

Protein-Protein Interaction Design with Geometric Hashing

Computational Geometry

Kyle Roberts, December 8, 2008

Protein-protein interactions (PPIs) are important for cell signaling, forming macromolecular scaffolds and complexes, gene expression, and many other biological functions. PPIs are often used as biological probes in experiments, such as with biosensors, and have become vital to understanding biological function. One potential method to study the function of PPIs is to design novel PPIs. One possible way to design novel PPIs is to replace a binding partner in a known complex with a protein with a similar binding interface and then utilize traditional design algorithms to optimize the new interface. Thus, an efficient geometric algorithm that can structurally align proteins can be used to locate potential proteins that could be used to design a novel PPI. This work will focus on one specific algorithm that can be used for structural alignment known as geometric hashing. This work will outline the geometric hashing algorithm and then survey its use throughout the biological areas of protein structural alignment, protein docking, and protein design. The ultimate goal of this work is to utilize the geometric hashing algorithm in order to design novel PPIs.

The Geometric Hashing Algorithm

The geometric hashing algorithm was first developed in the area of computer vision [1]. A key objective of computer vision is that of object recognition. In order for objects to be recognized they are stored in a database, and then searched for in a given scene. A good object recognition algorithm would be both efficient, not search for each database object sequentially, and be able to recognize objects that have undergone transformations or are partially covered. Geometric hashing was developed with these things in mind in order to allow a robot to detect a database of objects efficiently in spite of object occlusion or transformation.

Geometric hashing is an indexing method which allows for efficient searching of relevant information. In addition, the features that are indexed are local (so that occluded objects can still be recognized), and invariant to transformations (so that objects can be recognized if they have been transformed from the original database object orientation). Geometric hashing takes as input a database of objects and a scene in which to find the objects. The algorithm consists of two steps, a preprocessing step and a recognition step. During the preprocessing step the features

of the input object are extracted and hashed into a table. The recognition step similarly extracts features from the scene and then matches those features to the object hash table [1].

The first step of the preprocessing step is to extract the features of the objects that will be used for recognition. Features can be many things including points, linear and curvilinear segments, and corners. The set of features can be represented by a collection of dots, each dot containing the attributes related to the feature type. Once the features are extracted the algorithm must store them to be efficiently searched over. A hash table is used in order to store all the dots, but since the scene could contain a rotated or translated object, the dots are first scaled, rotated, and translated into a standard reference frame [2].

For example (example taken from [2]), consider Fig. 1 with an object consisting of 5 features located at dots 1 through 5. Consider using dots p_4, p_1 an ordered basis to the required reference frame. In order to construct the reference frame, the line segment $\overrightarrow{p_4 p_1}$ can be scaled to have a magnitude of 1 and then rotated and translated so that the median lies at the origin in the direction of the x-axis. This scaling, rotation, and translation to form the $\overrightarrow{p_4 p_1}$ basis can be applied to all the other points which are then stored in a quantized hash table. Since we are working under the assumption that object features can be occluded it is possible that dots p_4, p_1 could not be present in the scene. Thus, we must construct a basis for every pair of points and store the resulting transformed dots in the quantized hash table as well. Every point in the hash table will be associated with a given object and a given basis. Therefore, at the end of the preprocess phase we have a hash table that consists of entries related to each object for each pair wise basis generated for that model. [2]

The recognition phase, similar to the preprocess phase, begins by extracting the features from the input image. Once the features are extracted, two features are chosen as the candidate basis. The feature dots are transformed as in the preprocess step and the coordinates of the remaining feature points in the candidate basis are computed. The remaining features are then mapped to the hash table and all entries in the corresponding hash table bins receive a vote. Thus, it is expected that if we chose a basis in the image that corresponded to a basis for a given object, that object-basis entries in the hash table would get votes from all the non-occluded features from that object in the image. After the voting has occurred the algorithm can create a histogram of how many votes each object-basis pair received, and return the objects that received high amounts of votes as candidates for being present in the image [2].

