots, \frac{n}{2^j} - 1 \end{cases} . \tag{8}$$

Note that the averages and details are multiplied by a scale factor of $\sqrt{2}$ at each level. Given C^j, the smoothed chain at level j, and D^j, the detail coefficients of level j, it is possible to reconstruct C^{j-1} exactly by inverting the formulas of Equations 7 and 8. The Haar wavelet transform of a chain C is thus defined as:

$$\hat{C} = \left(C^{\log n}, D^{\log n}, D^{\log n - 1}, \ldots, D^1 \right) . \tag{9}$$

The length of $\hat{C}$ is the same as C and $C^{\log n}$ is the centroid of the entire chain. Since C can be exactly reconstructed from $\hat{C}$, no information is lost during the transform. This representation can then be com-

pressed by setting to 0 all coefficients in $\hat{C}$ smaller than some threshold. Since the coefficients are scaled, the square of the L_2 error of approximation in this case would be equal to the sum of the squares of the removed coefficients [40].

Given the normal approximation of the random chain construction, we can analytically determine the pdf (probability density function) of each of the coefficients in $\hat{C}$ by adding, subtracting and scaling independent normally distributed variables. The pdf of each detail coefficient in level j is:

$$\mathbf{d}^j \sim \mathcal{N}\left(\mathbf{0}, \frac{4^j+2}{36} \cdot \mathcal{I} \cdot l\right), \tag{10}$$

and the pdf of their squared L_2 norm is thus:

$$\|\mathbf{d}^j\|_2^2 \sim \chi^2(3dof) \cdot \frac{4^j+2}{36} \cdot l \tag{11}$$

with a mean of $\frac{4^j+2}{12} \cdot l$ and variance of $\frac{(4^j+2)^2}{216} \cdot l^2$. Since the pdfs of the detail coefficients are centered at the origin and their variance is decreasing by a factor of roughly 4, they are expected to be ordered (in absolute value) in $\hat{C}$, from largest to smallest. More precisely, the detail coefficients of level i are expected to be larger by roughly a factor of 4 than the detail coefficients of level $i-1$, and this holds true for all levels of the transform. Note that the $\sqrt{2}$ scaling during the construction of the coefficients would account for an average growth of a factor of 2 in their variance from one level to the next. The special geometry of random chains accounts for the second factor of 2. Hence, as a general policy, it is best to remove coefficients starting at the lowest level and climbing up. These coefficients have the lowest variance and thus contain the least information for determining structural similarity. The effect of averaging as described in Section 2.1 for $m = 2^v$ is the same as that of removing the lowest v levels of coefficients. Since these are expected to be the smallest coefficients, we can conclude that using an m-averaged chain will give the smallest expected error for a representation that uses only m Haar detail coefficients for each dimension. The wavelet analysis allows us to estimate the approximation mean squared (MS) error introduced by removing all coefficients of

level j. It can be computed to be $\frac{l}{12}\left(2^i + \frac{1}{2^{i-1}}\right)$. Therefore for an m-averaged approximation the MS error is expected to be on the order of $(m \cdot l)/6$.

As the above analysis shows, we can remove quite a few levels of coefficients without introducing a large error. This is due to the large *ratio of the average variances* (henceforth called RAV) between two successive levels of detail coefficients, which was shown to be approximately 4 for all levels. This behavior of the Haar detail coefficients is a result of the fact that on the average random chains grow further and further away from their starting point. The expected distance of the nth point from the origin is on the order of $\sqrt{n} \cdot l$. We see this behavior in Figure 1 where we compare the average variances of the coefficients of random chains to those of very compact random chains (chains forced to reside inside small spheres) and to those of point clouds sampled randomly from inside a sphere of radius $\sqrt{n} \cdot l$. All chains are of length $n = 64$. The RAV of the unconstrained random chain is significantly larger than that for the compact random chains, which decreases as the compactness of the chain is increased. The worst case is for a point cloud, where the average variances of all levels of detail coefficients have the same magnitude making all levels have the same importance.

2.3 Application to Proteins

While it is a well-known fact that the positions of neighboring Cα atoms along the backbone of native protein structures are highly correlated (e.g., see [24]), it was shown in [20] that treating the position of each Cα atom as uniformly distributed on a 3-D sphere centered at the center of the previous Cα atom yields a very good approximation of the average global behavior of native protein structures.

We performed the same wavelet transform on sets of conformations of actual proteins of length 64 residues (only the Cα atoms coordinates were used to describe the conformations) taken from *decoy sets* (conformations which are expected to be similar to the native conformation) generated by Park and Levitt [33,

32], containing 10,000 conformations each. We obtained results similar to those of random chains, namely the detail coefficients are ordered and have a RAV similar to that of random chains. The results for a few sets of proteins are presented in Figure 2. One important difference from random chains is that the RAV for decoy sets decreases considerably for the top levels. We explain this as follows. Small pieces of a protein cannot be highly packed because of steric clashes, while intermediate size pieces are often elongated (secondary structure helices and strands) and hence the variance of the coefficients grows considerably from one level to the next at the low and intermediate levels. The tight packing of the secondary structure of the native-like conformations makes the high-level coefficients considerably smaller than in the random chain model. We would have liked to give results for decoy sets of proteins significantly longer than 64 residues, however, no such sets were available to us.

Random protein conformations (generated as described in Section 2.4), on the other hand, are considerably less compact than the decoy sets, and hence behave much more like random chains at *all* levels of detail coefficients. As can be seen in Figure 2 the RAV at the lower levels is even bigger than what is observed for random chains. This is due to the limit on the packing density as a result of the space taken up by each residue. In our random chain model the points have no volume and the chain is allowed to cross itself. In the random protein conformations, however, atoms are not allowed to have any self-overlaps. Their behavior is actually modelled better by random chains in which the next step is sampled uniformly from a *hemisphere* defined by the direction of the previous step.

We thus conclude that, while decoy sets cannot be compressed as much as random sets, it is possible to remove the first few levels and still get a very good approximation.

2.4 Correlation of approximate and exact similarity measures

When using the *cRMS* measure to compute similarity between random chains of length 64 we find that the approximate m-averaged versions yield very good results for m as large as 9. For $m = 4, 6, 9, 12$ and 16, the correlation of the approximate measure to the true one is 0.99, 0.95, 0.89, 0.76 and 0.60, respectively. When using the *dRMS* measure to compute similarity between random chains the approximate m-averaged versions is highly correlated for m as large as 6. The correlation values obtained are 0.94, 0.81, 0.67, 0.55 and 0.46, respectively.

In order to verify that the analogous behavior of the detail coefficients of protein sets and random chains carries over to approximate similarity measures we chose 8 structurally diverse proteins used by Park and Levitt in [32, 33] (1CTF, 1R69, 1SN3, 1UBQ, 2CRO, 3ICB, 4PTI, 4RXN) having between 54 (4RXN) and 76 (1UBQ) residues. For these proteins we obtained (1) decoy sets generated by Park and Levitt [32, 33], containing 10,000 conformations each and (2) randomly generated conformation sets using the program FOLDTRAJ[1] [9], containing 5000 structures each. A decoy set is a collection of conformations, which are native-like in structure. As a result they span only a small portion of the entire conformation space and are thus expected to have significant pairwise similarity. On the other hand, the conformations in the random sets sample a much larger space and as a result have on average lower pairwise similarity.

