Accepted for Publication in **Proteins: Structure, Function and Bioinformatics, 2007**

Context Shapes: Efficient Complementary Shape Matching for Protein-Protein Docking

Zujun Shentu[a], Mohammad Al Hasan[a], Christopher Bystroff[b], and Mohammed J. Zaki[a⋆]
[a]Department of Computer Science, [b]Department of Biology,
Rensselaer Polytechnic Institute, Troy, NY 12180, USA

⋆ Contact Author: Mohammed J. Zaki
CSCI, Lally 307
RPI, 110 8th St
Troy NY 12180
518-276-6340 (ph), 518-276-4033 (fax)
zaki@cs.rpi.edu

Abstract

We describe an efficient method for partial complementary shape matching for use in rigid protein-protein docking. The local shape features of a protein are represented using boolean data structures called *Context Shapes*. The relative orientations of the receptor and ligand surfaces are searched using pre-calculated lookup tables. Energetic quantities are derived from shape complementarity and buried surface area computations, using efficient boolean operations. Preliminary results indicate that our context shapes approach outperforms state-of-the-art geometric shape-based rigid-docking algorithms.

Availability: The context shapes software is available online at `http://www.cs.rpi.edu/~zaki/software/ContextShapes`.

1 Introduction

All biochemical processes involve some kind of molecular recognition, which can be defined as the formation of an energetically favorable docking between two molecules. Docking between proteins is involved in many cellular processes including cell signaling, regulation of enzyme function, intracellular trafficking, transcriptional regulation, the immune response, and many others.

A growing database of molecular interaction data [37] combined with large amounts of structural data for proteins [2], sets the stage for the development of computational methods that address systems biology questions. For example, we might want to predict protein-protein interactions without previous knowledge of the protein partners involved. Given proteomic data and a database of protein structures, we would like to be able to search this database using molecular surfaces to predict which of the proteins interact with each other and how they interact. Such large scale docking calculation required us to re-think the data structures used for docking calculations. To contemplate such large-scale docking queries, we must overcome several computational hurdles, including the problem of efficiently encoding and searching molecular surfaces. This paper addresses the surface encoding problem with an eye towards enabling large-scale computational docking experiments.

Interactions between proteins are governed by the kinetics and stability of the docked conformation, or *pose*. The kinetics of interactions are governed by the concentration of each molecular species in the cell, its sub-cellular location, and by its surface electrostatics. The stability of a pose is governed by the surface electrostatics and by the surface shape complementarity. Shapes that are more nearly complementary will fill space better when docked. The energetic stability of a pose can be approximated by the amount of surface area excluded from solvent, which is approximately the total area over which the two surfaces are complementary.

For this work we assume proteins are static shapes, i.e., rigid-bodies. We are aware that true protein-protein interactions generally involve conformational changes of side chains and backbone atoms. In some cases, flexible docking is modeled as an *induced fit* process, where the conformational changes occur as refinements after initial rigid body docking [5]. In other studies, docking is a *selected fit* process [34], in which flexibility is modeled as an ensemble of possible conformations of each protein, with each conformation being treated as a rigid body when docking [31]. Selected fit is our preferred approach, although we do not discuss flexible docking in this study.

Our new shape matching approach docks proteins in a rigid-body sense. Protein surface features are captured using the novel notion of *Context Shapes*, which are boolean data structures describing the local solvent excluded surface. It is assumed that docked proteins have one or more points of direct atom-atom contact and the goal of our method is to find one such pair of points. A pair of context shapes contains sufficient information to calculate the position and energy of a two-body binding interaction, both accurately and rapidly. We search for the relative orientations of the two surfaces using a pre-calculated lookup table. Furthermore, we derive energetic quantities from shape complementarity and buried surface area computations using efficient boolean operations.

For a comprehensive evaluation, our approach is first trained on a set of bound complexes, and it is then evaluated on an independent set of 84 bound and unbound receptor-ligand pairs taken from the docking benchmark v2.4 [23]. Experimental results indicate that the crystallographically determined pose is generally found within 2$\mathring{A}$ *Root Mean Squared Deviation* (RMSD) in a majority of the cases for the bound complexes. Our results are also better than or competitive with the best previous shape-based approaches on the unbound cases.

1.1 Related Work

The geometric surface of a protein has been the basis of docking through shape matching in several previous studies summarized briefly in the following paragraphs. The reviews by Via et al. [33], Halperin et al. [16], and Mendez et al. [21, 22] provide a more detailed analysis on various protein-protein docking methods.

The Geometric Hashing [24, 36] technique, originally developed in the computer vision domain, was used in a protein docking algorithm [11, 14]. A set of critical points are derived from the solvent excluded surface of a protein. Two critical points along with the mean of their surface normals define a local reference frame onto which the local critical points are projected. These transformation-invariant coordinates are extracted from the target proteins and saved in a hash table. In the recognition stage, a similar process is carried out on the query protein, and the hash table is used to find matching reference frames with maximally similar coordinates for their local critical points. This method is reported to be fast at the recognition stage but it requires a slow post-processing step in which the query protein is rotated into position to detect physical penetration.

Critical points on molecular surface were first introduced in [9] where combinatorial search on quartets of critical points were carried out to find candidate poses. Various pruning approaches based on solid angles, local volume and "unit vector defined by local centroid" (called the Solid Vector in our paper) were also proposed. The approaches proposed in both [9] and [11, 14] suffered from the huge combinatorial space of critical surface points, even though various criteria were used to prune the search space. Further, both methods had to perform actual transformation of one of the two given proteins to check shape penetration, which is costly. Local reference frames defined on combinations of critical points (quartets of critical points in [9] and pairs of critical points in [11, 14]) are very sensitive to the local shape and these methods might not work well on cases where the docking interfaces do not have a very high degree of shape complementarity. We use the critical points in another way without the need of local reference frames. Also, since we precompute the transformation matrices in a table, shape penetration detection and shape complementarity checking can be done at the same time. In [9] a local shape sampling method based on spherical harmonics was suggested (but not implemented). We use a similar, radial sampling method defined on a uniform point set on a unit sphere. Our representation based on bit-vectors is, however, entirely novel.

BiGGER [25] used two 3D bit matrices defined on a 3D grid, to represent a protein's surface and the 3D volume enclosed by the surface (called the "core"). BiGGER has two main steps. In the first step, the complete 6-dimensional binding spaces of both molecules is systematically searched to find a set of candidate good poses. In each rotational and translational step, binary operations such as OR and AND are applied to the bit matrices from the two molecules. Surface-surface contact is preferable while core-core overlap implies that the current pose can be discarded. In a second step, an interaction scoring function is used to rank the putative docked structures. We also used binary representation of context shapes in this work but in a very different manner. Context shapes are sampled by a radial template of a set of context rays. Each ray is represented by a binary string. Matching between context shapes is driven by a matching table; there is never a need to actually rotate or translate any one of the given proteins.

TreeDock [13] represents both the protein molecules in a complex as a union of atoms. Its docking process is guided by the user through a pair of anchor atoms, one from each of the protein molecules, which are in contact in the docking complex. Out of the remaining five degrees of freedom, four are searched exhaustively with a pre-specified resolution, and the fifth is analyzed by their merry-go-round algorithm, which computes the minimum energy configuration. The only scoring funciton is L-J potential that calculates the energy. Note that, TreeDock is fast only if the docking-site information is supplied by the user. The authors reported that without docking-site information, the docking process for proteins with about 500 heavy surface atoms, can take 3 to 5 days with 16 CPUs. Our approach takes on average 35 minutes of computing time on a single CPU, and does not require any docking site information.

A docking algorithm based on the fast Fourier transform (FFT) and Fourier correlation theory was first proposed in [17]. The algorithm begins by discretizing the surfaces of the two molecules using a 3D grid. A correlation function is defined to compute a match. In the process of matching, the receptor molecule is fixed and the ligand is rotated. For each rotation, each cell in the ligand is shifted to match with each cell in the receptor. This operation takes $O(n^6)$ steps in the translational/rotational space, where n is the size of each dimension of the grid. The FFT reduces the complexity to $O(n^3 \ln n^3)$. This method has been used in FTDock [15], ZDOCK(PSC) [7, 8], DOT [20], and GRAMM [32] to search the translational space. FFT is also used in HEX [27] to search the rotational space.

2 Methods

2.1 Context Shapes Overview

The *shape* of a protein is defined by its *Solvent Excluded Surface (SES)* [3, 10], which is the boundary of the solvent excluded molecular volume, and it can be calculated by rolling a spherical probe of the size of the solvent molecule over the exposed contact surface of each atom. The SES is composed of faces having three types of curvature: (1) *contact face*: the solvent accessible atomic surface, (2) *toroidal face*: the saddle-shaped surface where the probe makes contact with two atoms and (3) *re-entrant face*: the concave bowl-shaped surface where the probe makes contact with exactly three atoms.

