Multiple Structural Alignment and Core Detection by Geometric Hashing

Nathaniel Leibowitz[1], Zipora Y. Fligelman[1], Ruth Nussinov[2,3]*; Haim J. Wolfson[1] †

[1]Dept. of Computer Science, School of Math. Sc., Tel Aviv University, Tel Aviv 69978, Israel,
Telefax : +972-3-640 6476, e-mail : wolfson@math.tau.ac.il;
[2]Sackler Inst. of Molecular Medicine , Sackler Faculty of Medicine , Tel Aviv University.
[3] IRSP - SAIC, Lab. of Experimental and Computational Biology, NCI - FCRDC,
Bldg 469, Rm 151, Frederick, MD 21702, USA

Abstract. A Multiple Structural Alignment algorithm is presented. The algorithm accepts an ensemble of protein structures and finds the largest substructure (core) of C_α atoms whose geometric configuration appear in all the molecules of the ensemble (core). Both the detection of this core and the resulting structural alignment are done simultaneously. Other large enough multi-structural superimpositions are detected as well. Our method is based on the Geometric Hashing paradigm and a superimposition clustering technique which represents superimpositions by sets of matching atoms. The algorithm proved to be efficient on real data in a series of experiments. The same method can be applied to any ensemble of molecules (not necessarily proteins) since our basic technique is sequence order independent.
Keywords : Multiple structural alignment; Geometric Hashing; invariants; structural core; transformation clustering.

Introduction

The rapidly increasing number of known protein structures, and the much faster increasing number of protein sequences whose structure is unknown, require the development of reliable and efficient techniques for protein sequence and structural comparison. The bioinformatics community has invested a sizeable effort in the development of pairwise and multiple sequence alignment algorithms (Doolitle 1996). However, since protein structure is significantly more conserved than its sequence, there is a clear need to develop efficient structural comparison techniques. These can be exploited both for the analysis of existing structures and for the development of techniques for novel protein structure elucidation using threading type techniques (Bowie, Luthy, & Eisenberg 1991). Robust multiple structural alignment algorithms are especially important to get insight into the structural core of a protein family. This in turn allows us deduction of the structurally conserved residues which are crucial for the function of the aligned protein ensemble. Multiple structural alignment has direct applications to computer assisted drug design. On one hand by a multiple structural alignment of target proteins, which interact with a given drug molecule, one can detect the functional site of these proteins. On the other hand by a multiple structural alignment of a family of drugs interacting with the same protein, one can detect the (structural) pharmacophore of these drugs. In the last two examples it is important to have an algorithm which can align geometrically congruent structures which are not necessarily represented by sequentially ordered amino acid chains. The algorithm that we present is not only sequence order independent, but essentially requires no order at all on the aligned atoms.

In the last decade several efficient pairwise protein structure comparison methods have been suggested. Many of them are influenced by the experience accumulated in the sequence alignment methods and apply sophisticated variations of the dynamic programming paradigm. The double dynamic programming method of Taylor and Orengo (Orengo & Taylor 1996) is a representative example. Other methods try to compare the pairwise distances within the structures, exploiting the fact that an inter-atomic distance does not change under rotation and translation. Such a method which is based on distance matrix exploration, named DALI, was suggested by Holm and Sander (Holm & Sander 1994). The above mentioned methods, especially the dynamic programming one, rely on the representation of a protein as a sequence of C_α (or C_β) backbone atoms. From a geometric perspective it views a protein as a curve in 3-D space. This reduces the matching problem to an essentially 1-D task, since curves are 1-D structures. Such methods have difficulty to tackle problems requiring alignment of 3-D structures which do not posses an inherent sequential order. Nussinov and Wolfson introduced the Geometric Hashing (Nussinov & Wolfson 1991) method to align structures in a sequence independent way.

There is a small number of methods which attempt to tackle the multiple structural alignment and core detection task. These methods can be roughly classified into two main categories. In the first one are algorithms

†Corresponding author - H.J. Wolfson, e-mail : wolfson@math.tau.ac.il.

which accept the multiple alignment from another procedure, be it sequence alignment, or secondary structure alignment, and concentrate on the detection and refinement of the structurally preserved **core**. In the second category are algorithms which tackle the structural alignment problem itself. These algorithms usually perform a series of pairwise structural alignments to deduce the multiple alignment.