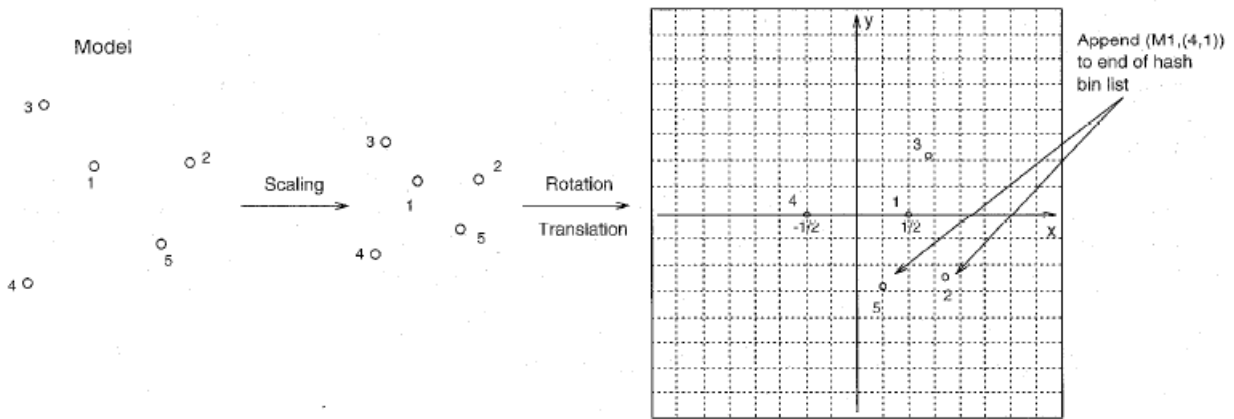


Figure 1. (Reproduced from [2]) Example of preprocessing steps of the geometric hashing algorithm for a given model using points 1 and 4 as the basis.

This finishes the description of the geometric hashing algorithm. In summary there are two steps that make up the algorithm. First, the features from the input objects are extracted and hashed into a table using all suitable tuples of features. Second, features are extracted from the image, and the features are hashed into the table and vote for all the entries in the bin of the hash table. It is expected that if a basis for a given object was chosen that it would get more votes than the other objects and could be reported by the algorithm. This recognition step can be done for all tuples of features in the image in order to locate all possible objects.

It is important that object recognition algorithms be able to handle many different types of transformations. Geometric hashing can handle many transformations and the only difference between the transformations is the number of features that must be used to form the basis. Both 2D and 3D data sets can be handled. In general the complexity of the preprocess phase will be $O(Mn^{c+1})$ where M is the number of objects in the database, n is the number of features the objects consist of, and c is the number of features needed to form a basis. The complexity of the recognition phase is $O(HS^{c+1})$ where S is the number of features in the input image, and H is related to the bin distribution of the hash table. If the hash table entries are relatively uniformly distributed then H will be close to a constant, but if the features all hash into a few bins H could be a function of the number of elements in the hash table which is $O(Mn^{c+1})$. One way around this is to use rehashing if the probability distribution of the index distribution is known. If this is

the case, the hash table entries can be rehashed such that the entries take on a uniform distribution in the hash table.[2]

One drawback of the geometric hashing algorithm as described is that it will only work for rigid objects that cannot undergo any changes in internal degrees of freedom. One idea to incorporate flexibility into the algorithm is to incorporate the additional degrees of freedom into the original basis transformations of the input objects. This is undesirable since this will most likely increase the complexity of the algorithm. Another approach is to locate each of the individual parts of an object separately, and then use some global consistency criteria in order to see if the individual parts fit together. This has the drawbacks that by the nature of flexible objects, it is often likely that the object might occlude itself and it will be difficult to locate every rigid piece of the object. [3]

Wolfson used a different approach to incorporate recognition of objects with rotary and prismatic joints without increasing the complexity or robustness of the algorithm [3]. The key observation of Wolfson's extension was to place the rotary joint position at the center of the reference frame. Therefore, when features from the image were extracted and used for voting, each feature could be used to both vote for an object and a joint location. Thus, to determine whether an object is present, the high scoring candidates could be checked that there are only two high scoring orientations at the given joint location corresponding to the two rigid parts of the rotary joint. Similar extensions were made for prismatic joints and objects with multiple joints. This extension avoids both the severe increase in complexity and susceptibility to occluded parts the other ideas for extension contained.