If we extend SES by 2.8 Å (the diameter of a water molecule), we can get a new surface, called the *Solvent Included Surface (SIS)*. The region between SES and SIS, called the *SIS layer*, includes a single layer of solvent molecules, a hydration layer. The stability of a pose can be approximated by the amount of surface area excluded from solvent, called the *Buried Surface Area (BSA).* The BSA of a pose is the sum of the BSA of both proteins. Each BSA is the area of the local SES that intersects with the SIS of the other protein.

In principle, when two rigid shapes are positioned as close as possible without passing through each other, there are at least three points of actual contact, called *Physical Contact Points* (*PCP*). In practice, crystal structures of oligomeric proteins may have fewer than three PCPs, or they may cross each other slightly and thus have many overlapping points. For rigid docking these are most likely to be deviations due to errors in the structure, or errors in the way the surfaces were calculated. On the other hand, PCPs are more difficult to find when only the unbound forms of the proteins are known before docking, due to the large amount of conformational change that is possible when proteins dock. Nevertheless, for rigid docking, deviations from the PCP assumption are expected to be small, and even for flexible docking, PCPs provide a reasonable approximation. Thus the task of finding the most stable pose reduces to the task of finding PCPs.

We define the notion of *Context Shape* (CS) as the shape of the protein that is inside of a sphere centered at a point on the surface. Context shapes are sampled by radial *Context Rays* which emanate from the center of the sphere and are uniformly distributed on the sphere. Each context ray is composed of 32 bits, which are assigned a value of one or zero depending whether the location of the bit is inside or outside of the surface or a surface layer. Superimposing two CSs implies superimposing two surface points. We score superimposed CSs for complementarity in order to determine whether the pair of points might be a PCP. So the task of finding PCPs reduces to the problem of evaluating the shape complementarity of all CS pairs. The best shape complementarity is determined by trying all rotations of one CS versus the other. At each rotated position, the complementarity is evaluated using boolean operations on the aligned context rays. Looking only for PCPs obviates the need for a translational search.

Dense sampling of the SES was found to be essential to the success of finding the PCPs, but using too many points would slow the search. The SES can be efficiently represented by a sparse set of *critical surface points* without significant loss of accuracy [14]. Critical surface points are the face centers of the contact and re-entrant portions of the SES, ignoring the toroidal faces. The face centers are computed by projecting the centroid of each face to a point on the face. Only the sparse critical surface points are considered to be possible PCPs, which dramatically reduces the search space.

The BSA, defined above, is the final score of a given pose. But before the relatively expensive calculation of the BSA, simpler filters were applied: the *Solid Vector*, *Solid Angle* and *Overlap Volume* filter.

The calculations of both overlap volume and BSA require a one-to-one association of the context rays in the two CSs. Boolean operations are carried out on associated rays. Two rays from different context shapes are associated if, after rotating one CS relative to the other, the rays are nearest neighbors on the sphere. This is defined later in more detail. Since all CSs are based on the same template, a set of rotations and nearest neighbor ray associations were pre-calculated and stored as a *Matching Table*, greatly speeding up the rotational search.

The first pruning step was based on *Solid Angle*. The solid angle of the region of each context shape inside the protein was measured for different radii [9] (see Figure 3). For complementary context shapes the solid angles should roughly sum to 4π, or else the pose can be pruned.

Another pruning step used the notion of a *Solid Vector*, which is defined as a ray from the PCP to the center of mass of the corresponding context shape. The solid vectors of two CSs should be approximately 180 degrees apart in true poses. Poses in which the angle between solid vectors was too small were assumed to be impossible, as illustrated in Figure 3, since this would lead to too much overlap. To speed the search, only entries in the matching table that had highly obtuse solid vector angles were used. The search for a PCP then reduces to evaluating the overlap volume and BSA over rays associated by the selected entries in the matching table of pre-computed rotations.

Since docked proteins cannot pass through each other, we further pruned poses by considering the overlap volume, which is defined as the extent to which the two surfaces penetrate each other in a pose. To calculate the overlap volume, we created multiple surface layers projected inward and outward from the SES. Using multiple layers enabled us to permit shallow penetration of surfaces but disallow deep penetrations. Shallow overlaps in near-native poses can be the result of conformational changes in the proteins, but deep penetrations should never occur in near-native poses. Having multiple layers enabled the assignment of different weights to different depths of penetration.

For a given CS pair, all poses that were not pruned were scored using the BSA calculation. The rotation with the highest BSA score was stored, to be ranked later with all other CS pairs.

In short, the method has three main steps: 1) surface sampling and local shape representation as CSs, 2) pruning and complementary shape matching on CS pairs, and 3) ranking of poses based on scores. Details of these steps are described next.

2.2 Local Shape Representation

The local shape features of a protein were captured using context shapes. A context shape was represented using a sphere of a given radius r centered on a surface point (a potential PCP). The local shape features were captured by sampling the parts of the protein body bounded by the sphere. The sampling of shape features within a context shape consisted of a set of κ vectors, called context rays (CR), originating from the center of the sphere and ending at points evenly distributed on the surface of the sphere. The sphere was sampled at a density high enough that there was one end point per $\rho\mathring{A}^2$, on a sphere of radius $r\mathring{A}$. Thus we need approximately $\kappa = \lfloor \frac{4\pi r^2}{\rho} \rfloor$ rays to sample the local features; for example, $\kappa = 1256$ rays to sample at a density $\rho = 1$ for $r = 10\mathring{A}$.

Each context ray was divided into β segments, where each segment had one of two possible states: inside (1) or outside (0). We used $\beta = 32$ bit words to represent a context ray, a size which allowed for fast bit operations on 32-bit machines (64 bits can be used for a more dense sampling if required). The notation $CR[i]$ denotes the i-th bit of context ray CR.

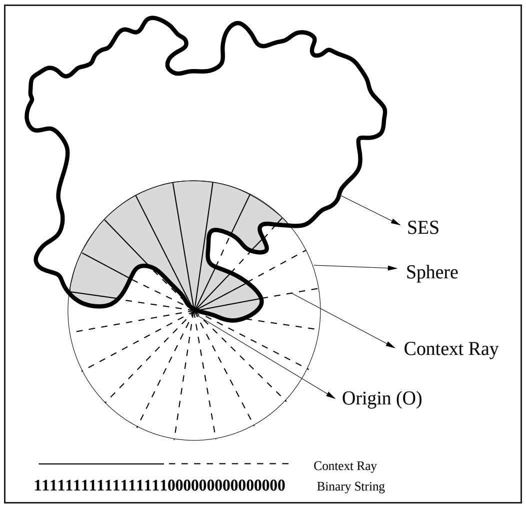
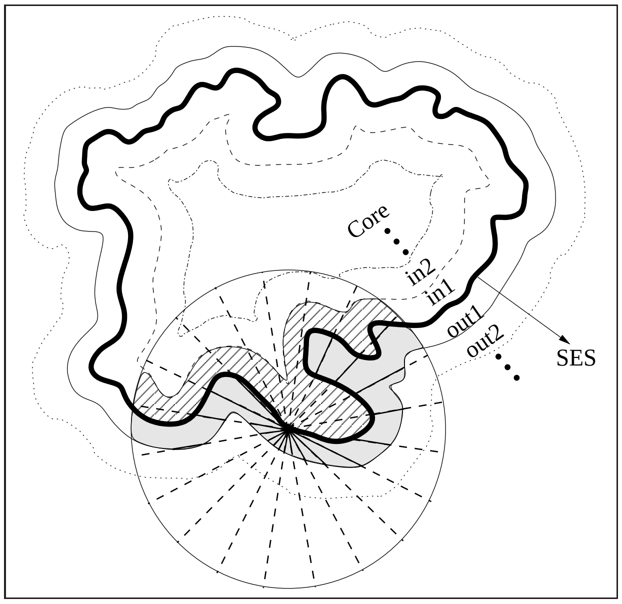


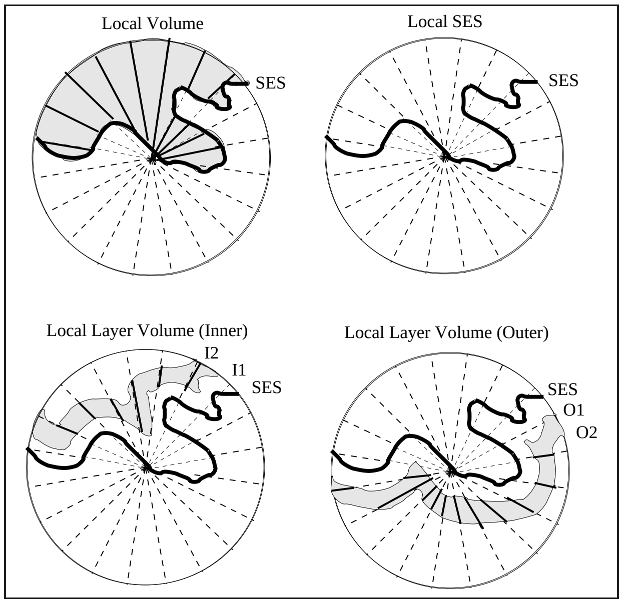
Figure 1: Context Shape Representing the Local Volume at a Critical Surface Point. The protein, SES, and the sphere centered at critical point O, are shown in 2D for simplicity. The shaded area gives the local volume of the protein at O. The context rays used to sample the context shape are also shown. Each segment of the ray has one of two possible states: inside "`1`" (shown in bold) or outside "`0`" (shown in dashes) of the local protein volume. The binary string composed of β bits for a given context ray is also shown.