A representative example of the first category is the structurally invariant core detection algorithm by Gerstein and Altmann (Gerstein & Altmann 1995). It accepts a multiply aligned set of structures and selects the position occurring in all the structures as an initial structural core. Then, an iterative procedure is applied which computes an average structure and removes the structurally most variable position from the core. This procedure is repeated iteratively until a certain cutoff is reached. The computation of an average structure of an ensemble is also done iteratively by performing in each iteration $O(N^2)$ (where N is the number of structures) pairwise best RMS fits (Arun, Huang, & Blostein 1987) between the ensemble structures. The method by Gelfand et al. (Gelfand *et al.* 1998) also accepts as its input an aligned set of structures. Then, for each pair of positions in the aligned ensemble, the average distance between these positions and the dispersion of this distance across the structures is computed. A "core" subset of positions is sought, where in each position the average dispersion computed relative to the other "core" positions is low. This class of algorithms circumvents the need to solve the difficult multiple structural alignment problem and focuses only on the refinement of a core for a given alignment.

Very few multiple structural alignment algorithms are discussed in the literature. In the SSAPm method by Orengo and Taylor (Orengo & Taylor 1996) all pairwise alignments of the structures are performed by the double dynamic programming SSAP method. The best fitting pair is chosen as a **seed** for the multiple alignment. An average consensus structure is computed and the information on the variance of each position is kept for subsequent stages. Then, iteratively, the best fitting structure is joined to the consensus with the consensus structure and positional variations being recalculated until all the structures are aligned. The position variance is used to extract weights both for the comparisons and for the assessment of positional conservation. In a quite analogous way Gerstein and Levitt (Gerstein & Levitt 1996) perform all pairwise structural alignments using their iterative dynamic programming structural alignment method. Then, they pick a 'median' structure which is on the average closest to all other structures in the least squares sense. All the other structures are aligned to that 'median' structure. As stated in (Gerstein & Levitt 1996) this does not automatically ensure geometric consistency at a given position across all the structures, and they suggest to double check such positions with the automatically generated pairwise alignments.

Our fully automated method solves the structural alignment and core detection tasks simultaneously. Relatively small structural fragments **appearing in all the structures** under consideration serve as initial seeds. These seeds, which are detected by geometric hashing of their invariants, induce pairwise transformations between the protein structures. In a subsequent step the initial seeds are merged into larger substructures so that all the induced pairwise rigid transformations are **simultaneously satisfied**. We have implemented the algorithm in C++ and started to experiment with it on ensembles of structures, which are known to be structurally related. The results obtained so far are very encouraging both in performance and run-time complexity. This project is in a preliminary stage and further improvements and experiments are being conducted.

The Multiple Structure Alignment Algorithm (MSTA)

In this section we outline our Multiple Structure Alignment **MSTA** algorithm. The input to this algorithm is an ensemble of N molecules, each being represented by the 3-D coordinates of its C_α atoms. The goal is to detect the largest geometric configuration of atoms which appears in all the molecules of the ensemble. We call this configuration the **geometric core** of the ensemble. By definition the **core** substructures belonging to the different molecules are all **congruent** up to a small error factor. Namely, for each pair of molecules M_I , M_J there is an alignment of the C_α atoms of both cores and a rigid transformation (3-D rotation and translation) T_{IJ} which superimposes these atoms with a small RMSD. Both the core and the induced alignments are solved simultaneously. Naturally, the structural alignment of the geometric cores induces structural alignments of the full molecules (see Fig. 2). A convenient way to represent this simultaneous N structure alignment is the following one. Let us pick one of the structures (e.g. the first) as a *reference structure* and compute all the $N-1$ rigid transformations between the remaining structures and the first one. The resulting $N-1$ dimensional vector of rigid transformations uniquely defines the multiple alignment, which superimposes the congruent cores. In the sequel we nickname the other $N-1$ structures as **source structures** and the resulting $N-1$ dimensional transformation vector as a **multi dimensional transformation**.

Note, that while we are aiming for the largest geometric core, a biologically interesting result might appear at a congruent substructure, which is not necessarily the largest one. The method we present allows detection of smaller substructures as well, as long as they pass the initial filtering stages of the algorithm.

The algorithm consists of three major stages:

1. Detection of seed matches and candidate multi-dimensional transformations.
2. Clustering of the multi-dimensional transformation

components and extension of the component seed matches.

3. Computation of the highest scoring multi-dimensional transformations.

Detection of seed matches and candidate multi-dimensional transformations

In this stage we detect **k-tuples** of atoms (points) whose geometric configuration appears in all of the N molecules . The points should not be collinear and the size of the k-tuple should be large enough to determine a rigid transformation between a pair of molecules ($k \geq 3$). The practical size of k is a compromise between the complexity of this first stage of the algorithm and the discriminatory power of a k-tuple structure. In the test cases presented in the "Experimental Results" section we used $k = 5$. Let us consider a substructure so that congruent copies of it appear in all the molecules. In order to detect such a substructure we first detect congruent copies of k-tuples appearing in all the molecules and then fuse them into larger multiply aligned substructures. A set of N congruent k-tuples, each belonging to a **different** molecule, induces a **multi dimensional transformation**.