For each set, we randomly chose between 1000 and 4000 pairs whose true *dRMS* distance was less than 5Å (10Å for the random sets, which have larger structural diversity) and computed their m-averaged distances for different values of m. The correlation of the m-averaged *cRMS* and *dRMS* measures to the true *cRMS* and *dRMS* measures for the different decoy sets can be found in Table 1a. For m as large as 9, the approximate *cRMS* measure is still highly correlated with the true *cRMS* measure, which means that a reduction factor of 9 still yields a very good approximation for proteins of this size. For *dRMS*, high

[1] `http://bioinfo.mshri.on.ca/trades/`

correlation is still achieved when $m = 6$, which means a reduction factor of 6 (since the complexity of *dRMS* is quadratic, the actual gain is by a factor of 36). We note that the correlation values obtained are quite similar to those computed for random chains.

The correlation of the m-averaged *cRMS* and *dRMS* measures to the true measures for the different random sets can be found in Table 1b. Here too, a reduction factor of 9 still yields a highly correlated approximation of the *cRMS* measure. For *dRMS*, high correlation is still achieved when $m = 9$, which means a reduction factor of 9 and an actual gain of 81. Here the correlation values are in fact better than those computed for random chains as would be anticipated from the higher growth ratio of the variance of the detail coefficients of random protein sets in comparison to those of random chains (see Figure 2). When examining all pairs and not only those whose *dRMS* distance is smaller than 5Å (10Å for the random sets), the computed correlation is even larger.

The analysis of random chains in the previous section entails that the approximation error caused by averaging depends only on the averaging factor m. The better correlation we get for the random sets over the decoy sets thus requires some explanation. First, the larger RAV of the random sets means that the low-level coefficients are less important than those in decoy sets. Second the distance between a pair of random conformations is larger than the distance between a pair of decoy conformations and as a result the approximation errors have a smaller effect on the computation of similarity.

We wanted to see how well averaging scales up as we increase the size of the protein. As we did not have access to decoy sets of larger proteins we examined only random conformation sets. We generated 1000 random conformations for the two proteins 1LE2 (144 residues) and 1HTB (374 residues). We chose 1100 pairs of 1LE2 conformation with a *dRMS* less than 18Å, and 1100 pairs of 1HTB conformations with *dRMS* less than 24Å. For 1LE2 the correlation of *cRMS* was above 0.9 for m as large as 27, and for *dRMS* it was above 0.9 for m as large as 15. For 1HTB the correlation of *cRMS* was above 0.9 for m as large as

40, and for *dRMS* it was above 0.9 for m as large as 30. Because the pairwise distances for these larger proteins were significantly larger than the distances of the smaller proteins, a larger approximation error can be tolerated without compromising performance. Thus larger values of m could generally be used for larger proteins, making the speedups that can be achieved larger when they are most needed.

3 Application 1: Nearest-neighbor search

Simulations and other conformational sampling methods generate large sets of conformations of a particular protein. For example, the project Folding@Home[2] runs parallel molecular dynamics simulations on thousands of computers across the Internet and then centrally analyzes the obtained data. An important step in evaluating such data, e.g. for clustering and related tasks, is the following: given a set of conformations of the same protein, find the k nearest neighbors (NNs) for each sample in the set. Typically, k is a small constant while the size of the set can be very large.

The straightforward ("brute-force") approach is to evaluate the similarity measure (*cRMS* or *dRMS*) for all pairs and then report the k NNs for each sample. However, the quadratic complexity makes this approach scale poorly. Spatial data structures such as the *kd-tree* [4] can avoid this complexity under certain circumstances [1, 7, 18, 21, 28, 34]. Note that these data structures allow for exact search, i.e., they return the same NNs as would the brute-force search. However, most of them require a Euclidean metric space of rather small dimensionality. Unfortunately, *cRMS* is not a Euclidean metric. Although *dRMS* is a Euclidean metric, the dimensionality of the space of intra-molecular distance matrices is far too high, since typically, for dimensions higher than a few tens, none of the nearest-neighbor data structures performs better than brute-force search. Therefore, if we hope to use a spatial data structure to speed up NNs search, we must use the *dRMS* measure, but find a way to significantly reduce the dimensionality of the structure descriptors

[2] `http://folding.stanford.edu`

below the averaged conformations we presented in Section 2.

3.1 Further reduction of distance matrices

We use the singular value decomposition (SVD) [14] to further compress the intra-molecular distance matrices of averaged proteins, that is to further reduce the number of parameters involved in computing $\overline{d}_m RMS$. SVD is a standard tool for principal components analysis (PCA) and computes directions of greatest variance in a given set of high-dimensional points. When considering the set of intra-molecular distance matrices these directions correspond to directions that contain the most dissimilarity information. These directions are called principal components (PCs). The SVD can be used to linearly map a set of high-dimensional input vectors (data points), stored in a matrix A, into a lower-dimensional subspace while preserving most of the variance. Such a transform can be found by decomposing the matrix A of the input vectors into $A = USV^T$, the SVD of A, where U and V are orthogonal matrices and S is diagonal with the singular values of A along the diagonal.

Our reduction algorithm has two steps:

1. Compute an m-averaged representation of each conformation in the set.

2. Use the SVD of the entire set to further reduce the description length of each conformation.

Efficient algorithms exist that compute the SVD in time $O(s^2t)$ where s is the smaller and t the larger dimension of A (rows or columns). Note that while in principle, SVD could be applied to intra-molecular distance matrices without first averaging sub-chains, the quadratic dependency on the smaller dimension of A shows the important advantage of averaging: usually, the larger dimension t will reflect the size of the conformational sample set while the smaller dimension s will correspond to the size of a single intra-molecular distance matrix. Reducing the distance matrix size by using averaged conformations, as described in Section 2, is therefore key to performing SVD in practice.

To perform SVD on a set of intra-molecular distance matrices derived from m-averaged conformations, each of these distance matrices is rewritten as an $r(r-1)/2$ dimensional vector (where $r = \lfloor n/m \rfloor$) and then SVD is applied to the matrix that contains all these vectors. Taking the resulting U matrix and removing all columns that correspond to small singular values (the directions of little variance), we have the linear map that takes the set of distance matrices into a lower-dimensional Euclidean space while preserving a high percentage of the variance and thus distance information. We found that in practice, a relatively small output dimensionality between 12 and 20 is sufficient to maintain about 90% of the variance of the distance matrices. In what follows, we will denote the *dRMS* measure obtained from an SVD compressed set of m-averaged distance matrices by $\overline{d}_m^{PC}RMS$. (PC stands for the number of principal components that are used after compression.)