Context shapes were used to represent different types of shape features such as surfaces and layers. The context shape representing a local solvent excluded volume is illustrated in Figure 1. Starting with the SES we computed several layers both inwards and outwards with each layer having a thickness of 1 Å. For any given distance $\delta \mathring{A}$, relative to the SES, we can compute the surface layer at that distance. Let $\mathcal{S}_\delta$ denote the surface at distance δ, where $\delta \in [-r, r]$, and where $\delta < 0$ indicates inner layers, $\delta > 0$ indicates outer layers, $\delta = 0$ indicates the SES, and where $\delta = -r$ and $\delta = r$ denote the sphere boundaries inside and outside the SES, respectively. See Figure 2 (a) for an illustration. A context shape in general is then the volume within the sphere bounded by two surface layers, and is given as $CS(\mathcal{S}_l, \mathcal{S}_u, r)$, where $\mathcal{S}_l$ and $\mathcal{S}_u$ denote the lower and upper surface boundaries, and r is the radius of the sphere. Since we used a fixed r, we denote a context shape as $CS(\mathcal{S}_l, \mathcal{S}_u)$. Figure 2(b) shows the different types of context shapes representing different shape features. For example $CS(\mathcal{S}_{-r}, \mathcal{S}_0)$ represents the solvent excluded volume. The context shape representing the SES is simply $CS(\mathcal{S}_0, \mathcal{S}_0)$. The figure also shows the context shapes for an inner layer volume $CS(\mathcal{S}_{-2}, \mathcal{S}_{-1})$ and for an outer layer volume $CS(\mathcal{S}_1, \mathcal{S}_2)$.

We refer to the different context shapes by simpler mnemonics: $CS_{vol} = CS(\mathcal{S}_{-r}, \mathcal{S}_0)$; $CS_{ses} =$



(a)



(b)

Figure 2: **(a)** Layers Inside and Outside the SES. Different layers are used to generate local shape features. Each layer is at a given distance from the SES (inwards or outwards). **(b)** Four main types of context shapes are shown (as the shaded region): i) local volume, ii) local SES, iii) local inner layer volume, and iv) local outer layer volume.

$CS(\mathcal{S}_0, \mathcal{S}_0)$; $CS_{inK} = CS(\mathcal{S}_{-K}, \mathcal{S}_{-K+1})$, for inner layers; and $CS_{outK} = CS(\mathcal{S}_{K-1}, \mathcal{S}_K)$, for outer layers (where K is in units of Angstroms Å). In this work we used four *in* and four *out* layers, namely: $in4$, $in3$, $in2$, $in1$, $out1$, $out2$, $out3$, and $out4$. See Figure 2(a) for an illustration of these layers. Finally, we refer to $CS(\mathcal{S}_{-r}, \mathcal{S}_{-4})$ as the CS_{core} region inside the SES, and we refer to $CS(\mathcal{S}_0, \mathcal{S}_{2.8})$ as the CS_{sis} region (for the solvent included surface), where $\delta = 2.8$ is the diameter of the water (solvent) molecule. Each context shape $CS(\mathcal{S}_l, \mathcal{S}_u)$ is given as a set of κ context rays $\{CR_i \mid i \in [1, \kappa]\}$, where each CR_i is a context ray, a binary string with a '1' for segments within the bounded region, and '0' outside the region. In the case of CS_{ses}, for a context ray, a single bit is set to '1' for each segment that intersects the SES and all other bits are set to '0'. Also note that a different number of rays may be used for different context shapes.

Generating Context Shapes

The different surface layers were computed using a 3D grid with a resolution of $0.2\mathring{A} \times 0.2\mathring{A} \times 0.2\mathring{A}$ (along the x, y, and z axes). Each cell has a type identifier $t \in \{ses, inner, outer, empty, core\}$, and is assigned cell coordinates $\langle c_x, c_y, c_z \rangle$, with $c_i \in \mathbb{N}$ (for $i \in \{x, y, z\}$). The coordinates are used to denote the distance (in terms of the number of cells) to the closest *ses* cell along the three axes. A pre-computed table is used to expedite the actual (Euclidean) distance calculation.

To identify the layers, in the first step, the cells that intersect with the SES were marked *ses*; the cells that were inside of the protein were marked *core* and the cells that were outside of the protein were marked *empty*. Then for each *ses* cell, we checked all of its local cells, within a distance of $4\mathring{A}$, to see if the current *ses* cell was their closest *ses* cell. If such a cell was currently marked as *core*, we changed its type to *inner* and updated its coordinates with respect to the current *ses* cell. Likewise, if a cell was marked as *empty*, we updated its coordinates to the *ses* cell and marked it as *outer*. Finally, if a cell was already marked *inner* or *outer* we only updated its coordinates if the current *ses* cell was closer. If any cell was updated, we recursively updated its local cells to see if the current *ses* cell was also closer to them. Once the local cells for all the *ses* cells were checked, we had two layers with a thickness of 4 Å from the SES, one inward and the other outward. A single pass was then made to generate all the context shapes $(CS_{vol}, CS_{ses}, CS_{inK}, CS_{outK}, CS_{core}, CS_{sis})$ based on the cell types and their Euclidean distance from the closest *ses* cell.

2.3 Complementary Shape Matching

A pose, π, between the context shapes CS^R_X from protein P_R (the receptor) and CS^L_Y from protein P_L (the ligand), with $X, Y \in \{vol, ses, sis, core, inK, outK\}$ and $K \in [1, 4]$, was represented by a one-to-one mapping of the context rays between the two context shapes. The feasibility of the pose was assessed using the overlap volume, defined as the volume that is labeled as inside in both CSs.

For a given pose π, the overlap volume of protein P_L's context shape CS^L_{vol} with respect to the context shape CS^R_X for layer X of protein P_R, is given as:

$$OV(CS^L_{vol}, CS^R_X, \pi) = \sum_{i=1}^{\kappa} V(CR^L_i \wedge CR^R_{i_\pi}) \tag{1}$$

where $CR^L_i \in CS^L_{vol}$, and $CR^R_{i_\pi} \in CS^R_X$ is a context ray mapped to CR^L_i according to pose π. Here $CR^L_i \wedge CR^R_{i_\pi}$ denotes the bitwise AND operation between the two context rays. The overlap volume within a single thin cone-shaped segment of the sphere is given as $V(CR^L_i \wedge CR^R_{i_\pi}) = \sum_{j=1}^{\beta} \nu(j)V[j]$, where $\nu(j) = (CR^L_i[j] \wedge CR^R_{i_\pi}[j])$ and $V[j]$ is the actual volume corresponding to the j-th segment of the context ray. Depending on the choice of the layer X above, we obtain different kinds of overlap volumes. The *total overlap volume* between two context shapes is a symmetric quantity, and is given as $OV(CS^L_{vol}, CS^R_{vol}, \pi)$. It represents the overlap of one protein's volume with the other, for a given pose. On the other hand the *layered overlap volume* is asymmetric, and is given as $OV(CS^L_{vol}, CS^R_X, \pi)$ and $OV(CS^R_{vol}, CS^L_X, \pi)$, where X is one of the inK or $outK$ layers. It represents the part of one protein's volume that overlaps a given layer X in the other protein, for a given pose.

2.3.1 Pruning Based on Solid Angle

The *solid angle* is defined as the area of the surface of a sphere that passes through the solvent excluded volume, divided by the radius squared. Figure 3 shows the solid angle $\theta^R(r)$ the receptor's context shape for a given radius value r. Given two Context Shapes, the sum of their solid angles $\theta^R(r) + \theta^L(r)$ at radius r Å$(0 < r \leq 10)$ should be less than but close to 4π, if they match very well [9]. Using a range of radii from $r = 5$ Å to 8 Å, with an increment of 1, we get an array of 4 solid angles for each Context Shape. We sum the solid angles and take the average over all the 4 radii, then set the lower bound to $0.75 \times 4\pi = 3\pi$ and the upper bound to $1.05 \times 4\pi = 4.2\pi$. The upper bound is a little bit bigger than the theoretical maximum value 4π to allow for flexibility.