Actually, we handle only k-tuples satisfying certain constraints which aim is to enhance the numerical stability of subsequent calculations, to reduce the run-time complexity, and to sift biologically irrelevant intermediate results. Specifically, for each atom we create its $k-tuples$ consisting of this atom and $k-1$ atoms residing in a spherical shell centered at that atom (the two radii defining the shell is a user dependent parameter). In addition the atoms should not belong to consecutive residues. The $k-tuple's$ atoms are ordered internally in both increasing and decreasing order of the sequence. In case we want to ignore this local sequence information, one can order the $k-tuple$ atoms according to geometric criteria.

In order to detect efficiently the congruent k-tuples, we employ the Geometric Hashing method (Lamdan & Wolfson 1988). Each $k-tuple$ is represented by a parameter vector which is invariant to 3-D rotation and translation and allows reconstruction of the k-tuple. Thus, congruent k-tuples are represented by identical parameter vectors. The k-tuples are inserted into a hash-table according to invariants so that congruent $k-tuples$ reside together.

In the case of an ordered 5-tuple, which can be considered as a closed polygon in space, it is fully determined by 9 parameters consisting of the ordered set of the lengths of the 4 edges, the 3 angles between them, and the two torsion angles between consecutive triangles. In practice, due to numerical stability, we use a somewhat different set of invariant parameters which consists of 9 inner distances out of the 10 available inner distances, classifying their possible symmetries. For each symmetry, we construct a 9-dimensional hash table, and perform the following steps:

```
For each of the source molecules
  For each k-tuple in a spherical shell
      neighborhood
    Compute the 9-inner-distances-invariant and
        find its symmetries.
    Insert the k-tuple to the appropriate
        hash table, using the invariant as a key.
  End-For
End-For

For each k-tuple in the reference molecule in a
    spherical shell neighborhood
  Compute the 9-inner-distances-invariant and
        find its symmetries.
  Query the appropriate hash table, using the
        invariant as a key.
  Store the reference k-tuple and the query
        results in a bucket.
End-For
```

Now, each of the buckets we have formed is associated with a reference k-tuple. In addition it contains all the source k-tuples congruent with it. We are interested only in those buckets which contain representatives from all the source molecules. Therefore, all the buckets in which at least one source molecule is not represented are ignored. This constitutes a major reduction in the complexity of the problem.

Finally, we compute the *multi-dimensional transformations* with the corresponding seed cores. Each remaining bucket determines at least one multi-dim. transformation. Let m_i, i=1,...,N-1 be the number of $k-tuples$ belonging to the i'th source molecule in the bucket under consideration. Then, $\prod_{i=1}^{N-1} m_i$ multi-dim. transformations are generated with regard to the reference $k-tuple$ which defines the bucket. Such a bucket is called a **combinatorial bucket** (see Fig. 1). Obviously, this discussion is meaningful only if all the m_i are strictly positive. Each multi-dim. transformation has a **seed (geometrical) core** associated with it. This core consists of the aligned $k-tuples$ which determined the multi-dim. transformation. In the following stages of the algorithm we will cluster these initial transformations and extend their associated seed cores.

Clustering of the multi-dimensional transformation components and extension of the component seed matches

In this stage we would like to extend the seed geometric cores by clustering the multi-dim. transformations. Since each component of a multi-dimensional transformation is by itself a rigid transformation between the reference molecule and a source molecule, it is defined by three rotational and three translational parameters. Thus, one could cluster these parameters. We have rejected such an approach for several reasons. First, it is not obvious what relative weight should be assigned to the rotation parameters versus the translation parameters. Another problem is the numerical stability. For

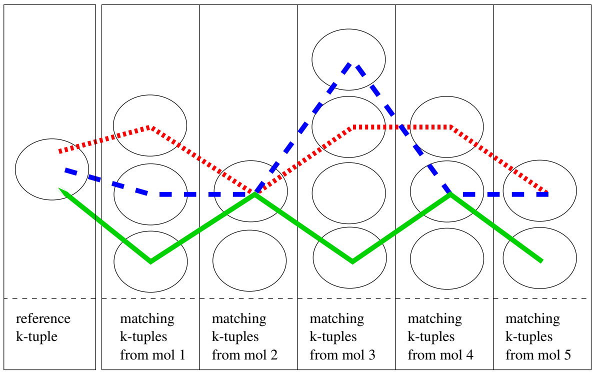

Figure 1: This figure illustrates how a combinatorial bucket defines core yielding multi dimensional transformations. The reference k-tuple that generates the bucket, is pictured at the first column. Congruent k-tuples from each source molecule, are arranged in separate columns. A path defines a combination of k-tuples from different molecules, that are congruent with the reference k-tuple.

example, a small change in the rotation angles induces larger dispositions of points which are farther from the origin. Also, the differences in rotation parameters can be somewhat compensated by counter changes in the translation parameters. Since we are looking for a core of matching atoms, we decided to adopt a clustering technique which measures the distance between transformations on the basis of the difference between the sets of points that they superimpose. This should ensure that transformations belonging to the same cluster, map relevant 3-D points to almost identical locations.