Since the cost of the SVD is quadratic in the length of the protein (assuming a very large set of conformations), its dependence on m, the averaging factor, is quartic. Therefore, one would like to use a large m when averaging. On the other hand, averaging is lossy, and thus it would be best to apply SVD to a representation that has the least amount of averaging possible. Figure 3 shows the effectiveness of the SVD for reducing the description length of protein conformations. Figure 3a presents the correlation between *dRMS* and $\overline{d}_4^{PC}RMS$ on some of the decoy sets as the number of PCs that is used increases. With as little as 16 PCs a correlation of 0.9 is achieved. Figure 3b presents the correlation on some of the random sets as the number of PCs that is used increases. Here as little as 12 PCs suffice to get a correlation of 0.9. For comparison we present in Figure 4 the correlation when SVD is applied after averaging with a factor of $m = 9$. Here the same number of PCs yields lower correlation than was achieved for $m = 4$. The computation time would be more than 16 times faster though.

3.2 Evaluation of approximation errors for nearest-neighbor search

Using an approximate measure to find the k NNs of a conformation entails that some true NNs will be missed while other conformations would be chosen instead, which are not true NNs. As described above the approximate measure we propose to use is $\overline{d}_m^{PC} RMS$. We tested the usefulness of this measure on both the decoy sets and the random sets used in Subsection 2.4.

Given a set of conformations S and a query conformation q from that set, we define two orderings of the elements in S: S_1 and S_2. S_1 is the ordering of the conformations in S by their exact *dRMS* distance from q. S_2 is ordered by the $\overline{d}_m^{PC} RMS$ distance of the conformations in S from q. Thus, S_2 is the approximation of S_1 using our reduced similarity measure. In our test we used $m = 4$ for both random and decoy sets. We used 16 PCs for the decoy sets but only 12 for the random sets. Based on these two sets we compute two error measures and two parameters to help evaluate the performance of the approximate similarity measure for the k NNs application.

The two error measures we use are:

$\mathbf{err_1}$ Given the *dRMS* distance of the kth conformation in S_2 (the approximate kth NN) and the *dRMS* distance of the kth conformation in S_1 (the true kth NN), how much is the former larger than the latter. This error is reported as percentage of the *dRMS* distance of the kth conformation in S_1.

$\mathbf{err_2}$ Given the average *dRMS* distance of the first k conformations in S_2 (the average distance of an approximate NN) and the average *dRMS* distance of the first k conformations in S_1 (the average distance of a true NN), how much is the former larger than the latter. This error is reported as percentage of the average *dRMS* distance of the top k conformations in S_1

We also computed two other parameters that are indicative of how well the approximate measure finds true NNs:

NN1 The number of elements among the top k of S_1 found in the top k of S_2. In other words, the number of true NNs found by the approximate measure.

NN2 The maximal index in S_2 of an element from the top k in S_1. In other words, how many approximate NNs do we need to find in order to have all the true NNs.

We first looked at the decoy sets of size 10,000 for each of the eight proteins. We used $m = 4$ and 16 PCs ($\bar{d}_4^{16} RMS$ measure)to find approximate NNs. For each set, we randomly chose 250 query conformations and evaluated err_1, err_2, *NN1* and *NN2* for each of them. Table 2 shows the mean values of these parameters over the 250 queries (standard deviations were generally small).

The err_1 error was about 15% for all proteins, which translates to an error of no more than 1.5Å in the furthest NN that is returned. The err_2 error is always smaller than 7%, which means that on average the ith approximate nearest neighbor is no more than 0.7Å further away than the ith true NN. The *NN1* parameter reveals that at least 70% of the true NNs are returned by the approximate measure, and the *NN2* parameter indicates that almost always it is enough to find $3k$ approximate NNs in order to guarantee that all true k NNs are found.

Next we looked at the random sets, each containing 5000 conformations and performed the exact same calculations as we did for the decoys. However, here we used $m = 4$ and 12 PCs ($\bar{d}_4^{12} RMS$ measure) to find approximate NNs. The results are shown in Table 3. Here the err_1 error was less than 10%, which means the approximate furthest NN was less than 1.5Å further away than the true furthest NN. Note that the random sets cover a larger part of the conformation space and thus the distance between a pair of conformations would be larger than what we find for decoy sets, which span only a small portion of conformation space. The err_2 error is in general smaller than 3% which translates to about 0.5Å. While here the approximate method found somewhat less true nearest neighbors when $k = 10$ than for the decoy sets (*NN1* ≥ 6.5) its performance was equivalent to the decoy sets for $k = 25$ and superior (*NN1* ≥ 81) for $k = 100$. In general,

about $2.5k$ approximate NNs were enough to ensure that all true k NNs are found.

Finally, we looked at some larger sets of 100,000 conformations each. We ran the same test as we did for the smaller conformation sets. Approximate nearest neighbors were computed using the $\overline{d}_4^{16}RMS$ measure for the decoy sets and the $\overline{d}_4^{12}RMS$ measure for the random set. The results are displayed in Table 4. In general the results are slightly worse than what we observed for the smaller sets in terms of the error percentages and the number of true NNs found by the approximate measure. This is due to the fact that the SVD is less effective for the larger sets. Namely, more PCs are needed to capture well enough the variance of all the conformations. Indeed when we used $\overline{d}_4^{20}RMS$ instead of $\overline{d}_4^{16}RMS$ for the 1CTF decoys, the results were comparable to what we saw with the smaller decoy set (10,000 conformations). Still the *NN2* statistic is very promising, requiring only about $6k$ approximate NNs to capture all true k NNs.

3.3 Running time

We now consider the running times in a concrete nearest-neighbor task: given a set of $N = 100,000$ random conformations of protein 1CTF, find $k = 100$ nearest neighbors for each sample in the set. The reported times in this section refer to a sequential implementation in C running on a single 1GHz Pentium processor on a standard desktop PC.

We first compare the running times for a brute-force (all-pairs) implementation using both *cRMS* and *dRMS*, and their corresponding averaged similarity measures $\overline{c}_m RMS$ and $\overline{d}_m RMS$. All measures were used to find the $k = 100$ nearest neighbors for each of the N samples. Table 5 shows that the latter measures already result in a notable speed-up. For $\overline{d}_4 RMS$, a significant speed-up over *dRMS* is obtained due to the quadratic down-scaling of intra-molecular distance matrices by averaging proteins. For $\overline{c}_4 RMS$, the improvement over *cRMS* is smaller. This is because the reduction by averaging affects the number of involved points only linearly, and the main effort in computing *cRMS* comes from finding an optimal

rigid-body alignment of two point sets.

Note that the increase in running times agrees quite well with the expected quadratic scaling of the brute-force nearest neighbor approach. The times for $N = 100,000$ samples were therefore extrapolated from the actual running times measured for the smaller values of N. In fact, for *dRMS* using all Cα atom coordinates, we could not store all intra-molecular distance matrices. This problems does not occur with averaged proteins and $\overline{d}_m RMS$.

We next address the quadratic scaling problem of the brute-force approach. To be able to apply a kd-tree, we first further reduced the 120-dimensional space of $\overline{d}_4 RMS$ for 1CTF using SVD and retained 16 principal components. This further compression took about one minute for the complete set of 100,000 samples. Building the kd-tree for the resulting 16-dimensional data took only about 4 seconds.