2.3.2 Pruning Based on Overlap Volume

The optimal pose will have a low overlap volume. In the ideal case, the docked surfaces from the two proteins around the PCP should not have any overlap. However, since we use discrete steps to rotate the proteins to a candidate pose, we might miss the ideal pose. Data sampling also contributes some error, so a small amount of overlap must be allowed. Overlap volume is used as a filter to prune infeasible poses, especially overlap volume in deep layers. Poses that have a small but deep overlap volume (i.e., *sharp penetration*) are more likely to be false than poses that have shallow overlap volumes. In our method, any pose with a non-zero overlap in layers $in3$, $in4$ and $core$, more than $5\mathring{A}^3$ overlap in $in2$, or a total overlap volume of $80\mathring{A}^3$ or more was pruned.

The overlap filters can be formally stated as follows:

- *Prune Large Overlap*: if $OV(CS^L_{vol}, CS^R_{vol}, \pi) > 80\mathring{A}^3$, then reject pose π.
- *Prune Sharp Penetration*: For $A, B \in \{L, R\}$, reject pose π if either:
  (a) $OV(CS^A_{vol}, CS^B_X, \pi) > 0$ for $X \in \{in3, in4, core\}$, or
  (b) $OV(CS^A_{vol}, CS^B_{in2}, \pi) > 5\mathring{A}^3$

Note that L and R represent the ligand and receptor proteins, and the expression $A, B \in \{L, R\}$, means that the calculations were done twice, switching L and R superscripts.

2.3.3 Enumerating Rotations

The comparison of two context shapes, CS^L from the ligand protein and CS^R from the receptor protein, begins by superimposing their centers. Then there are three rotational degrees of freedom that must be sampled in order to find all possible poses. We can sample the rotational space by first rotating the z-axis to evenly distributed points on the surface of the sphere (e.g., we can use the context rays, CR_l, of CS^L and align z to each ray, CR_i, of CS^R), and then rotating in steps of γ° around each rotated z-axis, yielding $\lfloor \frac{360^\circ}{\gamma^\circ} \rfloor$ poses (e.g., for $\gamma = 5.8^\circ$, we get 62 poses). The first step assures that the z-axis is positioned uniformly on the sphere, and the second step assures that the x and y-axes positions are uniformly sampled. Using this approach, we generated a total of $1256 \times 62 = 77872$ rotations.

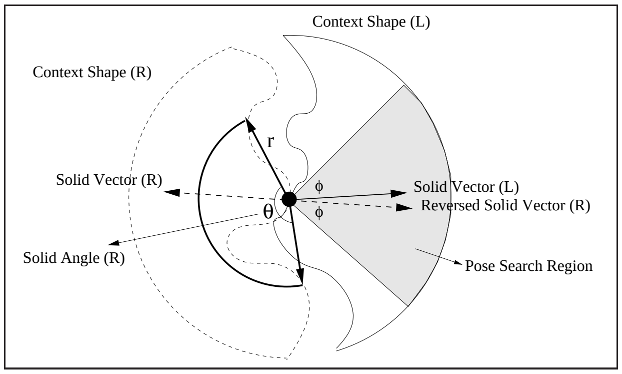
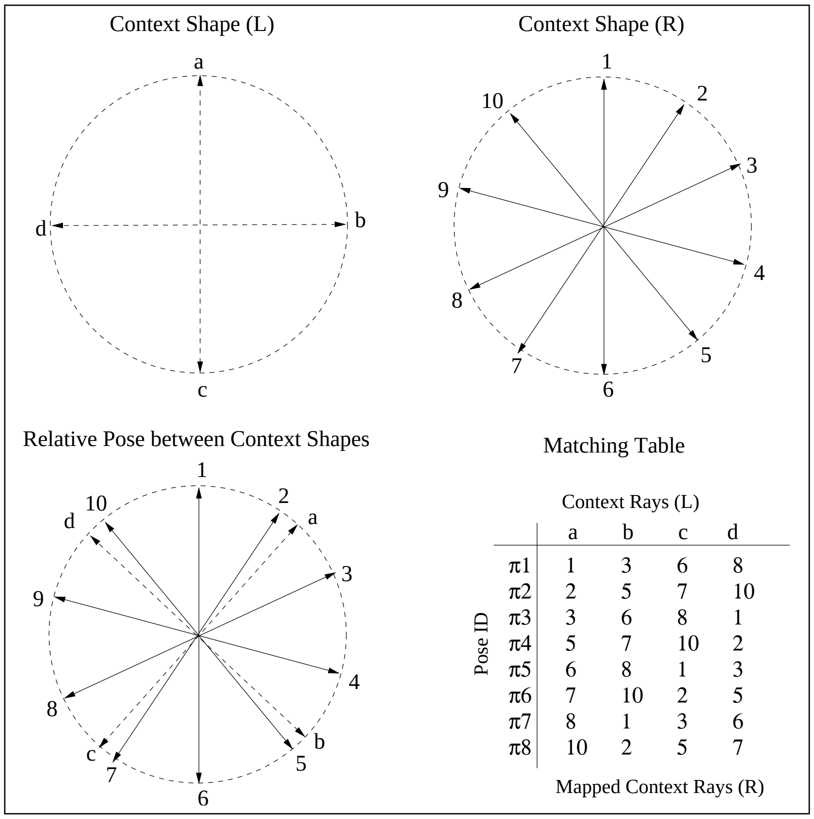


Figure 3: **Solid Angle Pruning**: For a given pose π if the sum of the solid angles $\theta^R(r)$ and $\theta^L(r)$ is not within $[3\pi, 4.2\pi]$, we reject the pose. **Solid Vector Pruning**: For a given pose π if the reversed solid vector of CS^R_{vol} is not within an angle of ϕ around the solid vector of CS^L_{vol}, we reject the pose.

To prune obviously impossible poses, we computed the center of mass of each context shape CS_{vol}. For two CS_{vol}s, the vectors to the center of mass, called the *Solid Vectors*, should be almost in the opposite directions when correctly docked. We can safely restrict the search for good poses to those where the reversed solid vector of CS^R_{vol} lies within an angle of ϕ° (we used $\phi = 15^\circ$) around the solid vector of CS^L_{vol}, as illustrated in Figure 3.

For each of the rotated positions, a one-to-one association of context rays between two context shapes was pre-computed and stored in a *Matching Table*, since table checking is much faster in general than rotation operations. Note that for a given pose, since one set of uniformly sampled rays (say from the ligand) will not, in general, exactly match the other set of rays (say, from the receptor), we sample the receptor ray set at a much finer detail, so that the approximate ray correspondences would have negligible impact on the accuracy of the pose. Thus, in order to get a one-to-one association, one of the CSs, CS^R, was sampled at a higher density, between 2κ and 3κ rays, versus κ context rays for CS^L. Context rays in CS^L were then associated one-to-one with a subset of the context rays in CS^R by rotating and then finding the nearest neighbor.

Consider the example shown in Figure 4. Context shape CS^L has $\kappa = 4$ context rays and CS^R has $2.5\kappa = 10$ rays. Both sets of rays are evenly distributed inside the sphere (shown here in 2D for simplicity), and are labeled consecutively in the clockwise direction. Assume we sample the poses in steps of 45° (clockwise) starting from the alignment of ray $a \in CS^L$ with ray $1 \in CS^R$. Then there are eight possible poses, since $\frac{360^\circ}{45^\circ} = 8$. For each of these poses, we would have to rotate the rays of CS^L and find the closest rays of CS^R. An example pose (after rotating CS^L by 45°) is illustrated in the figure, given as $\pi_2 = \{a \to 2, b \to 5, c \to 7, d \to 10\}$.



Pose ID	Context Rays (L) a	b	c	d
π1	1	3	6	8
π2	2	5	7	10
π3	3	6	8	1
π4	5	7	10	2
π5	6	8	1	3
π6	7	10	2	5
π7	8	1	3	6
π8	10	2	5	7

Mapped Context Rays (R)

Figure 4: Matching Table. CS^L has 4 context rays, whereas CS^R has 10. The context shapes are shown in 2D. There are 8 candidate poses using a step size of 45°. Once such pose corresponding to the second row $\pi 2$ in the matching table is shown on the left, i.e., $\pi_2 = \{a \rightarrow 2, b \rightarrow 5, c \rightarrow 7, d \rightarrow 10\}$.

2.4 Ranking Matches

To find the best docking orientation of two proteins, we estimated the amount of surface excluded from the solvent, called the *Buried Surface Area (BSA)*. The BSA approximates the desolvation energy, which is the primary driving force of protein-protein interactions. The more surface excluded from the solvent, the better the binding energy. The BSA of one protein is the amount of its SES area that overlaps in the SIS layer of the other protein, as shown in Figure 5. The sum of the buried surface area from both context shapes is used to score the quality of the pose. In practice, to account for the approximate nature of PCPs, we used the weighted sum of the buried area in several external layers.