Application to Protein Structural Alignment and Structural Motif Discovery

Protein structural alignment and structural motif discovery is very important to understanding protein function. Proteins often consist of domains that have a given structure that confer function. Often the structure is relatively independent of sequence so truly geometric methods are needed to identify these similarities between proteins. The ability to quickly and efficiently align proteins and identify structural motifs in proteins will allow the classification and further the understanding of the protein function.

Nussinov and Wolfson were the first to apply geometric hashing to the area of protein alignment [4]. At the time of the introduction of this new technique to a new field, the current comparison methods mainly used the primary sequence of proteins for alignment. This is

undesirable because many structural motifs have little sequence similarity but are still related structurally and functionally. The current structural algorithms at this time were limited to looking for predefined motifs in the secondary structure which were related sequentially by the primary sequence. In addition, the algorithms compared every pair of atoms in any two structures, which is very time consuming. Introducing geometric hashing allowed for the true search of the 3D structures of proteins without needing any information about primary/secondary structure or predefined motifs. Also, the indexing method of computational geometry allows for more efficient comparison of structures than comparing atoms pair-wise.

The problem statement for the structural comparison of two proteins is very similar to that originally formulated for geometric hashing. The “objects” that were identified in the original algorithm are now protein molecules and the “features” can be a number of biologically relevant characteristics, but in the early applications were usually just C α atoms. Nussinov and Wolfson implemented a fairly straightforward adaptation of the geometric hashing algorithm to protein structural alignment [4]. By comparing only C α atoms they were able to reproduce the results of many alignment algorithms before them. Impressively, they did not require any information about the motifs they were searching for ahead of time, or utilize any primary sequence information, demonstrating the power of this method. The method was extended to use seeds generated from the geometric hashing algorithm in order to find larger structural motifs [5]. Nussinov and Wolfson demonstrate that geometric hashing allows for the identification of alignments that would be unable to be found using a sequence-based approach. The geometric hashing algorithm is robust to insertions, deletions, gaps and displacements of substructures, which would disrupt linear and sequence-based methods.

Since its introduction into the realm of structural alignment geometric hashing and variants have been applied to many biological problems. Wallace et al. develop a geometrical hashing algorithm that focuses on using side chain features to find active site motifs [6]. They manually created templates for various active site motifs, which could discriminate based on both residue type and coordinates, and then use their geometric hashing algorithm to identify new active sites in other PDB structures. This allows for the ability to keep an up to date database of important active site motifs.

The alignment method CTSS based on geometric hashing was also developed [7]. The key improvement that CTSS offers is the incorporation of local features as well as biologically

relevant features. In order to account for experimental error in PDB structures, CTSS approximates the protein backbone using an approximation spline which smoothes the data points. In addition, local curvature and torsion along the spline at each $C\alpha$ is used as a feature in the hashing algorithm. Also, the secondary structure of the protein is also utilized as a feature. The CTSS algorithm computes structural alignments efficiently and can also find new and longer structural alignments that other alignment methods overlooked.

Brakoulias and Jackson used a variant of geometric hashing to explore the different binding environments of phosphate binding sites [8]. The main advancement was that they used an all against all structural comparison that allowed for no previously defined motifs. Geometric hashing was utilized, but since an all against all method was used, no preprocessing step was needed and matching was done “on the fly.” A similarity matrix was generated for all pair-wise comparisons of phosphate binding sites. This matrix was then clustered in order to identify the binding motifs for phosphate binding proteins. The authors found the hashing algorithm to be two orders of magnitude faster than an alternative maximum clique detection algorithm, again demonstrating the benefits of geometric hashing.

Geometric hashing has also allowed for a database of protein-ligand binding sites to be created [9]. By utilizing hashing the database could be efficiently created and all ligand-binding pockets could be compared pair-wise to discover binding sites that are similar which can be used to aid in structure based design. The creators of the database made a key observation that the structural comparisons could be used in drug design. Often, if the 3D structure of a receptor is not known, but some ligands are, the structures of the ligands are used to model the receptor site. Similarly, the ligand binding sites of the known ligands found through the searchable database provide a clear model for how all of the known ligands could possibly bind in a similar fashion.