We then ran both the brute-force and the kd-tree approach using $\overline{d}_4^{16} RMS$ as similarity measure. Table 6 shows the running times for finding $k = 1$ and $k = 100$ nearest neighbors for each sample in the full set of 100,000 samples. The obtained total speed-up of our NN search (approximate similarity measures and a kd-tree) over the current best approach (brute-force search using all Cα coordinates) is several orders of magnitude ($\sim$84 hours down to 19 minutes).

In general, the speed-up obtained by using a kd-tree can be expected to increase with increasing sample set size N. Due to its quadratic scaling, brute-force search will become very slow for larger sample sets. On the other hand, the sub-quadratically scaling kd-tree approach should allow to process even much larger sample sets within a few hours without parallelization on a standard desktop PC.

4 Application 2: structural classification

Given a set of native protein structures each having a different amino-acid sequence, such as the Protein Data Bank (PDB)[3] [5, 6], we would like to classify the structures into groups according to their structural similarity. This task has been performed manually in the SCOP (structural classification of proteins) database[4] [29], where protein structures are hierarchically classified into *classes*, *folds*, *superfamilies* and *families*. It has also been done semi-automatically in the CATH[5] [31] database, where protein structures are hierarchically classified into *classes*, *architectures*, *topologies* and *homologous superfamilies*. An automatic classification can be found in the FSSP (Fold classification based on Structure-Structure alignment of Proteins) database[6] [16], where the DALI program is used to classify all structures in the PDB into families of similar structures.

The major difficulty in automatically classifying protein structures lies in the need to decide, given two protein structures, which parts of both structures should be compared, before it can be determined how similar these parts are. There is an inherent trade-off between the length of the compared parts and the level of similarity that is found. The longer the compared parts the less similar the two structures will be, and vice-versa. This *correspondence problem* does not arise when different conformations of the same protein are compared because in that case the correspondence is trivially determined. For this reason, computing the similarity between structures of different proteins requires considerably more computation than the methods described in Section 2.

Several algorithms have been proposed for structural classification. Some of the more popular ones are briefly described here, for a survey of other methods see [22].

[3] `http://www.rcsb.org/pdb/`

[4] `http://scop.mrc-lmb.cam.ac.uk/scop/`

[5] `http://www.biochem.ucl.ac.uk/bsm/cath/`

[6] `http://www.ebi.ac.uk/dali/fssp/`

DALI [7] The internal distances matrices of both proteins are computed. Then all pairs of similar sub-matrices of small fixed size (one from each protein distance matrix) are found and a Monte Carlo algorithm is used to assemble the pairs into larger consistent alignments. For more details see [15].

PROSUP [8] Similar fragments are identified in both proteins. They are iteratively expanded to create alignments. A dynamic programming algorithm is then used to iteratively refine each alignment and finally insignificant alignments are removed. For more details see [25].

CE [9] Each structure is cut into small fragments and a matrix of all possible aligned fragment pairs (AFPs) is created. Combinations of AFPs are selectively extended or discarded leading to a single optimal alignment. For more details see [35].

STRUCTAL [10] The backbones of the two protein structures are directly matched by iteratively cycling between a dynamic programming algorithm that computes optimal correspondence given the current orientation of the two structures, and a least-square fitting that optimally orients the structures to minimize coordinate difference between the corresponding parts. For more details see [11, 41].

4.1 STRUCTAL and m-averaging

All the above methods stand to gain in performance by using our averaging scheme. In order to verify this claim, we tested the speedup and accuracy obtained by using the STRUCTAL method on averaged protein structures. We could not test our approach on PROSUP, DALI and CE because these servers did not accept

[7] `http://www2.ebi.ac.uk/dali`

[8] `http://lore.came.sbg.ac.at/CAME/CAME_EXTERN/PROSUP`

[9] `http://cl.sdsc.edu/ce.html`

[10] `http://bioinfo.mbb.yale.edu/align`

our averaged structures since they are not valid protein structures. These algorithms were too involved for us to implement ourselves reliably.

The STRUCTAL algorithm starts with an initial alignment of the backbone Cα atoms of the two structures according to one of a number of possible heuristics (aligning the beginnings, the ends, random segments, by sequence similarity, etc). Then a two step process is repeated until convergence. First a dynamic programming algorithm analogous to the Needleman and Wunsch sequence alignment algorithm [30] finds the correspondence between the two structures that yields the highest score. Scoring is based on assigning a cost to each possible corresponding pair, which is inversely proportional to the distance between Cα positions, and a gap penalty for every gap in the alignment (for more details see [11]). Computing the best correspondence thus requires $O(n_1 n_2^2 + n_2 n_1^2)$ time (n_1 and n_2 are the number of residues in each structure). Second, an optimal relative orientation is computed for the two structures, based on the previously computed correspondence, by using the method in [17, 19]. The score of the final correspondence is returned together with the *cRMS* distance of the corresponding parts and the length of the correspondence. Since the result is sensitive to the initial alignment, the algorithm is usually run a number of times for each pair of structures, each time using a different initial alignment. The results of the highest scoring run is kept.

STRUCTAL also computes a P value for the final score, which can be used to determine the statistical significance of the similarity that was computed. It gives the probability that a similarity of a given score and length could occur at random. As described in [27], this value is a function of the STRUCTAL alignment score and the length of the correspondence. It was computed based on structural comparison of the entire SCOP database. In what follows we will call a statistically significant alignment ($P < 0.005$) a *good* match and all other alignments *bad* matches.

We investigate the performance of STRUCTAL when used together with m-averaging to compute structural similarity among the native structure of different proteins in terms of the tradeoff between the gain

in speed and the amount of error that results. We use STRUCTAL as a black box and input m-averaged structures as if they were true protein structures. An m-averaged structure is created by cutting the protein into sub-chains of length m starting at its N-terminal and computing the average of all Cα atoms in each fragment. The sub-chain at the C-terminal could have between 1 and m Cα atoms. It is possible to shift the sub-chains by t residues, where $1 \leq t \leq (N \bmod m)$, and get a different m-averaged structure. The choice of t_1 and t_2 for a pair of m-averaged structures could affect the computed similarity score, however we expect this effect to be small. In what follows we use $t = 0$ (no shift) when computing all m-averaged structures. Since both n_1 and n_2 are reduced by a factor of m, the complexity of the dynamic programming, which is the main part of the STRUCTAL, is reduced by a factor of m^2. The complexity of the optimal orientation procedure is reduced by a factor of m.

4.2 Experimental results

In this section we evaluate our method based on how well it is able to pick out the good matches from the bad matches (as defined in the previous subsection). We cannot test m-averaging in this context by evaluating its performance in finding NNs since it is difficult to impose a strict ordering on a set of proteins, by measuring their similarity to one structure in the set. This is due to the aforementioned inherent tradeoff between the length of the correspondence and the quality of the similarity that is achieved for a given pair.

For our tests we used the latest version of STRUCTAL (courtesy of Michael Levitt). This program inputs two protein structures in the form of PDB files, and outputs their alignment and its score.