For a given pose π, the buried surface area of protein P_L's context shape CS^L_{ses} with respect to the context shape CS^R_X for layer X of protein P_R, is given as:

$$BSA(CS^L_{ses}, CS^R_X, \pi) = \sum_{i=1}^{\kappa} A(CR^L_i \wedge CR^R_{i_\pi}) \tag{2}$$

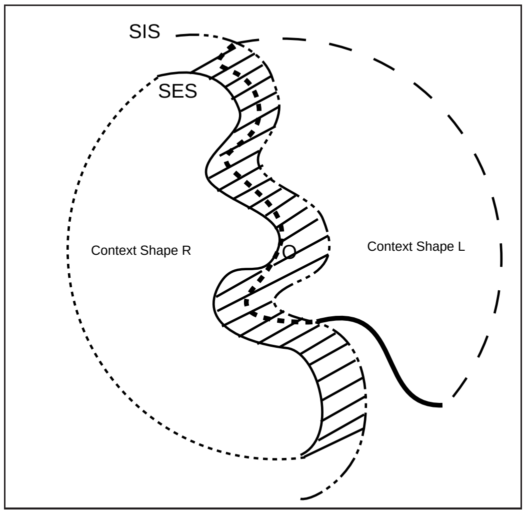


Figure 5: Buried Surface Area. CS^R's SIS layer can include a single layer of water molecules. If any part of CS^L's SES falls in the region, we can safely claim that to be buried. The buried part of CS^R's SES can be calculated in the same way by using CS^L's SIS layer.

where $CR_i^L \in CS_{ses}^L$, and $CR_{i_\pi}^R \in CS_X^R$ is a context ray mapped to CR_i^L according to pose π. The buried area is given as $A(CR_i^L \wedge CR_{i_\pi}^R) = \sum_{j=1}^{\beta} \alpha(j)A[j]$, where $\alpha(j) = (CR_i^L[j] \wedge CR_{i_\pi}^R[j])$ and $A[j]$ is the actual area corresponding to the surface point represented by bit j. Note that the total buried surface area for layer X is the sum $BSA(CS^L, CS^R, X, \pi) = BSA(CS_{ses}^L, CS_X^R, \pi) + BSA(CS_{ses}^R, CS_X^L, \pi)$.

The scoring function used to rank different poses is the weighted sum of the buried area across several layers, given as follows:

$$Score(CS^L, CS^R, \pi) = w_1 \times BSA(CS^L, CS^R, in1, \pi) + \sum_{K=1}^{3} w_K \times BSA(CS^L, CS^R, outK, \pi) \quad (3)$$

The weights for each layer were chosen empirically to optimize the rankings; we used $w_1 = 4$, $w_2 = 1$, and $w_3 = 0.25$, indicating the relative importance of the buried area in each of the inner/outer layers. Finally, for the two context shapes CS^L and CS^R we find the best pose π, which has the highest score, given as: $Match(CS^L, CS^R) = \max_\pi Score(CS^L, CS^R, \pi))$. To match two proteins, we check all the possible context shape pairs and rank them in decreasing order of their scores.

2.5 Context Shapes: The Complete Method

The complete context shapes method is composed of two steps: off-line preprocessing and online shape matching.

The off-line preprocessing step is used to generate the context shapes and the matching table.

Given a protein complex along with information about the chains to use as the ligand and receptor, we use the MSMS algorithm [29] to generate the SES. The SES is given as a triangular mesh consisting of a set of surface vertices and a set of surface triangles, also called the *faces*. Next we generate a 3D grid with cell size $(0.2\mathring{A})^3$ to compute the surface layers, the context shapes, and the context rays, as described previously. The matching table, a common device in matching of any pair of context shapes, is also pre-computed off-line.

```
1  $\mathbf{CS_R} \leftarrow$ context shapes from receptor protein $P_R$;
2  $\mathbf{CS_L} \leftarrow$ context shapes from ligand protein $P_L$;
3  Candidate-Pairs $\leftarrow \emptyset$;
4  foreach (Context Shape $CS_R$ in $\mathbf{CS_R}$) do
5      foreach (Context Shape $CS_L$ in $\mathbf{CS_L}$) do
6          Calculate the average Solid Angle sum $\Theta$ over $CS_R$ and $CS_L$;
7          Calculate angle $\phi$ between Solid Vector of $CS_L$ and Reversed Solid Vector of $CS_R$;
8          if ($\phi$ and $\Theta$ exceed corresponding pruning thresholds) then
9              Skip to next context shape in $\mathbf{CS_L}$;
10         foreach (Pose $\pi$ of context shapes $CS_R$ and $CS_L$) do
11             Calculate the overlap volume $OV$, under the given pose $\pi$;
12             if ($OV$ exceeds threshold value) then
13                 Reject pose $\pi$;
14             Calculate the buried surface area $BSA$, under the given pose $\pi$;
15         Only keep the best pose $\pi$ with the largest buried surface area;
16         Insert the tuple ($CS_R$, $CS_L$, $\pi$, $BSA$) in Candidate-Pairs;
17 Sort Candidate-Pairs based on $BSA$ (decreasing order);
```

Figure 6: Context Shape Algorithm

In the online matching step, given the matching table and the set of context shapes from the ligand and receptor proteins, our matching algorithm outputs a ranked list of possible poses. The pseudo-code for our context shape matching algorithm is given in Figure 6. For each pair of context shapes, CS_R in the receptor protein P_R (line 4) and CS_L in the ligand protein P_L (line 5), we first apply the solid angle and vector pruning (line 8). We next enumerate all the possible poses π (line 10) using the matching table. Of the poses that pass the filters on the overlap volume (line 12), we retain only the pose that has the largest buried surface area (line 15). Finally the candidate pairs of context shapes, along with their pose, are sorted based on their buried surface area, and the top ranked ones are reported (line 17).

It is worth mentioning that we adopted a database oriented approach to the problem of setting the various parameter settings, and for better experiment management. Given two sets of context shapes, one from a receptor and the other from a ligand, the algorithm generated a set of SQL (Structure Query Language) statements to insert the pose information, including partial scores, into a SQLite database (`http://www.sqlite.org/`). Then we wrote various SQL scripts to extract and analyze the data using different parameter settings. For example, we tried varying the weights for the different surface layers and varying pruning cutoffs while optimizing the ranks of the true poses.

3 Results and Discussion

To evaluate the effectiveness of our context shapes (CS) approach, we compared it with two state-of-the-art geometry-based docking algorithms: ZDOCK(PSC) [7] (a FFT based approach; also called ZDOCK v2.1) and PatchDock [11, 30] (a Geometric Hashing based approach, along with other optimizations). The executable code for both methods was downloaded directly from their respective web-sites: `http://zlab.bu.edu/zdock/` for ZDOCK v2.1 and `http://bioinfo3d.cs.tau.ac.il/PatchDock/` for PatchDock. Unless noted otherwise, all experiments were done on a PC with dual-core 2.2Ghz Opteron CPU, with 32GB of physical memory, running Linux.

In the evaluation, a prediction is called a *hit* if the interface RMSD between the predicted ligand pose and the original ligand (after superimposing them), is less than a threshold value (e.g., $2.5\mathring{A}$). The same threshold value was used for ZDOCK(PSC) and PatchDock. In all three methods an interface RMSD value is calculated over the interface C_α atoms of the ligand protein. A C_α atom of the ligand protein is in the interface if any atom of the receptor is within a distance of $10\mathring{A}$. Note also that for all three methods, for each complex, the ligand is randomly rotated before docking.

3.1 Comparison with ZDOCK(PSC) and PatchDock

We compared our new CS approach against ZDOCK(PSC) and PatchDock on the 84 test cases from benchmark v2.4 [23]. For all three methods only the top 3600 predictions are considered, since ZDOCK(PSC) returns a maximum of 3600 predictions. Three sets of results are presented: i) *R-bound/L-bound*: Here the receptor and ligands are both bound, i.e., the receptor and ligand from the co-crystallized protein complexes are used. ii) *R-unbound/L-bound*: Here the receptor is taken from the unbound form of the protein, whereas the ligand is taken from the bound co-crystallized complex. iii) *R-unbound/L-unbound*: Here both receptor and ligand are in their unbound form. In all three cases, as mentioned above, randomly rotated ligands are used to test each method. In the results shown below, we mainly refer to the *best ranked hit*, which is not necessarily the same as the *best rmsd hit*. Please note that a *hit* always has an RMSD lower than the chosen cut-off (like $2.5\mathring{A}$); it is the best ranked hit, if it has the *least rank*, and it is best rmsd hit if it has the *least rmsd*, among all hits.

For CS, MSMS [29] was used to calculate the SES of both the receptor and ligand proteins, with probe-radius $1.4\mathring{A}$ and surface point density of two points per $1\mathring{A}^2$. The surface layers were computed using a 3D grid with a resolution of $(0.2\mathring{A})^3$. The sparse critical surface points were calculated based on the analytical representation of the SES. On the receptor protein P_R, we used only the centers of re-entrant faces, whereas on the ligand protein, only the centers of contact faces were used. In both cases, if the area of a re-entrant or contact face was less than $1\mathring{A}^2$, it was ignored. If a contact face was bigger than $4\mathring{A}^2$, we evenly re-sampled the face to obtain one point per $2\mathring{A}^2$. These sparse critical surface points were used as the centers of the context shapes. The radius of the template sphere was $r = 16\mathring{A}$.