In an attempt to create a structure alignment algorithm that can be as ubiquitously used as the BLAST program is for sequence alignment, Milledge et al. implemented the SBLAST algorithm which applies a variant of geometric hashing in order to provide very fast query searches in a large database of protein structures [10]. The authors felt that the requirement that all possible pair-wise basis representations of models be contained in the hash table an unrealistic requirement for searching through large databases of protein structures. Thus, similar to the method of BLAST, SBLAST searches for hits of local structural similarities and the hits are extended outward to find larger structural alignments. The algorithm extracts triples of $C\alpha$ atoms

and uses the distances between the atoms as a 3-dimensional hash key. After preprocessing the triangles, triangles from the query protein are searched against the hash table to find initial hits. Once a hit is found it is extended to the longest common substructure within a given root mean square deviation using a depth first search on an adjacency list of triangles. Milledge et al. also demonstrate the power of geometric hashing to be parallelized. They are able to achieve a 70% speedup with 500 processors, which will probably increase when more proteins are incorporated into the search database.

Application to Protein Docking

The goal of protein docking is to find a ligand that binds into a given pocket of a protein. Geometric hashing has been utilized to locate pockets in proteins that ligands can bind. The geometric hashing algorithm can be straightforwardly translated to protein docking. The only change that needs to be made is the definition of the features that are being hashed. In docking, the surface of the ligand must fit into the surface of the protein. Thus, the ligand surface is used as the input object and the protein surface is used as the query image in which the ligand surface is searched for. An accurate surface representation of the protein and ligand is needed as the feature set of the hashing algorithm. For example, Fischer et al. rolled a ball over the van der Waals surface of a molecule to create a molecular surface composed of different types of faces [11]. The centroid of each face and its surface normal were the features that were used in their implementation. Fischer et al. report that geometric hashing was efficiently applied to protein docking and obtained very accurate recovery of native binding conformations while using time much less than previous algorithms.

Sandak et al. extended the docking algorithm to include flexibility by utilizing geometric hashing with rotary joints as discussed above. [12] The ligands being docked are allowed to hinge, which is represented as a rotary joint on a given atom. The algorithm including flexibility was tested on 4 structures and was successful in each case. Interestingly, in addition to finding the native crystal bound structure, other high confidence binding modes were found suggesting multiple binding modes for the given ligand. Unfortunately, this technique is limited to knowing where the hinge bend will occur before the actual docking run. This is remedied in [13], although geometric hashing is replaced with an atom distance-based superposition followed by finding shortest paths on a directed acyclic graph. In this case vertices can represent rigid fragments while edges correspond to flexible hinges in the protein.

It is important to note that although there are many docking algorithms available geometric hashing has definite advantages over other techniques. The first advantage is the ability to search over large sets of data and not require specific knowledge of where to start searching in the scene. In a review of many docking algorithms it is noted that only three of the reviewed methods, including geometric hashing, are able to handle the search over entire molecular docking surfaces with any prior knowledge of the binding site [14]. In addition, while geometric hashing runs in a matter of minutes, the other two algorithms run on the order of hours and days. Also, the output of the algorithms tested on biological molecules show that the three algorithms all perform very similarly in terms of quality.

Application to Protein Design

Protein design seeks to alter function, improve function, or introduce new function into proteins. Computational protein design seeks to do this by finding low energy conformations for a specified target protein structure. Recently, the first successful reports of introducing an enzymatic reaction, for which there is no natural enzyme, into a protein was published [15, 16]. This monumental breakthrough for the field utilized geometric hashing techniques in order to find the scaffolds for the new proteins.

In these computational designs it was necessary to find a suitable scaffold in which to design the active site. The active site positioning of amino acids was determined by motif analysis in conjunction with quantum mechanical methods. The next step was to search through given protein scaffolds in order to find structures that could accommodate the new binding site. Two different searches were employed which utilized geometric hashing [17]. The first approach involved an inverse-rotamer tree. In this technique it was assumed that the desired 3D arrangement of amino acids in the active site is known. Then using a rotamer library the active site side-chains are grown out such that a candidate ensemble of backbone structures is generated. These backbone structures are then searched against a protein database to find candidate structures. The described approach started from the side chains and built out an ensemble of backbones while the next technique goes from backbone to side-chain. For a given backbone scaffold, potential residue sites are chosen that are allowed to contain the active site. All rotamers in the rotamer library are placed at these positions in the protein. The resulting TS model placements are recorded into a hash table and by the nature of the hash function, if each bin of the hash table is full, a complete active site can be computed.