The set of structure pairs on which we tested our method was constructed as follows. We examined the clustering results based on STRUCTAL comparisons presented by Erik Sandelin on his PROTOFARM web site[11] and picked out 3,691 pairs of structures, most of which were found to be good matches. These pairs all

[11] `http://csb.stanford.edu/sandelin/protofarm/html/`

have a match correspondence of at least 150 Cα atom pairs and are taken from a set of 256 different protein structures of length between 180 and 420 residues each. We then randomly picked 6,375 more pairs from our set of 256 structures. Altogether we examined 10,066 pairs of structures, of which 4052 where found by STRUCTAL to be good matches and 6,014 were found to be bad matches. For each of the structures we generated an m-averaged approximation for $m = 3, 4$ and 5. Finally we ran STRUCTAL on the set of pairs using the 3 sets of averaged structures and recorded the results.

We restrict our analysis to the STRUCTAL score that is computed for each pair, since we do not have a method for computing P values for evaluating the statistical significance of a match between two m-averaged structures. Developing such a method would require in depth analysis of the distribution of scores of m-averaged pairs (see [27]), which is beyond the scope of this work.

In Figure 5 we show the structural comparison results in four graphs. Each depicts two histograms, one for the score distribution of the good matches and one for score distribution of the bad ones. The results are binned by the score of each aligned pair. Since the P value is based on both the score and the length of the correspondence (see [27] for details), the score alone is not enough to completely separate the two histograms even when using the full structures (Figure 5a). Thus some matches will inevitably be misclassified. Figures 5b, 5c and 5d show the histograms obtained for the m-averaged structures with $m = 3, 4$ and 5 respectively. One can still see that the bulk of the matches are classified well by the STRUCTAL score, with the amount of overlap between the histograms increasing as m grows larger.

for the full structures ($m = 1$) and the m-averaged structures ($m = 3, 4, 5$).

The exact numbers of misclassified pairs can be found in Table 7. The numbers are given for both the STRUCTAL P value cutoff of $P \leq 0.005$ and a more stringent cutoff of $P \leq 0.001$. These numbers are computed by finding the value of the STRUCTAL score that minimizes the amount of total misclassifications for each value of m. Less than 7% of the pairs are misclassified when we use m-averaging with $m = 3$, less

than 10% of the pairs are misclassified when $m = 4$ and about about 15% misclassified when $m = 5$. The number of false negatives clearly increases with m, since averaging may obfuscate the similarity. Moreover, the scoring function that was optimized for protein structures becomes less and less effective as the m-averaged structures grow less and less protein-like as m increases, and as a result matching parts can be missed. For the more stringent cutoff of $P \leq 0.001$ we observe in Table 7 that there is a considerable drop in the number of misclassified pairs compared to the first P value used. This indicates that a large number of misclassifications are borderline matches.

When precision is important, m-averaged structures can be used as a fast filter to quickly separate a small set of potentially good matches from the significantly larger amount of bad matches. Then the full structures can be used to weed out the false positives that snuck in. In this scenario the false negatives are the only true errors since the false positives can be removed in the second stage. Their number can be significantly reduced by shifting the cutoff score down. Thus the number of false negatives will be reduced at the cost of allowing more false positives. For example, when $m = 3$, it is possible to reduce the number of false negatives by 40% while increasing the total number of misclassified pairs by less than 40%.

Another indication that m-averaging introduces only a small error is the correlation between the STRUCTAL score using the full structures and the score using the m-averaged structures. It is 0.924 when $m = 3$, 0.908 when $m = 4$ and 0.85 when $m = 5$. This entails that the m-averaged scores are a good predictor of the true score and could also be used for classification into more than just 2 classes (good or bad matches).

4.3 Discussion

The test results presented in the previous subsection show the effectiveness of m-averaging for speeding up structural comparison using the STRUCTAL algorithm. It can be used both as a fast method for structural classification with a small amount of error, or as a fast filter that is able to remove most bad matches, leaving

only a small subset of pairs to be evaluated using full protein structures.

In this work we treated STRUCTAL as a black box. however, a scoring function that is better suited for m-averaged structures can be devised for STRUCTAL. Such a function would take into account the fact that the distance between neighboring points along the backbone chain in an m-averaged structure is different — in fact, larger — than the fixed 3.8Å between Cα atoms in real proteins.

We propose a hierarchical use of STRUCTAL with m-averaging, in which the value of m is decreased gradually. At first a high value is used to quickly remove the very bad matches. Then as m is decreased more and more bad matches are discovered and removed. In this scheme, the alignment that is computed for a pair of structures, for some value of m, could serve as an initial alignment for aligning the two structures at subsequent iterations with smaller values of m, in addition to the other heuristics that are used. Indeed, in most good matches that were detected using the m-averaged structures, the rotation and translation that was used for the alignment (the transformation that superimposes the two structures) was very similar to the one computed using the full structures.

Another good measure for structural alignment that can be used instead of the STRUCTAL score, is the ratio of the *cRMS* distance between the corresponding Cα atoms of the two structures to the length of the correspondence. This ratio is highly correlated with the STRUCTAL score and yields only a small increase in the number of misclassifications. It is also independent of the comparison algorithm that is used, and can be obtained from other alignment algorithms such as DALI or CE.

5 Conclusion

Two general properties of proteins, their chain topology and limited compactness, are exploited to uniformly reduce the number of features for structural similarity computations. Substantial savings in terms of storage and running time are attained with small errors.

In applications in which no approximation error is tolerable, our approach can be used as a first step to filter a small subset of pairs that are within some tolerance band around the desired similarity. More expensive exact similarity measures can then be used on the reduced set of pairs.

Two possible applications were presented: finding k nearest neighbors in large sets of conformations of the same protein and classification of different proteins using the STRUCTAL algorithm. For the first application the introduced error was low and the correlation to the true similarity measure was very high. For the second application the number of misclassified matches increased only slightly when using m-averaged structures. In both applications, the running times were reduced from days to hours or even minutes.

In general, applications that input very large sets of proteins or that employ computationally intensive algorithms on large proteins stand to benefit from approximation of structures as suggested in this paper.

6 Acknowledgments

The authors would like to thank Patrice Koehl, Pankaj Agarwal, Michael Levitt and Jean-Claude Latombe for comments, Patrice Koehl for providing the full Park-Levitt decoy set for the nearest-neighbor experiments and Michael Levitt for providing the latest version of STRUCTAL. This work was partially funded by NSF_ITR grant `DUKE UNI/01-SC-NSF-1009-01` and a grant from Stanford's Bio-X program.

References

[1] P. Agarwal. Range searching. In J. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 575–598. CRC Press, 1997.

[2] M. Apaydin, D. Brutlag, C. Guestrin, D. Hsu, and J.-C. Latombe. Stochastic conformational roadmaps for computing ensemble properties of molecular motion. In *Fifth International Workshop on Algorith-*

mic Foundations of Robotics (WAFR), Nice, France, Dec 2002.

[3] M. Apaydin, D. Brutlag, C. Guestrin, D. Hsu, J.-C. Latombe, and C. Varma. Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. *J. Comp. Biol.*, 10:257–281, 2003.