Let us consider separately a certain component in all the multi-dimensional transformations that we have accumulated so far. We obtain a set of rigid transformations between the reference molecule and the source molecule corresponding to that component. Our task is to cluster this set and compute a **prototype** transformation for each cluster.

Definition of a distance between a pair of transformations. First, we have to define a distance between a pair of transformations. We represent each transformation between the source and reference molecules by its **match list** which pairs the atoms in the congruent k-tuples. In the first iteration each transformation has a match list of size k, where k is the tuple size. The union of all the pairs in these match lists is created and indexed. Obviously, each transformation is consistent with at least k components of this list. We apply each transformation to the source molecule atoms of this entire union match list and check which atoms are mapped in a vicinity of their corresponding reference molecule atoms. Such matching pairs are **consistent** with our transformation. This way one can compute for each transformation the list of pairs it is **consistent** with. This is defined as **the transformation's mask**. The **distance** between two transformations is defined as the number of *consistent pairs* which are not shared by both lists. This is exactly the symmetrical difference of the masks we have defined. The distance definition complies with our intuitive notion that similar transformations should have similar match lists.

Clustering. Given a distance metric there are many standard clustering methods. We have used a technique which iteratively clusters proximate transformations and replaces these clusters by prototypes which are defined below.

Prototype generation. The next step is to compute a **representative prototype** for each cluster of transformations. This is again based on the match list information. For a given cluster the union of its match lists is created and a new rigid transformation, which provides the best superimposition of the match list pairs in the least squares (RMSD) sense (Arun, Huang, & Blostein 1987), is computed. We emphasize, that the prototypes emerge only from seed match lists that include representatives from all the different molecules. Therefore, this method of generating prototypes is well suited to the multiple aspect of the problem.

Iterative clustering of the representative prototypes. The transformation clustering procedure is applied iteratively until a stable situation is reached. The input for a new clustering iteration are the prototype transformations with their associated match lists generated by the previous iteration. The iterations are continued until the associated match lists do not grow any longer. The following pseudo-code summarizes the clustering stage :

```
For each source molecule
    Until steady state is reached
       Cluster the transformations
       For each cluster
          Compute its prototype
       End-For
    End-Until
End-For
```

Computation of the highest scoring multi-dimensional transformations

The input to this stage are sets of prototype transformations between all the source molecules and the reference molecule. A solution to the multiple alignment problem, consists of a combination of transformations from these sets. A-priori the combinatorial complexity of all these combinations is of the order of $\prod_{i=1}^{N-1} P_i$ where P_i is the size of the prototype set for the i'th source molecule. However, one should note that not all the prototype combinations produce multi-dimensional transformations. The search can be limited only to those prototypes which appear simultaneously in some combinatorial bucket (see the subsection on "Detection of Seed Matches"). Thus, we consider only transformations appearing in combinatorial buckets and replace them by their representative prototypes. Next, we remove those prototypes whose match list is below a required threshold parameter, and update the list of combinatorial buckets (removing those buckets that do not have prototypes from all the molecules). Finally, notice that the number of pre-clustered transformations of a given molecule is usually reduced to a smaller set of non-redundant post-clustered prototypes. All the above mentioned operations significantly reduce the number of multi-dimensional transformations to be explored compared to $\prod_{i=1}^{N-1} P_i$. An example of the size of multi-dim. transformations explored in our experiments appear in columns 5-7 of Table 1. The fifth column presents $\sum_{j \in Comb.\ Buckets} \prod_{i=1}^{N-1} m_i^j$, which is the number of all possible multi-dimensional transformations before the clustering stage. In the seventh column we present $\prod_{i=1}^{N-1} P_i$, which is the *a-priori* pairwise combinatorial solution space of the prototype transformations after clustering. The sixth column gives the number of multi-dimensional transformations defined by the combinatorial buckets based on the prototype transformations. Let $m{*}_i^j$ denote the number of non-redundant prototypes belonging to the i'th source molecule in the j'th combinatorial bucket. The complexity of the final solution space explored (see column six) is therefore $\sum_{j \in Remaining\ Comb.\ Buckets} \prod_{i=1}^{N-1} m{*}_i^j$.