Protein-Protein Interface Design

Protein-protein interactions are vital to cells and organisms ranging in function from biological signals to immune defense. Recently, computational protein design algorithms have been applied to redesign PPIs [18]. There are many challenges relating specifically to designing PPIs. Notably, PPIs vary in both shape and chemical makeup. In addition, dynamics and variability at the interface of proteins makes designs related to protein-protein interactions difficult.

One major component of the field of PPI design is the design of protein-protein specificity interactions. Kortemme and Baker review four major cases in which protein-protein specificity design is successfully conducted[18]. One approach was able to successfully design a calmodulin-ligand peptide complex in order to increase the binding affinity of one ligand 86-fold higher than other ligands. Similarly, another group redesigned a PDZ domain in order to bind novel peptide sequences. Kortemme developed a scheme that involved making destabilizing mutations to one binding partner and making complementary stabilizing mutations on the other partner in order to confer specificity. These designs demonstrate the promise of computational protein-protein interaction design. Although none of the algorithms, and to my knowledge no algorithms that redesign protein-protein interactions computationally find a novel pair of proteins in order to redesign and create a novel binding pair; this is what my project aims to do.

My Project

While working in the Donald Lab I have extended the lab's design algorithms to handle protein-peptide and protein-protein interactions. My project goal is to develop a pipeline that will allow for the design of novel protein-protein interfaces. The main outline of the pipeline is as follows. The design process will start with a known protein complex. The interface structure of one of the binding partners will be used as a query structure to a geometric hashing algorithm which will find proteins with similar structural motifs within a protein database. These candidate proteins can be aligned with the original binding partner and protein-protein interface redesign using the K* algorithm[19] could be employed to find new candidate complexes.

I implemented a geometric hashing algorithm based on the techniques introduced by [4]. I extract each triple of C α atoms and rotate and translate the space such that all three points lie in the xz-plane and the first two points form a vector along the z-axis. Next, I grid the space into 1 \AA^3 cubes and hash the transformed atoms into the bins. During the recognition step, for each

query protein I randomly chose a basis triple a hash the atoms according to the transformation. Since motifs in proteins are structurally local, I'm exploring choosing my basis triple in a locally relevant way instead of just randomly.

In order to test the algorithm I tried to align two proteins which had been implicated by the lab as potential design targets, the zinc finger domain of pol eta and MDM2. Both consist of a similar alpha helix, antiparallel beta sheet structure. I used my hashing algorithm to align the structures (PDB codes 2I5O and 2GV2) and one of the top scoring alignments is shown in Fig. 2. The important alpha helix feature that both proteins share was aligned by the algorithm, but unfortunately the beta sheet motif could not be aligned. It is possible that this occurred because the structure in the 2GV2 is not actually a zinc finger domain, such that both structure features do not align well simultaneously. In addition, I aligned two zinc finger binding domains, 2I5O and 3ZNF, and although the geometric hashing algorithm could not align the whole structure, I was impressed how much better the geometric hashing alignment was to the built in pymol structural alignment algorithm.

Unfortunately, I have not set up the entire protein-protein interaction design pipeline. All the algorithms are implemented (within the same system), but the ability for the geometric hashing algorithm to search through a database of structures must be implemented and further tested to ensure desired results. I will continue to work to try to instantiate the design pipeline which will hopefully lead to the computational design of novel protein-protein interactions.