[4] J. Bentley. multidimensional binary search trees used for associative searching. *commun. ACM*, 18:509–517, 1975.

[5] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, and H. W. et al. The protein data bank. *Nucl. Acids Res.*, 28:235–242, 2000.

[6] F. C. Bernstein, T. F. Koetzle, G. Williams, D. J. Meyer, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The Protein Data Bank: a computer-based archival file for macromulecular structures. *J. Mol. Biol.*, 112:535–542, 1977.

[7] K. Clarkson. Nearest neighbor queries in metric spaces. In *Proc. Symp. Theo. Comp. (SToC)*, pages 609–617, 1997.

[8] B. Fain and M. Levitt. A novel method for sampling alpha-helical protein backbones. *J. Mol. Biol.*, 305:191–201, 2001.

[9] H. Feldman and C. Hogue. A fast method to sample real protein conformational space. *Proteins*, 39(2):112–131, 2000.

[10] Z. Feng and M. Sippl. Optimum superimposition of protein structures: ambiguities and implications. *Fold. Des.*, 1:123–132, 1996.

[11] M. Gerstein and M. Levitt. Comprehensive assessment of automatic structural alignment against a manual standard, the scop classification of proteins. *Prot. Sci.*, 7:445–456, 1998.

[12] J. Gibrat, T. Madej, and S. Bryant. Surprising similarities in structure comparison. *Curr. Opin. in Struct. Biol.*, 6(3):377–385, 1996.

[13] A. Godzik. The structural alignment between two proteins: is there a unique answer? *Prot. Sci.*, 5:1325–1338, 1996.

[14] G. Golub and C. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[15] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, 233(1):123–128, 1993.

[16] L. Holm and C. Sander. Mapping the protein universe. *Science*, 273:595–602, 1996.

[17] B. Horn. Closed form solution of absolute orientation using unit quaternions. *J. Optic. Soc. A*, 4(4):629–642, April 1987.

[18] P. Indyk. *High-dimensional Computational Geometry*. PhD thesis, Stanford University, September 2000.

[19] W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystal. A*, 34:827–828, 1978.

[20] K. Kedem, L. Chew, and R. Elber. Unit-vector RMS (URMS) as a tool to analyze molecular dynamics trajectories. *Proteins*, 37:554–564, 1999.

[21] J. Kleinberg. Two algorithms for nearest-neighbor search in high dimension. In *Proc. Symp. Theo. Comp. (SToC)*, pages 599–608, 1997.

[22] P. Koehl. Protein structure similarity. *Curr. Opin. Struct. Biol.*, 11:348–353, 2001.

[23] P. Koehl and M. Delarue. Building protein lattice models using self-consistent mean field theory. *J. Chem. Phys.*, 108(22):9540–9549, 1998.

[24] R. Kolodny, P. Koehl, L. Guibas, and M. Levitt. Small libraries of protein fragments model native protein structures accurately. *J. Mol. Biol.*, 323(2):297–307, Oct 2002.

[25] P. Lackner, W. Koppensteiner, M. Sippl, and F. Domingues. ProSup: a refined tool for protein structure alignment. *Prot. Eng.*, 13(11):745–752, 2000.

[26] S. Larson, C. Snow, M. Shirts, and V. Pande. Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology. In R. Grant, editor, *Computational Genomics*. Horizon Press, 2003.

[27] M. Levitt and M. Gerstein. A unified statistical framework for sequence comparison and structure comparison. *Proc. Nat. Acad. Sci.*, 95:5913–5920, 1998.

[28] S. Maneewongvatana and D. Mount. The analysis of a probabilistic approach to nearest neighbor searching. In *Proc. Workshop Algo. Data Struct. (WADS)*, pages 276–286, 2001.

[29] A. Murzin, S. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247(536–540), 1995.

[30] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[31] C. Orengo, A. Michie, S. Jones, D. Jones, M. Swindells, and J. Thornton. CATH– a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1108, 1997.

[32] B. Park, E. Huang, and M. Levitt. Factors affecting the ability of energy functions to discriminate correct from incorrect folds. *J. Mol. Biol.*, 266:831–846, 1997.

[33] B. Park and M. Levitt. Energy functions that discriminate X-ray and near-native folds from well-constructed decoys. *J. Mol. Biol.*, 258:367–392, 1996.

[34] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[35] I. Shindyalov and P. Bourne. protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Prot. eng.*, 11(9):739–747, 1998.

[36] M. Shirts and V. Pande. Mathematical analysis of coupled parallel simulations. *Phys. Rev. Lett.*, 86(22):4983–4987, 2001.

[37] D. Shortle, K. Simons, and D. Baker. Clustering of low-energy conformations near the native structures of small proteins. *Biophysics*, 95:11158–11162, 1998.

[38] K. Simons, R. Bonneau, I. Ruczinski, and D. Baker. Ab initio protein structure prediction of CASP III targets using ROSETTA. *Proteins*, 37(3):171–176, 1999.

[39] G. Song and N. Amato. Using motion planning to study protein folding pathways. In *Proc. Res. Comp. Mol. Biol. (RECOMB)*, pages 287–296, 2001.

[40] E. Stollnitz, T. DeRose, and D. Salesin. Wavelets for computer graphics: A primer (part 1). *IEEE Comp. Graph. Appl.*, 15(3):76–84, 1995.

[41] S. Subbiah, D. Laurents, and M. Levitt. Structural similarity of DNA-binding domains of bacteriophage repressors and the globin core. *Curr. Biol.*, 3:141–148, 1993.

[42] W. Taylor and C. Orengo. Protein structure alignment. *J. Mol. Biol.*, 208:1–22, 1989.

[43] B. Zagrovic, E. Sorin, and V. Pande. β-hairpin folding simulations in atomistic detail using an implicit solvent model. *J. Mol. Biol.*, 313:151–169, 2001.

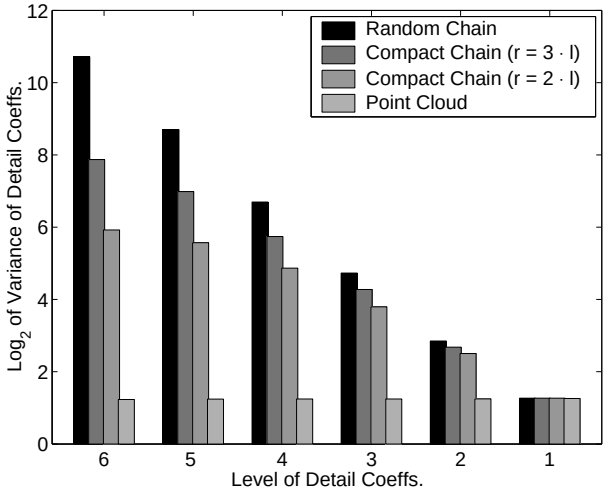


Figure 1: Average variance of the Haar detail coef. for random chains, compact random chains and random point clouds.