Computing the core of candidate solutions. The **MSTA** algorithm concludes by inspecting all the combinations defined in the combinatorial buckets. For each combination we compute the intersection of the N-1 match lists, which are ordered according to the reference molecule atoms. If the intersection is larger than the minimal number of atoms we require in the geometric core, this combination is stored as a solution to the problem. Finally, we **rank the solutions by their core size**. To summarize :

```
Replace transformations in combinatorial
    buckets by their prototypes.
Reduce bucket complexity by removing
    irrelevant transformations.
For each combinatorial bucket
    For all the combinations it defines
        compute match list intersection
        If above minimum
           store as a solution
        end-If
    end-For
end-For
```

Experimental Results

We are in the process of conducting a large number of experiments to asses the performance of our algorithm. In these experiments we placed an emphasis on the evaluation of the quality of the results, the sensitivity of the algorithm to different inputs and parameters, and to the consumption of computer resources. Regarding the sensitivity to parameters, we should note that we have conducted several experiments alternating the, so called, reference molecule. This had no effect on the final results.

Here we present a subset of the experiments that we have conducted so far. Some of the multiple alignment results are shown in the (black and white) figures which have been created using the VMD (Humphrey, Dalke, & Schulten 1996) viewer. The original colored figures can be found on our WWW site *(http://silly6.math.tau.ac.il:8080/ISMB99/Results.html)* .

The data sets

The SCOP (Murzin *et al.* 1995) database presents the following hierarchical classification of the Protein Data Bank . **Classes** are defined at the upper level. Each class has several **folds**. Each fold has several **superfamilies**. Each superfamily has several **families** that are composed of the different *proteins* which are further classified according to their *species*. As you descend the SCOP tree the structural similarity among the proteins increases. We performed multiple structural alignments of protein ensembles belonging to various levels of the SCOP tree.

The first set of experiments was conducted on molecules belonging to the **serpin** fold. This fold has only one superfamily and one family, which consists of different proteins from different species. We have considered the following 13 molecules listed by their PDB code - 7apiA, 8apiA, 1hleA, 1ovaA, 2achA, 9apiA, 1psi, 1atu, 1ktc, 1athA, 1attA, 1antI, 2antI. These include antitrypsin alpha, elastase inhibitor, ovalbumin, antichymotrypsin alpha-I, antitrypsin, and antithrombin proteins from the human, horse, bovine, and hen species. The number of the C_α atoms ranges from 337 to 420. We have conducted 4 experiments by structurally aligning an increasing number of molecules. The results of these experiments, named serp 6, serp 9, serp 11 and serp 13 refer to the alignment of the first 6/9/11/13 molecules respectively. These results listed in Table 1 show (as expected) high similarity among the molecules. Obviously, the core size decreases with the increase in the number of molecules aligned.

Fischer et al. (Fischer *et al.* 1995) computed a structurally non-redundant dataset of the 1994 PDB release, by performing all against all pairwise structural alignments using the Geometric Hashing paradigm (Nussinov & Wolfson 1991) . Molecules with high structural similarity were grouped into one representative cluster. In addition clusters which are similar, although could not be grouped together have been outlined.

In the next example we chose proteins from the **calcium binding** cluster in (Fischer *et al.* 1995) . Although these proteins belong to the same fold and the same **EF hand like** superfamily, some of the molecules are from different families and proteins. The proteins are 4cpv from the parvalbumin family, 2scpA, 2sas, 1top, 1scmB from the calmodulin like family and 3icb

Experiment Name	Num of Molec	Avg Num of C_α	Num of k-tuples inserted	preCluster MultiD Trans.	postCluster MultiD Trans.	Prototype MultiD Trans	Num of C_α in top solutions(%)
serp 6	6	347	57183	1.23E+06	3.16E+05	5.77E+13	233(69.1%)
serp 9	9	356	94599	2.15E+08	2.46E+06	4.61E+20	180(53.4%)
serp 11	11	365	125862	7.71E+09	3.28E+07	3.44E+25	176(52.2%)
serp 13	13	372	154982	4.40E+11	2.40E+08	1.64E+30	163(48.4%)
serine prot	5	277	118023	6.26E+06	1.65E+06	9.68E+14	220(80.3%)
cal bind	6	140	12435	9.90E+04	1.01E+03	1.01E+07	31(41.3%)
globin 1.7	7	148	23331	2.73E+06	3.63E+05	1.52E+13	84(59.6%)
globin 1.9	9	147	29198	2.85E+08	1.27E+06	2.32E+15	73(51.8%)
globin cross	4	166	78693	4.02E+05	1.06E+05	7.52E+09	46(32.6%)
globin grst	7	146	159976	1.95E+05	7.64E+02	1.78E+07	71(52.2%)
Tim 2	3	388	115104	1.21E+05	5.93E+04	3.64E+07	191(53.5%)
Tim c3	3	341	64699	6.37E+04	3.06E+04	8.36E+05	98(40.0%)
Tim c5	5	383	135179	2.93E+06	6.26E+05	6.62E+13	66(27.0%)
Tim c7	7	391	55802	5.19E+11	1.74E+09	4.00E+18	40(16.2%)
tpi	8	249	31791	1.56E+12	4.00E+09	7.70E+19	216(87.8%)
hbundle 8	8	129	77498	4.78E+09	1.74E+08	4.85E+16	31(39.2%)
hbundle 9	9	127	82118	8.55E+11	1.19E+09	6.34E+17	31(39.2%)
hbundle 10	10	140	46039	1.68E+11	5.66E+07	6.00E+14	27(34.2%)