ZDOCK(PSC) can be run in two modes: coarse or dense. In the default coarse mode, used in this and previous studies [7], the rotational sampling density is of 15 degrees, while in dense mode it is 6 degrees. The dense mode is around 15 times slower than the coarse mode, and is suggested to be used only if further refinement of the structures is required. PatchDock was also run under its default parameters. Note that the results on ZDOCK(PSC) reported below, cannot be directly compared with those reported in [8], since the latter presents results on cases from benchmark v0.0, whereas we consider the latest benchmark v2.4. Furthermore, the latter uses the much slower dense sampling mode, whereas we use the coarse mode in this study.

We first trained/optimized the parameters of our CS approach on 25 R-bound/L-bound cases from the docking benchmark v0.0 [6]. In other words, we optimized CS on the receptor and ligand parts of the co-crystallized protein complexes. The results are shown in Table I. The table gives details on the complexes used, as well as the rank and RMSD for the best ranked hits (with RMSD cut-off of $2.5\mathring{A}$). Out of the 25 cases, CS failed to return a hit among the top 3600 predictions for two cases, namely 1DFJ and 2PCC. On 16 cases the best rank hit is found in the top 10 predictions.

After optimizing the default parameters for CS, we did a comprehensive evaluation of CS against ZDOCK(PSC) and PatchDock on 84 test cases from the most recent docking benchmark v2.4 [23]. Benchmark v2.4 has 12 cases in common with the subset of 25 complexes from benchmark v0.0. Note that whereas 1AHW appears in both benchmarks, it refers to different complexes with different chains. Furthermore, even the common cases do not refer to identical complexes, presumably because benchmark v2.4 refers to the latest cleaned PDB (Protein Data Bank) structures. It is important to note that CS was optimized only on the 25 bound complexes (R-bound/L-bound) from benchmark v0.0, but it was tested on an independent test set of 84 bound complexes as well as the corresponding unbound conformations (R-bound/L-bound, R-unbound/L-bound and R-unbound/L-unbound) from benchmark v2.4. In other words the test cases were not used for training the parameters of CS.

We first consider the R-bound/L-bound scenario. Table II shows the best ranked hit, along with its RMSD, for each of the 84 test cases from benchmark v2.4, for the three methods. Note that the 12 common cases with benchmark v0.0 are included here only for completeness. Note also that the results on the common cases do not match those reported in Table I due to the differences in the complexes between the two benchmarks. On the 84 test cases, a RMSD cut-off of $2.5\mathring{A}$ was used to obtain a hit. In the vast majority of the cases (56 out of 84) the best ranked hit for CS had an RMSD under $2\mathring{A}$. In comparison ZDOCK returns 39, and PatchDock 26 cases under $2\mathring{A}$. On 10

Table I: 25 R-bound/L-bound complexes from docking benchmark v0.0 [6] used for optimizing the parameters of CS. *PDB* column gives the PDB id for the protein complex, as well as the chains used as the receptor and ligand. *Atoms* gives the size of the receptors and ligands. *RMSD* and *Rank* give RMSD and the rank of the best ranked hit (using a RMSD cut-off of 2.5$\mathring{A}$).

PDB	Atoms	Rank	RMSD ($\mathring{A}$)
1ACB:E,I	(1733, 521)	6	1.97
1AHW:DE,F	(1737, 1595)	1	1.85
1AVW:A,B	(1631, 1268)	2	1.28
1AVZ:B,C	(873, 461)	1	2.08
1BRC:E,I	(1642, 410)	16	1.26
1BRS:A,D	(863, 692)	5	2.37
1BVK:DE,F	(1742, 983)	38	1.91
1CGI:E,I	(1798, 439)	1	1.18
1CHO:E,I	(1748, 399)	7	1.33
1CSE:E,I	(1919, 521)	11	1.09
1DFJ:I,E	(3410, 950)	-	-
1DQJ:AB,C	(1666, 1000)	87	1.73
1FSS:A,B	(4225, 463)	82	1.78
1MAH:A,F	(4104, 459)	1	1.89
1MDA:HL,A	(3378, 790)	223	2.29
1MLC:AB,E	(3288, 1000)	5	1.27
1TGS:Z,I	(1628, 415)	3	1.37
1UGH:E,I	(1807, 646)	1	1.05
1WEJ:LH,F	(1702, 822)	135	1.56
1WQ1:G,R	(2531, 1321)	1	1.76
2KAI:AB,I	(1790, 438)	6	1.17
2PCC:A,B	(2370, 846)	-	-
2PTC:E,I	(1628, 445)	2	2.15
2SIC:E,I	(1937, 763)	7	1.68
2SNI:E,I	(1937, 512)	1	1.62

cases (as listed in the table), all three methods failed to return a hit in the top 3600 predictions. On the remaining 74 cases, CS failed only on two cases (1KAC and 2PCC), whereas ZDOCK failed on 19, and PatchDock failed on 32, additional cases.

Table III and Table IV show the results for R-unbound/L-bound and R-unbound/L-unbound scenarios, respectively. Here too, an RMSD cut-off of 2.5$\mathring{A}$ was used to report hits.

The results from all three scenarios are summarized in Table V. Consider first the results using the RMSD cut-off of 2.5$\mathring{A}$ for a hit. Table V(a) shows the number of cases with a success for all three methods, where success means that a hit was found in the top 3600 ranked predictions. The number of successes decreases sharply as we move from the bounded to unbounded scenarios. For example, on R-unbound/L-unbound, CS returned a hit in only 18 out of the 84 test cases, whereas

Table II: R-bound/L-bound: Comparisons between CS, ZDOCK(PSC) and PatchDock on 84 test cases from Benchmark v2.4. *PDB* gives the PDB id for the protein complex. *RMSD* and *Rank* give the RMSD and rank of the best ranked hit (using 2.5$\mathring{A}$ cut-off). 1DFJ*, 1FQ1, 1GHQ, 1HE8, 1IJK, 1NSN, 1SBB, 2HMI, 2PCC*, and 2VIS are not listed above, since all three methods failed to return a hit in the top 3600 predictions for these cases. * denotes complexes common with the 25 cases from benchmark v0.0 [6]. † and ‡ refer to the same protein, but using two different set of chains as the receptor.

	Context Shape		ZDOCK(PSC)		PatchDock	
PDB	Rank	RMSD ($\mathring{A}$)	Rank	RMSD ($\mathring{A}$)	Rank	RMSD ($\mathring{A}$)
1A2K	40	1.08	570	2.41	300	1.47
1ACB*	8	2.32	6	0.82	10	1.60
1AHW	7	1.20	56	1.18	40	1.55
1AK4	2925	2.08	3471	1.14	-	-
1AKJ	265	2.15	448	1.88	-	-
1ATN	49	2.10	558	1.15	-	-
1AVX	10	1.76	1	1.96	43	2.14
1AY7	193	1.23	46	1.68	24	2.07
1B6C	11	1.78	24	1.69	40	1.92
1BGX	1	1.96	-	-	-	-
1BJ1	1	1.05	3	1.42	-	-
1BUH	61	1.55	393	1.43	83	1.14
1BVK*	45	1.69	1087	1.43	131	2.12
1BVN	1	1.55	10	1.24	1	0.75
1CGI*	1	1.37	1	1.12	1	1.08
1D6R	4	1.68	35	1.04	-	-
1DE4	13	1.21	452	1.62	-	-
1DQJ*	67	1.65	19	2.00	83	1.71
1E6E	1	1.58	58	2.06	2	2.29
1E6J	1337	1.92	699	2.02	1706	1.43
1E96	1206	1.84	-	-	1767	1.44
1EAW	1	1.41	1	1.75	1	0.99
1EER	1	1.62	-	-	1	1.66
1EWY	518	2.26	-	-	139	1.42
1EZU	1	1.60	-	-	1	0.94
1F34	1	1.99	-	-	1	1.90
1F51	7	2.01	-	-	1	1.92
1FAK	1997	1.70	-	-	-	-
1FC2	7	1.85	55	2.18	49	1.24
1FQJ	12	1.94	120	1.94	248	1.48
1FSK	9	2.06	19	1.70	218	1.57
1GCQ	2	1.26	382	1.81	-	-
1GP2	53	1.86	-	-	-	-
1GRN	1	1.84	7	2.26	3	1.45
1H1V	14	2.37	1510	2.40	-	-
1HE1	1	1.44	7	1.67	1	1.06
1HIA	2	1.07	1	1.70	14	1.19
1I2M	6	1.36	14	1.80	-	-
1I4D	104	1.42	793	2.08	167	1.05
1I9R	-	-	1271	2.04	-	-
1IB1	2	1.48	-	-	-	-
1IBR	1	2.05	-	-	-	-
1IQD	14	1.19	55	1.83	-	-
1JPS	2	1.26	23	2.30	96	1.87
1K4C	5	0.88	30	1.16	337	1.53
1K5D	2	2.06	10	2.11	-	-
1KAC	-	-	381	1.52	-	-
1KKL	226	1.67	-	-	-	-
1KLU	1108	1.80	-	-	-	-
1KTZ	2280	1.41	-	-	-	-
1KXP	3	2.17	-	-	-	-
1KXQ	229	1.51	30	1.60	29	1.63
1M10	-	-	33	2.23	-	-
1MAH*	1	1.45	1	1.91	1	1.27
1ML0	569	1.91	75	1.94	7	0.58
1MLC*	30	1.15	1205	1.37	516	1.79
1N2C	3	1.36	-	-	-	-
1NCA	3	1.77	20	1.48	-	-
1PPE	1	2.32	2	1.21	1	1.03
1QA9	972	1.30	-	-	-	-
1QFW†	1247	2.21	16	2.46	-	-
1QFW‡	38	2.13	54	1.84	-	-
1RLB	311	1.63	-	-	3143	2.32
1TMQ	1	2.32	8	1.79	1	1.52
1UDI	3	1.52	1	1.50	1	1.97
1VFB	8	1.50	-	-	-	-
1WEJ*	496	1.25	1120	1.11	-	-
1WQ1*	1	1.14	4	2.04	1	0.84
2BTF	4	1.13	21	1.21	137	1.82
2JEL	56	1.40	532	1.77	282	1.65
2MTA	21	1.45	1447	2.26	115	1.71
2SIC*	4	1.36	9	1.19	-	-
2SNI*	2	1.27	4	2.50	13	2.10
7CEI	123	1.90	5	2.18	-	-