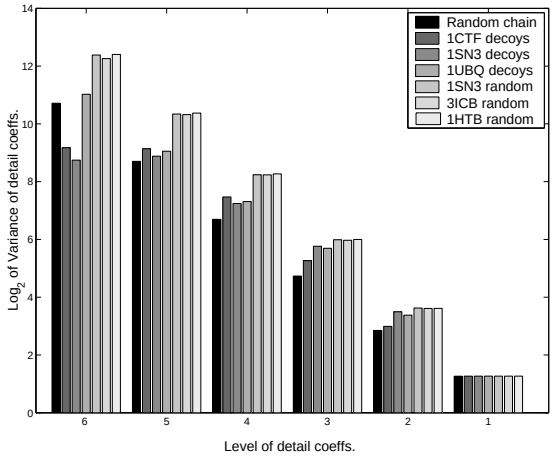


Figure 2: Average variance of the Haar detail coef. for random chains, decoy sets and randomly generated conformations.

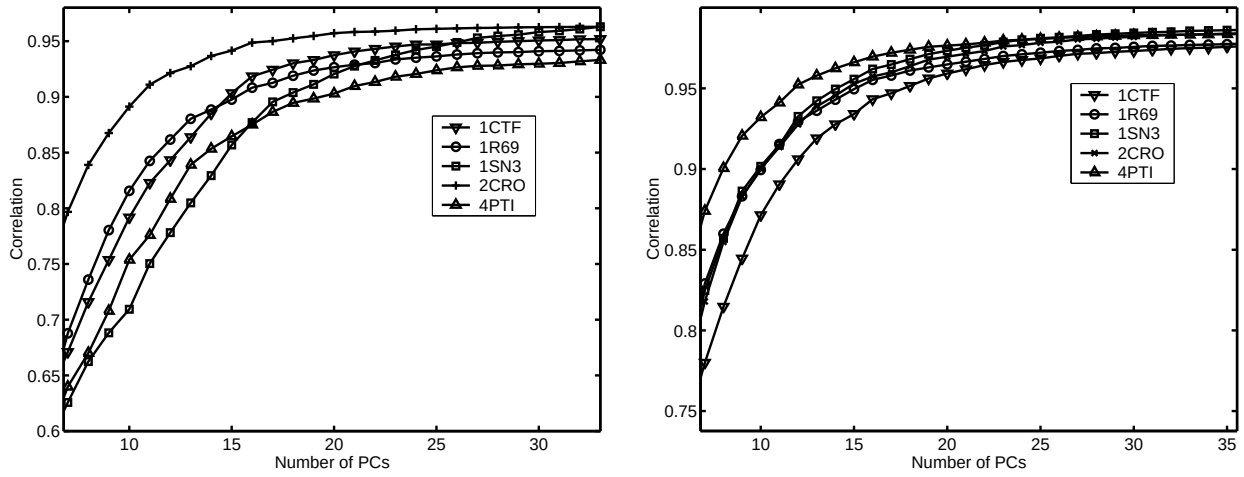
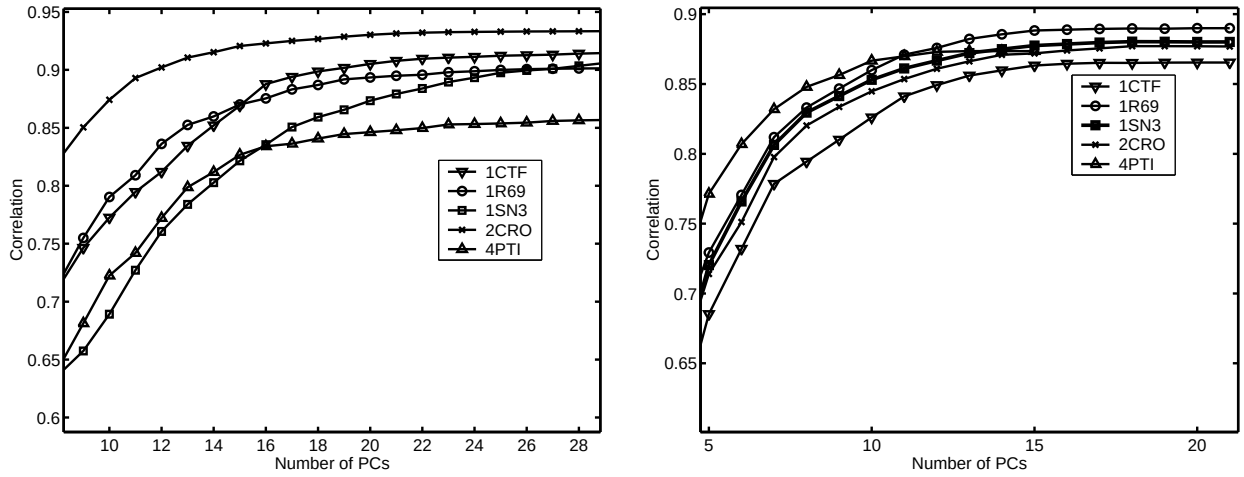


Figure 3: Correlation between *dRMS* and $\overline{d}_4^{PC} RMS$ for different number of PCs on (a) decoy sets and (b) random sets of some proteins.



(a) (b)

Figure 4: Correlation between *dRMS* and $\bar{d}_9^{PC}RMS$ for different number of PCs on (a) decoy sets and (b) random sets of some proteins.