Table 1: **Summary of the experiments** - The data appearing in the columns is : 1) name of the experiment; 2) number of aligned molecules; 3) number of C_α atoms per molecule; 4) number of collected k-tuples; 5) complexity of the 'combinatorial buckets' (i.e number of multidimensional transformation that they generate) before the clustering stage; 6) complexity of these buckets after clustering; 7) number of all the possible multidimensional transformations that can be composed from the prototypes. The latter value is presented only to emphasize the reduction in the complexity we achieve by our algorithm (see the subsection on Computing the highest scoring multiD transformations). 8) the number of C_α atoms in the geometric core. The percentage in brackets, is computed **in comparison to the smallest participating molecule**, since the core cannot exceed the size of this molecule.

from the calbindin D9K family. Their sizes are from 75 to 185 (see Table 2 for full details). The results are summarized in Table 1 and displayed in Fig. 2.

Yet another set of tests was performed on a set of proteins from the **α-globin fold** : 1mbc, 1hlb, 2lh3, 1ecd, 2lhb, 3sdhA, 1thbA, 1mba, 1ith, 1cpcL, 1colA, whose size ranges from 136 for 1ecd to 197 for 1colA. The first 9 molecules are globins while the last two are a phycocyanin and a coilcin respectively. The latter belongs both to a different fold (toxins membrane translocation), and a different superfamily (coilcin). We carried out two types of multiple structural alignments. The first aligned only **globins** in increasing number of molecules (called globin 1.7 and 1.9 in Table 1) , while the other aligned molecules from **different families** (globin cross in Table 1). As expected the first type of tests yielded higher similarities than the latter.

We also tested the set of seven **globin** molecules that are mentioned in (Gerstein & Levitt 1996) the results can be found in Table 1 under *globin grst*.

We then proceeded to experiment on the **TIM-Barrel** fold. We conducted trials that included molecules from the same superfamily (e.g. Tim 2 in Table 1). We also aligned molecules that come from different superfamilies (e.g. Tim C3 - Tim C7 in Table 1). Examples of aligned core fragments appear in Fig.s 3 and 4. The structural classification details of these experiment can be found in Table 3. (Notice, that the first three examples in this table come from the same superfamily and the same family, however, these are different proteins belonging to different species. These are: 4enl-Enolase, 2mnr - Muconate lactonizing enzyme like, 1chrA - Mandelate racemase.)

Further experimentation on the TIM-Barrel fold was conducted on the **triose phosphate isomerase** family (tpi in Table 1). This example was chosen from the *HOMSTRAD* database (Mizuguchi *et al.* 1998). These molecules come from the same fold, superfamily, family and protein but from different species. We took the appropriate chains, and found the the mulitple structural alignment. Such an alignment can be used for modeling purposes.

The last set of experiments was performed on molecules from different folds of the same **all alpha** class. The proteins 1flx, 1aep, 1bgeB, 1le2, 1rcb, 256bA, 2ccyA, 2hmzA, 3inkC of sizes from 79 (1flx) to 159 (1bgeB) were considered. The 10 bundle experiment used different spherical shell parameters than the other two. Thus the reduction in the initial number of $k-tuples$ inserted, though the number of molecules increased.

pdb code	Num of C_α	Family	Protein
4cpv	108	parvalbumin	parvalbumi
2scpA	174	calmodulin like	sarcoplasmic calcium binding protein
2sas	185	calmodulin like	sarcoplasmic calcium binding protein
1top	162	calmodulin like	troponin
1scmB	138	calmodulin like	troponin
3icb	75	calbindin D9K	calbinding D9

Table 2: The Structural classification by SCOP of the calcium binding experiment.

The Parameters

The following are the main parameters used by the MSTA algorithm :

- minShell,maxShell - Defines the shell that is used when collecting k-tuples. For each atom, we form k-tuples from it and the atoms appearing in the 3-D spherical shell whose inner radius is minShell and outer radius is maxShell. The shell size influences the number and the type of congruent k-tuples that the algorithm begins the process with. We used two main ranges 6 - 10 angstroms and 9 - 12 angstroms, the first sensitive to local motifs and the latter sensitive to global motifs.
- maxRMS - This value is used for collecting fully preserved k-tuples to decide if a a source k-tuple and a reference k-tuple, are congruent. It therefore affects the number of k-tuples that pass on to the following stage, and the measure of their similarity. Our tests were performed with maxRMS range of 0.8-1.5. Lower RMS values are well suited for similar molecule.
- transformationClusterDist - Relates to the transformation distance (i.e. the difference between their masks) which we defined in our clustering algorithm. It specifies the allowed percentage of **consistent pairs** that are not shared by the two transformations. We used a value of 0.35.
- minCoreSize - specifies the minimum size of the geometrical core we are looking for. In protein alignment experiments this parameter is at least 20.
- vicDist - determines when two atoms (from different molecules) are in the same vicinity. If the distance between the atoms is below this value, they are defined as a matching pair. There is a tradeoff in the definition of this value. An increase of it, will produce larger cores, alas with larger RMSDs.