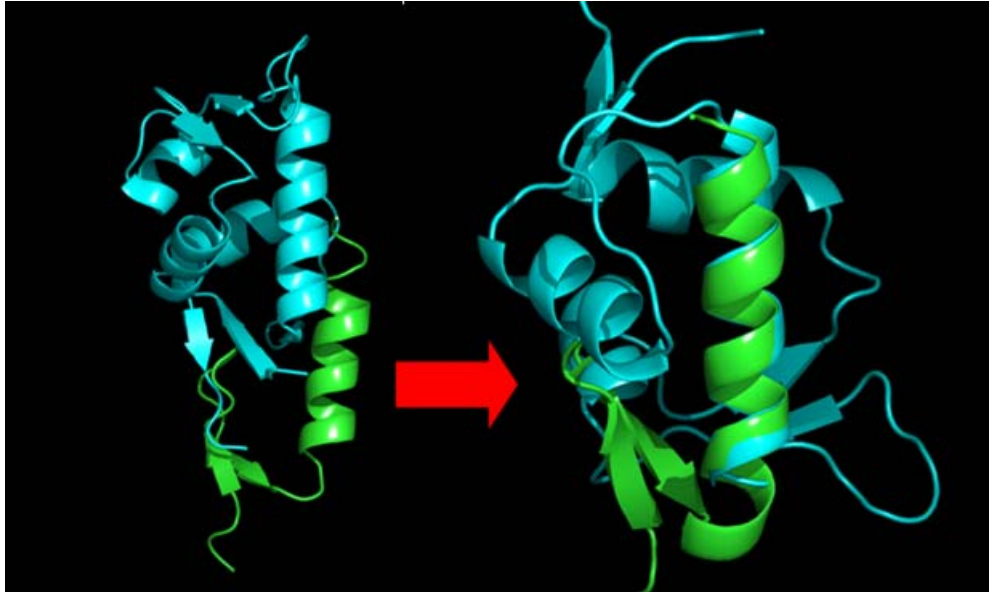


Figure 2. Alignment of 2I5O and 2GV2 using my geometrical hashing algorithm.

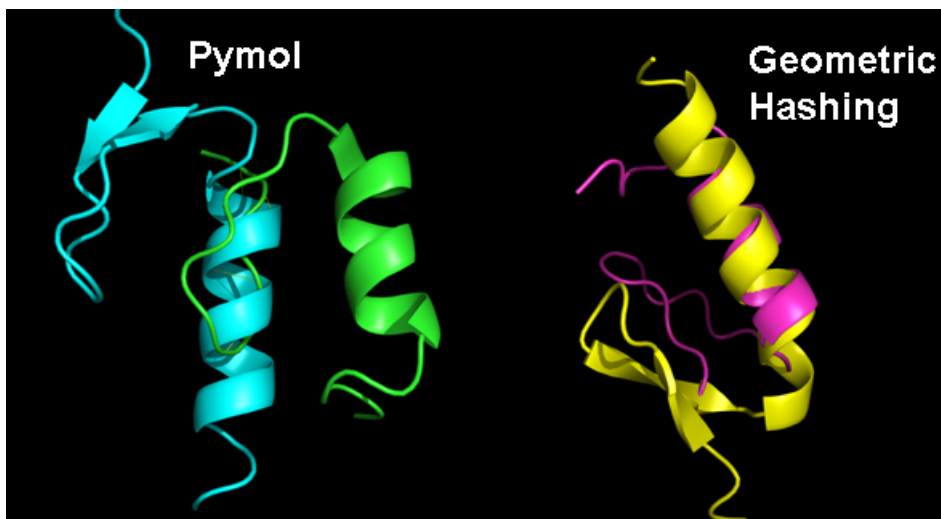


Figure 3. Alignment of two zinc finger binding domains, 2I5O and 3ZNF. Alignment of structures using the built in pymol function is shown on the left, and the geometric hashing result is shown on the right.

References

- [1] Y. Lamdan and H. J. Wolfson, "Geometric Hashing: A General And Efficient Model-based Recognition Scheme," in *Computer Vision., Second International Conference on*, 1988, pp. 238-249.
- [2] H. J. Wolfson and I. Rigoutsos, "Geometric Hashing: An Overview," *IEEE Comput. Sci. Eng.*, vol. 4, pp. 10-21, 1997.
- [3] H. J. Wolfson, "Generalizing the Generalized Hough Transform," *Pattern Recognition Letters*, vol. 12, pp. 565-573, Sep 1991.