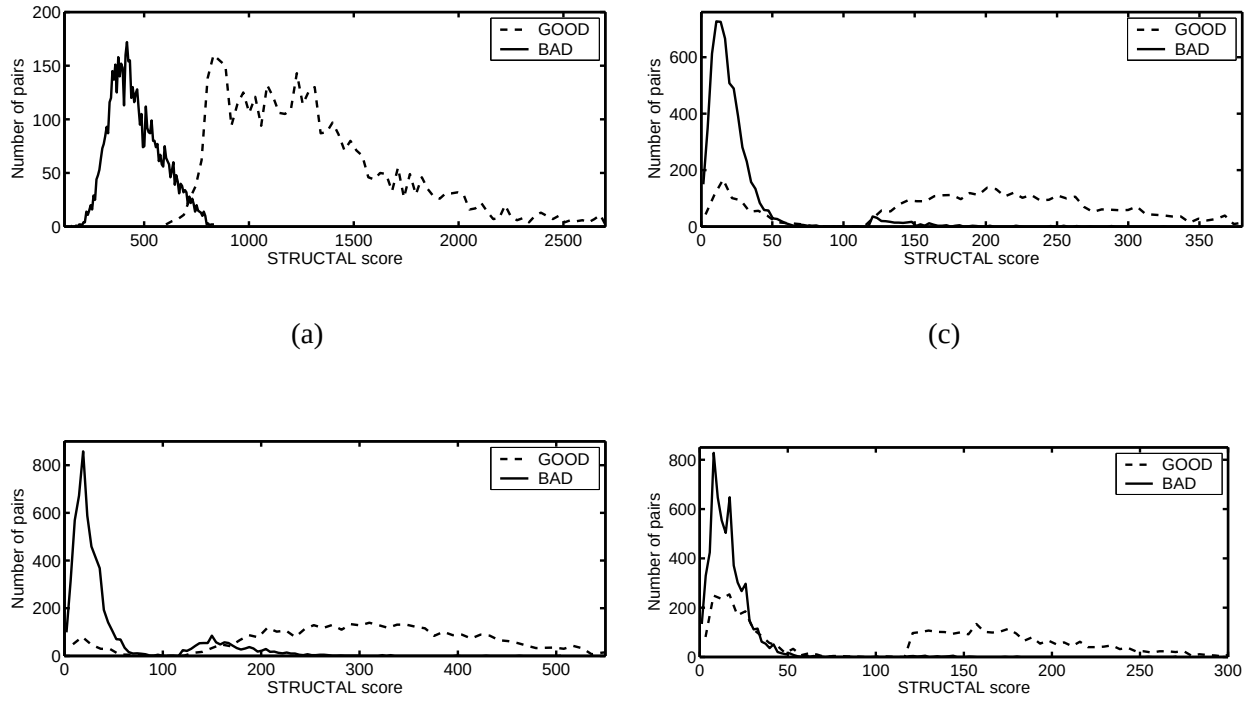


Figure 5: Histogram of scores obtained for both the good and the bad matches using STRUCTAL on (a) the full structures, (b) m-averaged structures with $m = 3$, (c) m-averaged structures with $m = 4$ and (d) m-averaged structures with $m = 5$.

	1CTF		1R69		1SN3		1UBQ		2CRO		3ICB		4PTI		4RXN	
m	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*
3	0.99	0.97	0.99	0.96	0.99	0.98	0.99	0.98	0.99	0.98	0.99	0.97	0.99	0.97	0.99	0.98
4	0.99	0.95	0.99	0.95	0.99	0.97	0.99	0.97	0.99	0.97	0.99	0.95	0.98	0.94	0.99	0.96
6	0.98	0.91	0.98	0.91	0.97	0.90	0.98	0.91	0.99	0.93	0.99	0.91	0.96	0.87	0.92	0.78
9	0.86	0.71	0.96	0.82	0.89	0.73	0.83	0.71	0.95	0.87	0.98	0.96	0.90	0.72	0.81	0.65
12	0.81	0.67	0.84	0.64	0.71	0.54	0.75	0.52	0.86	0.66	0.92	0.69	0.74	0.59	0.54	0.57
16	0.39	0.46	0.66	0.47	0.55	0.44	0.58	0.48	0.75	0.53	0.81	0.54	0.42	0.46	0.49	0.49

(a)

	1CTF		1R69		1SN3		1UBQ		2CRO		3ICB		4PTI		4RXN	
m	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*	*cRMS*	*dRMS*
3	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.98
6	0.99	0.96	0.98	0.96	0.98	0.95	0.99	0.95	0.98	0.95	0.99	0.95	0.98	0.96	0.99	0.97
9	0.94	0.87	0.95	0.89	0.95	0.88	0.95	0.85	0.95	0.87	0.96	0.86	0.92	0.88	0.93	0.90
12	0.91	0.81	0.84	0.76	0.84	0.75	0.90	0.73	0.84	0.74	0.90	0.73	0.86	0.81	0.87	0.84
16	0.70	0.62	0.69	0.64	0.68	0.61	0.82	0.61	0.71	0.62	0.83	0.64	0.73	0.70	0.73	0.74

(b)

Table 1: The correlation coefficient for different m values evaluated for *cRMS* and *dRMS* of (a) decoy sets and (b) randomly sampled conformations of various proteins.

	$k = 10$				$k = 25$				$k = 100$			
	err_1	err_2	*NN1*	$NN2$	err_1	err_2	*NN1*	*NN2*	err_1	err_2	*NN1*	*NN2*
1CTF	11%	3%	7.8	20	12%	3%	19	67	10%	2%	81	260
1SN3	14%	5%	7.5	23	15%	4%	18	69	12%	2%	76	301
1UBQ	15%	7%	7.0	38	14%	5%	18	102	13%	3%	77	332
2CRO	12%	4%	8.3	15	13%	3%	20	48	11%	2%	82	212
3ICB	13%	5%	7.8	23	13%	4%	19	68	12%	2%	78	292
4PTI	15%	6%	7.5	24	17%	5%	18	80	14%	3%	73	343
4RXN	13%	5%	7.9	19	16%	4%	19	62	16%	3%	78	295

Table 2: Mean values of err_1, err_2, *NN1* and *NN2* for 250 queries of k nearest neighbors.

	$k = 10$				$k = 25$				$k = 100$			
	err_1	err_2	*NN1*	*NN2*	err_1	err_2	*NN1*	*NN2*	err_1	err_2	*NN1*	*NN2*
1CTF	8%	3%	6.8	20	8%	2%	18	61	9%	1%	82	203
1SN3	8%	3%	6.8	23	9%	2%	18	62	9%	1%	82	197
1UBQ	8%	3%	6.9	38	9%	2%	18	60	9%	<1%	83	190
2CRO	6%	2%	7.8	15	6%	1%	21	41	5%	1%	91	134
3ICB	9%	3%	6.6	23	10%	2%	18	67	9%	1%	81	217
4PTI	8%	3%	6.9	24	9%	2%	19	58	8%	1%	83	194
4RXN	8%	3%	6.9	19	9%	2%	19	58	9%	1%	82	198

Table 3: Mean values of err_1, err_2, *NN1* and *NN2* for 250 queries of k nearest neighbors.

	$k = 10$				$k = 25$				$k = 100$			
	err_1	err_2	*NN1*	*NN2*	err_1	err_2	*NN1*	*NN2*	err_1	err_2	*NN1*	*NN2*
1CTF decoys	20%	9%	7.1	39	22%	8%	17	132	20%	5%	67	571
2CRO decoys	22%	10%	7.1	43	24%	9%	17	156	21%	6%	65	656
1CTF random	15%	7%	4.6	85	16%	6%	12	192	17%	4%	60	615

Table 4: Mean values of err_1, err_2, *NN1* and *NN2* for 250 queries of k nearest neighbors.

N	*cRMS*	$\overline{c}_4 RMS$	*dRMS*	$\overline{d}_4 RMS$
1,000	18.6s	12.4s	31.0s	2.2s
2,000	74.4s	50.0s	137.5s	8.0s
5,000	464.8s	312.0s	759.8s	43.4s
100,000	∼52h	∼35h	∼84h	∼4.8h

Table 5: Brute-force search using *cRMS* vs $\overline{c}_4 RMS$ and *dRMS* vs $\overline{d}_4 RMS$

k	Brute-force	kd-tree
1	30min	4min 10sec
100	41min	19min

Table 6: Brute-force vs kd-tree search for k nearest neighbors.

	$P \leq 0.005$			$P \leq 0.001$		
m	Total	FP	FN	Total	FP	FN
1	157 (1.6%)	48	109	130 (1.3%)	39	91
3	765 (7.6%)	331	434	646 (6.6%)	199	447
4	966 (9.6%)	205	761	782 (7.8%)	238	544
5	1572 (15.6%)	87	1485	1107 (11%)	136	971

Table 7: The total number of misclassified pairs, false positives (FP) and false negatives (FN).

.