Performance

The experiments were performed on a PC with a 400 Mhz processor, 256$Mbyte$ RAM memory, under the Linux operating system. The code is written in C++. The running times ranged from seconds to hours. They depend on the number of molecules, their sizes, the similarity between them and the values of the above mentioned parameters. Most sessions terminated within minutes. Long running times (a few hours) were required for the TIM barrels. Since our algorithm is memory intensive, we believe that by increasing the RAM size even faster results could be achieved.

Conclusions and Future Work

The algorithm presented performed well on a large set of examples which included structures of different degrees of similarity. The fact that we start by detecting local structures (k-tuples) which appear simultaneously in all the molecules ensures performance which is superior to pairwise structural alignments, which may suffer from spurious matches induced by the density of the structures.

It should be noted that this is just a preliminary prototype of the method and we are working on improving all of the stages of our algorithm both in performance and computational complexity. We also plan a large set of additional experiments to evaluate its performance.

A major task that we intend to tackle is a multiple structural alignment algorithm which not only aligns the molecular structures of an ensemble and finds the geometric core which is shared by all the molecules in the ensemble, but also finds a subset of such an ensemble which gives a large multiple structural alignment. This problem is vaguely defined, since it is obvious that the smaller a subset, the easier it is to find a larger aligned substructure, yet some balance has to be achieved between the size of the structure ensemble and the size of the multiply matching substructure.

Acknowledgments

We thank Meir Fuchs for contributing software to this project. The research of H. J. Wolfson and R. Nussinov in Israel has been supported in part by grant number 95-00208 from BSF, Israel, by a grant from the *Israel Science Foundation* administered by the *Israel Academy of Sciences*, by the Israeli Ministry of Science grant, by the Tel Aviv University Basic Research fund. The research H.J.W. is partially supported by the Hermann Minkowski-Minerva Center for Geometry at Tel Aviv University. This project has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, under contract number NO1-CO-56000. The content of this publication does not necessarily reflect the view or policies of the Department of Health and Human Services, nor does mention of trade names, commercial products, or organization imply endorsement by the U.S. Government.

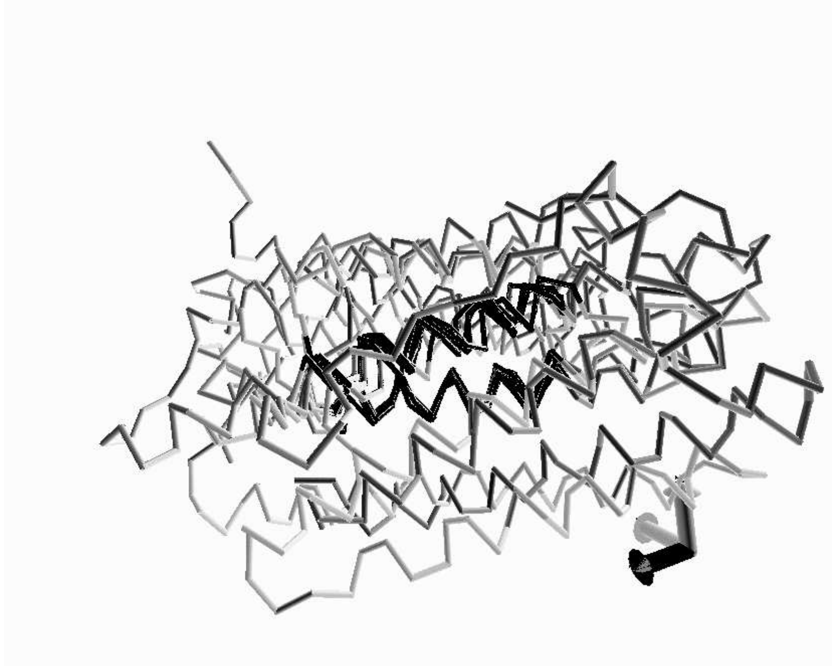

Figure 2: Alignment of 6 cal-binding molecules. Each molecule has a different color, however the common core is highlighted in black.

Figure 3: The core of the *Tim 2* experiment (3 molecules). Note the preservation of several β-sheets and their adjacent α- helices.