Table III: R-unbound/L-bound: Comparisons between CS, ZDOCK(PSC) and PatckDock on 84 test cases from Benchmark v2.4. *PDB* gives the test case. *RMSD* and *Rank* give the RMSD and rank of the best ranked hit (using 2.5Å cut-off). Results for cases where all three methods failed to return a hit in the top 3600 predictions are not shown.

	Context Shape		ZDOCK(PSC)		PatchDock	
PDB	Rank	RMSD (Å)	Rank	RMSD (Å)	Rank	RMSD (Å)
1ACB	-	-	1395	2.03	-	-
1AHW	3	2.46	127	2.47	-	-
1AVX	1022	2.36	291	1.67	2490	1.26
1AY7	667	1.79	-	-	1109	1.92
1B6C	128	2.35	541	1.82	-	-
1BJ1	1	1.05	3	1.42	-	-
1BUH	77	1.64	267	1.52	135	1.65
1BVN	24	1.13	-	-	27	2.50
1CGI	-	-	2879	2.41	-	-
1D6R	237	1.56	227	1.40	-	-
1DQJ	3317	1.96	232	1.91	-	-
1E6E	-	-	-	-	69	2.24
1EAW	6	2.40	23	1.92	23	1.17
1EWY	1325	2.20	-	-	-	-
1EZU	33	2.48	-	-	-	-
1F34	1	1.72	-	-	-	-
1F51	3061	2.47	-	-	-	-
1FSK	9	2.06	19	1.70	218	1.57
1GCQ	535	2.28	-	-	973	2.01
1GRN	266	1.98	997	2.18	73	1.52
1HE1	77	1.19	56	1.64	2	2.10
1HIA	16	1.31	7	1.68	-	-
1I4D	343	1.98	-	-	770	1.45
1I9R	55	1.64	-	-	-	-
1IQD	14	1.19	55	1.83	-	-
1JPS	1957	2.33	188	1.54	481	2.13
1K4C	5	0.88	30	1.16	337	1.53
1KAC	1489	2.05	-	-	-	-
1KXP	-	-	2	2.48	-	-
1KXQ	247	2.04	272	1.38	10	1.82
1M10	3092	2.49	-	-	-	-
1MAH	18	1.65	3	1.49	157	2.49
1ML0	2467	2.05	-	-	-	-
1MLC	361	2.15	-	-	-	-
1N2C	13	1.54	-	-	-	-
1NCA	3	1.77	20	1.48	-	-
1PPE	2	1.55	4	2.28	1	2.27
1QFW†	1247	2.21	16	2.46	-	-
1QFW‡	38	2.13	54	1.84	-	-
1SBB	-	-	-	-	3547	2.06
1TMQ	9	1.25	59	2.03	1	1.14
1UDI	109	2.12	496	2.40	169	2.46
1VFB	77	2.10	858	1.26	360	2.16
1WEJ	-	-	1289	0.96	-	-
1WQ1	12	1.77	1	2.42	3	1.43
2JEL	56	1.40	532	1.77	282	1.65
2MTA	29	1.60	744	1.75	-	-
2SIC	3069	2.30	150	1.19	-	-
7CEI	191	2.23	319	2.24	-	-

Table IV: R-unbound/L-unbound: Comparisons between CS, PatchDock and ZDOCK(PSC) on 84 test cases from Benchmark v2.4. *PDB* gives the PDB id for each test case. The *Rank* and *RMSD* of the best ranked hit (using 2.5$\AA$ cut-off) are also shown. Results for cases where all three methods failed to return a hit in the top 3600 predictions are not shown.

	Context Shape		ZDOCK(PSC)		PatchDock	
PDB	Rank	RMSD ($\AA$)	Rank	RMSD ($\AA$)	Rank	RMSD ($\AA$)
1AHW	402	2.46	-	-	181	2.49
1BJ1	1893	1.93	224	1.66	-	-
1BVK	-	-	-	-	2757	2.27
1BVN	34	2.41	76	1.11	-	-
1CGI	-	-	68	2.43	1120	2.11
1EAW	94	2.29	96	1.74	85	2.09
1F34	-	-	21	2.02	490	1.81
1FSK	20	1.57	16	1.15	221	2.39
1HE1	1029	2.17	-	-	-	-
1KXQ	2226	1.73	310	1.26	-	-
1MAH	597	1.16	176	1.55	887	2.38
1ML0	-	-	1	2.43	231	2.02
1MLC	18	2.28	2198	2.11	-	-
1NCA	26	1.79	-	-	-	-
1PPE	2	2.31	4	2.20	-	-
1QFW†	597	1.73	213	1.93	-	-
1QFW‡	33	2.32	-	-	-	-
1TMQ	783	1.68	-	-	1	1.96
1UDI	2469	2.14	-	-	27	2.42
1VFB	228	2.46	-	-	-	-
1WEJ	-	-	462	1.90	-	-
2MTA	-	-	-	-	515	2.19
2SIC	1077	2.28	1154	2.22	-	-
7CEI	2290	1.90	366	1.07	-	-

ZDOCK(PSC) returned 15 hits, and PatchDock only 11 hits. The number of instances where all three methods fail to return a hit is also shown. For the R-unbound/L-unbound cases, all three methods failed in 59 cases.

Table V(b) shows the win-tie-loss-failure records for CS versus ZDOCK(PSC) and PatchDock. Comparing CS against ZDOCK(PSC), for R-bound/L-bound, we find that CS returns a better ranked hit than ZDOCK in 57 cases, whereas ZDOCK returns a better hit in 14 cases. CS and ZDOCK tie in 3 cases, and both fail on 10 cases. On R-unbound/L-bound CS wins in 33 cases as opposed to 15 cases for ZDOCK. Finally on R-unbound/L-unbound cases, the win-loss record is 12-11. Comparing against PatchDock, CS once again has a better performance across all three

Table V: Summary of results for bound and unbound test cases using 2.5$\mathring{A}$ and 5.0$\mathring{A}$ cut-off for ranking. (a) Number of test cases with success (i.e., a hit found in the top 3600 predictions) for each method, and the number of test cases where all three methods fail (*All Fail*). (b) Win-tie-loss summary for CS versus ZDOCK and PatchDock; *both-fail* gives the number of test cases on which both methods fail. (c) Average running time over all 84 test cases.