- [4] R. Nussinov and H. J. Wolfson, "Efficient Detection of 3-Dimensional Structural Motifs in Biological Macromolecules by Computer Vision Techniques," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 88, pp. 10495-10499, Dec 1991.
- [5] O. Bachar, D. Fischer, R. Nussinov, and H. Wolfson, "A computer vision based technique for 3-D sequence-independent structural comparison of proteins," *Protein Eng.*, vol. 6, pp. 279-287, April 1, 1993 1993.
- [6] A. C. Wallace, N. Borkakoti, and J. M. Thornton, "TESS: a geometric hashing algorithm for deriving 3D coordinate templates for searching structural databases. Application to enzyme active sites," *Protein Sci*, vol. 6, pp. 2308-23, Nov 1997.
- [7] T. Can and Y. F. Wang, "CTSS: a robust and efficient method for protein structure alignment based on local geometrical and biological features," in *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, 2003, pp. 169-179.
- [8] A. Brakoulias and R. M. Jackson, "Towards a structural classification of phosphate binding sites in protein-nucleotide complexes: an automated all-against-all structural comparison using geometric matching," *Proteins*, vol. 56, pp. 250-60, Aug 1 2004.
- [9] N. D. Gold and R. M. Jackson, "A Searchable Database for Comparing Protein-Ligand Binding Sites for the Analysis of Structure-Function Relationships," *J. Chem. Inf. Model.*, vol. 46, pp. 736-742, 2006.
- [10] T. Milledge, G. Zheng, T. Mullins, and G. Narasinihan, "SBLAST: Structural basic local alignment searching tools using geometric hashing," in *7th IEEE International Conference on Bioinformatics and Bioengineering*, Boston, MA, 2007, pp. 1343-1347.
- [11] D. Fischer, S. L. Lin, H. L. Wolfson, and R. Nussinov, "A geometry-based suite of molecular docking processes," *Journal of Molecular Biology*, vol. 248, pp. 459-477, 1995.
- [12] B. Sandak, R. Nussinov, and H. J. Wolfson, "An automated computer vision and robotics-based technique for 3-D flexible biomolecular docking and matching," *Comput Appl Biosci*, vol. 11, pp. 87-99, Feb 1995.
- [13] M. Shatsky, R. Nussinov, and H. J. Wolfson, "Flexible protein alignment and hinge detection," *Proteins*, vol. 48, pp. 242-56, Aug 1 2002.
- [14] B. M. Inbal Halperin, Haim Wolfson, Ruth Nussinov,, "Principles of docking: An overview of search algorithms and a guide to scoring functions," *Proteins: Structure, Function, and Genetics*, vol. 47, pp. 409-443, 2002.
- [15] D. Rothlisberger, O. Khersonsky, A. M. Wollacott, L. Jiang, J. DeChancie, J. Betker, J. L. Gallaher, E. A. Althoff, A. Zanghellini, O. Dym, S. Albeck, K. N. Houk, D. S. Tawfik, and D. Baker, "Kemp elimination catalysts by computational enzyme design," *Nature*, vol. 453, pp. 190-195, 2008.
- [16] L. Jiang, E. A. Althoff, F. R. Clemente, L. Doyle, D. Rothlisberger, A. Zanghellini, J. L. Gallaher, J. L. Betker, F. Tanaka, C. F. Barbas, III, D. Hilvert, K. N. Houk, B. L. Stoddard, and D. Baker, "De Novo Computational Design of Retro-Aldol Enzymes," *Science*, vol. 319, pp. 1387-1391, March 7, 2008 2008.
- [17] A. Zanghellini, L. Jiang, A. M. Wollacott, G. Cheng, J. Meiler, E. A. Althoff, D. Rothlisberger, and D. Baker, "New algorithms and an in silico benchmark for computational enzyme design," *Protein Sci*, vol. 15, pp. 2785-2794, December 1, 2006 2006.

- [18] T. Kortemme and D. Baker, "Computational design of protein-protein interactions," *Current Opinion in Chemical Biology*, vol. 8, pp. 91-97, Feb 2004.
- [19] R. H. L. Ivelin Georgiev, Bruce R. Donald,, "The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles," *Journal of Computational Chemistry*, vol. 29, pp. 1527-1542, 2008.