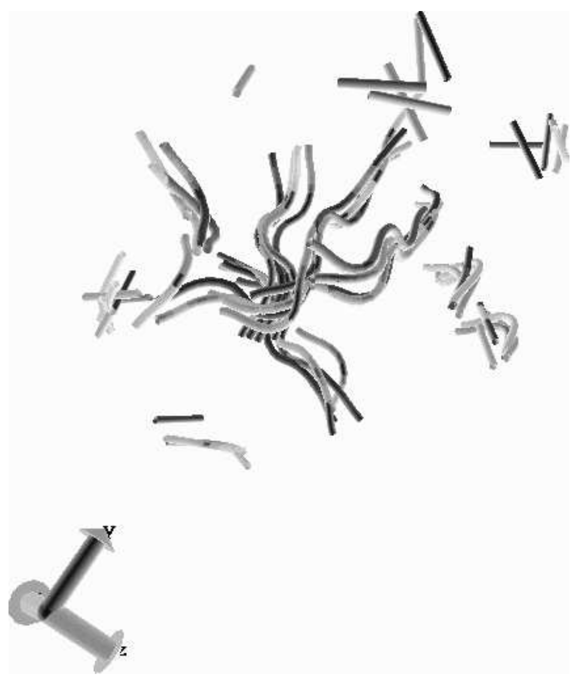

Figure 4: The core of the *Tim c5* experiment (5 molecules). You can notice that for proteins belonging to different superfamilies the β-sheet arrangement was preserved (as expected). Also, a part of a connecting α-helix can be observed (on the right hand side).

PDB code	Num of C_α	Superfamily	Family
4enl	436	enolase	muconate-lactonizing enzyme, C-term
2mnr	357	enolase	muconate-lactonizing enzyme, C-term
1chrA	370	enolase	muconate-lactonizing enzyme, C-term
7timA	247	triosephosphate isomerase	triosephosphate isomerase
1tml	286	cellulases	cellulases
1btc	491	glycosyltransferases	beta-Amylase
1pii	457	Tryptophan biosynthesis enzymes	Tryptophan biosynthesis enzymes
6xia	387	Xylose isomerase	Xylose isomerase
5rubA	436	RuBisCO, C-terminal domain	RuBisCO, large subunit, C-terminal domain
2taa	478	glycosyltransferases	alpha-Amylases N-terminal domain

Table 3: The TIM-Barrels' structural classification by SCOP.

References

Arun, K.; Huang, J.; and Blostein, S. 1987. Least Squares Fitting of Two 3-D Point Set. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 9.

Bowie, J. V.; Luthy, R.; and Eisenberg, D. 1991. Method to identify protein sequences that fold into a known three-dimensional structure. *Science* 253:164–170.

Doolitle, R., ed. 1996. *Computer Methods for Macromolecular Sequence Analysis*, volume 266 of *Methods in Enzymology*. San Diego: Academic Press.

Fischer, D.; Tsai, R.; Nussinov, R.; and Wolfson, H. 1995. A 3-D Sequence-Independent Representation of the Protein Databank. *Protein Engineering* 8(10):981–997.

Gelfand, I.; Kister, A.; Kulikowski, C.; and Stoyanov, O. 1998. Geometric Invariant Core for the V_L and V_H Domains of Immunoglobulin Molecules. *Protein Engineering* 11(10):1015–1025.

Gerstein, M., and Altmann, R. 1995. A Structurally Invariant Core for the Globins. *Computer Applications in the Biosciences (CABIOS)* 11:633–644.

Gerstein, M., and Levitt, M. 1996. Using Iterative Dynamic Programming to Obtain Accurate Pairwise and Multiple Alignments if Protein Structures. In *ISMB'96*, 59–67. AAAI Press.

Holm, L., and Sander, C. 1994. Searching protein structure databases has come of age. *PROTEINS: Structure, Function and Genetics* 19:165–173.

Humphrey, W.; Dalke, A.; and Schulten, K. 1996. VMD - Visual Molecular Dynamics. *J. Mol. Graph.* 14(1):33–38.

Lamdan, Y., and Wolfson, H. J. 1988. Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In *Proceedings of the IEEE Int. Conf. on Computer Vision*, 238–249.

Mizuguchi, K.; Deane, C.; Blundell, T.; and Overington, J. 1998. HOMSTRAD: a database of protein structure alignments for homologous families. *Protein Science* 7:2469–2471.

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

Nussinov, R., and Wolfson, H. 1991. Efficient detection of three-dimensional motifs in biological macromolecules by computer vision techniques. *Proc. Natl. Acad. Sci. USA* 88:10495–10499.

Orengo, C., and Taylor, W. 1996. SSAP: Sequential Structure Alignment Program for Protein Structure Comparison. In Doolitle, R., ed., *Methods in Enzymology, Vol. 266*. San Diego: Academic Press. 617–635.