RMSD <= 2.5$\mathring{A}$				
	CS	ZDOCK(PSC)	PatchDock	All Fail
R-bound/L-bound	71	55	42	10
R-unbound/L-bound	43	33	22	34
R-unbound/L-unbound	18	15	11	59

RMSD <= 5.0$\mathring{A}$				
	CS	ZDOCK(PSC)	PatchDock	All Fail
R-bound/L-bound	76	71	63	4
R-unbound/L-bound	52	52	50	19
R-unbound/L-unbound	41	43	38	35

(a)

RMSD <= 2.5$\mathring{A}$								
	CS vs. ZDOCK(PSC)				CS vs. PatchDock			
	win	tie	loss	both-fail	win	tie	loss	both-fail
R-bound/L-bound	57	3	14	10	54	11	6	13
R-unbound/L-bound	33	0	15	36	36	0	9	39
R-unbound/L-unbound	12	0	11	61	14	0	9	61

RMSD <= 5.0$\mathring{A}$								
	CS vs. ZDOCK(PSC)				CS vs. PatchDock			
	win	tie	loss	both-fail	win	tie	loss	both-fail
R-bound/L-bound	50	11	18	5	54	14	10	6
R-unbound/L-bound	29	2	29	24	34	4	22	24
R-unbound/L-unbound	20	0	28	36	27	0	19	38

(b)

	CS	ZDOCK(PSC)	PatchDock
Avg. Time	2126s	3091s	1164s

(c)

scenarios; it has 54-6, 36-9, and 14-9 win-loss record against PatchDock for the R-bound/L-bound, R-unbound/L-bound and R-unbound/L-unbound cases, respectively.

Table V (a) and (b) also show the comparative performance of CS versus ZDOCK and PatchDock, using an RMSD cut-off of 5.0$\mathring{A}$ for a hit. As expected, Table V (a) shows that across all three methods, more successes are obtained using the higher cut-off. Table V (b) shows the win-tie-loss-failure comparison. Compared to PatchDock, CS remains superior across all three scenarios. CS also outperforms ZDOCK(PSC) on the R-bound/L-bound cases. On R-unbound/L-bound, both methods perform equally. Finally, on R-unbound/L-unbound ZDOCK(PSC) has better results; CS returns better ranked hits in 20 cases, whereas ZDOCK does better in 28 cases.

Table V (c) summarizes the average running time (in seconds) for the three methods across all 84 test cases (for R-bound/L-unbound). Pre-processing time is not included. For example, for CS, the SES surface and context shapes are computed off-line. Likewise, the time to calculate the SES for PatchDock, and surface residues for ZDOCK, is also not included. We find that on average, PatchDock is the fastest approach. It is about 2 times faster than CS. On the other hand CS is about 1.5 times faster than ZDOCK(PSC).

3.2 Performance Analysis of CS

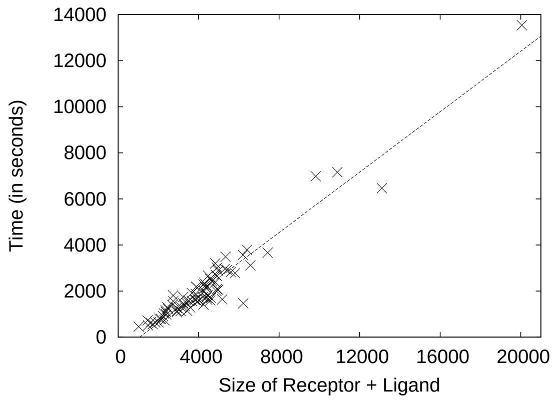


Figure 7: Scatter-plot of receptor plus ligand size versus the running time for CS on the 84 benchmark v2.4 test cases (R-bound/L-bound). The best fitting line through the points illustrates a clear linear relationship.

Figure 7 shows the scatter-plot of the combined receptor+ligand size (number of atoms) versus the running time (in seconds) for CS on the 84 test cases from benchmark v2.4 (R-bound/L-bound). We can see clearly that the running time for CS is linearly proportional to the number of atoms in the complex.

We also studied the effect of increased and decreased sampling of the context rays on the performance of CS. We selected a representative sample of 11 R-bound/L-bound complexes from Benchmark v2.4 so that the size of the receptor/ligand ranged from small to large. Table VI shows the ranks of the hits (with RMSD cutoff of 2.5Å) when we use coarser and denser sampling of the context rays. The base scenario, `CS(Base)`, used in the experiments above, used $\kappa = 1256$ context rays for the context shapes of the ligand (CS^L), and $3\kappa = 3768$ rays for the receptor (CS^R). For the dense sampling, `CS(Dense)`, we used twice as many rays, namely, $\kappa = 2518$ for CS^L and $2.5\kappa = 6295$ rays for CS^R. For the coarse sampling, `CS(Coarse)`, we used about half as many rays, namely,

Table VI: Effect of Coarse and Dense Ray Sampling in ContextShape. R-bound/L-bound results (with RMSD cutoff of 2.5$\mathring{A}$) shown on a sample of 11 representative complexes from Benchmark v2.4.

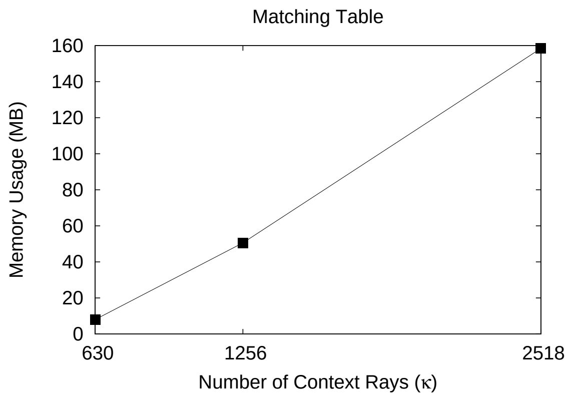
PDB	Complex	CS (Base)			CS (Dense)			CS (Coarse)		
ID	Size (R,L)	Rank	RMSD($\mathring{A}$)	Time	Rank	RMSD($\mathring{A}$)	Time	Rank	RMSD($\mathring{A}$)	Time
1BGX	(3245, 6570)	1	1.96	2461.0s	1	2.08	6596.8s	1	1.25	1104.0s
1DE4	(3063, 10044)	13	1.21	2354.3s	21	1.61	5943.5s	9	1.28	1068.9s
1FC2	(354, 1656)	7	1.85	378.3s	18	1.78	654.4s	6	1.83	127.5s
1GCQ	(468, 558)	2	1.26	216.0s	3	1.51	450.6s	2	1.11	75.7s
1GRN	(1522, 1586)	1	1.84	504.7s	1	1.29	1123.2s	1	2.34	191.7s
1HE8	(6070, 1358)	-	-	1341.1s	-	-	3349.4s	-	-	575.4s
1K4C	(3252, 765)	5	0.88	722.6s	5	1.01	1560.8s	54	1.89	278.7s
1KXP	(2767, 3431)	3	2.17	1265.8s	3	1.39	3202.9s	1	1.51	550.3s
1N2C	(15926, 4132)	3	1.36	4742.6s	12	1.79	12287.1s	5	1.61	1899.0s
1RLB	(3760, 1411)	311	1.63	655.9s	703	1.69	1601.2s	515	1.93	293.3s
2HMI	(7630, 3264)	-	-	2563.1s	-	-	6349.6s	-	-	1163.6s
Average Time				1564.1s			3919.9s			666.2s

$\kappa = 630$ for CS^L and $2.5\kappa = 1575$ rays for CS^R. We find that the dense sampling slightly decreases the quality of the ranking and RMSD values, since it may generate more false positives. On the other hand, the coarse sampling did not have a noticeable impact on the quality of the results. In fact, except for 1RLB where the ranks are significantly different, all three sampling scenarios have comparable results. Looking at the time, it is clear that coarser sampling results in faster times, whereas a denser sampling takes considerably more time. Thus by using a coarser sampling, it is possible to further improve the running times for CS, without a significant loss in the accuracy of the results.

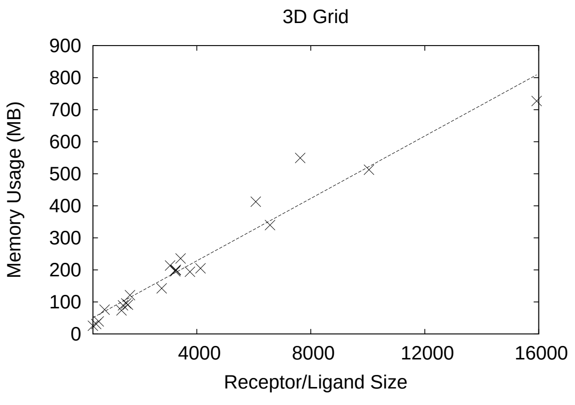
Finally, we analyzed the memory requirements of our CS approach in terms of the 3D grid size used to generate surface layers, the context shapes for the ligand/receptor, and the matching table. Figure 8(a) shows the memory usage for the matching table for the different context ray sampling scenarios. Note that the Matching Table size is independent of the protein size, but it does depend on the sampling density. We find that the coarse sampling takes about 8.0MB, the base sampling takes 50.6MB, and the dense sampling takes 150.5MB.

Figure 8(b) shows that the 3D grid used to generate the surface layers during context shapes creation, which depends on the size of the protein, can consume memory anywhere from 25.7MB for the smallest protein (1FC2, receptor with 354 atoms) to 727.2MB memory for the largest protein (1N2C, receptor with 15926 atoms). The results follow a clear linearly increasing memory usage trend with increasing protein (receptor/ligand) size.

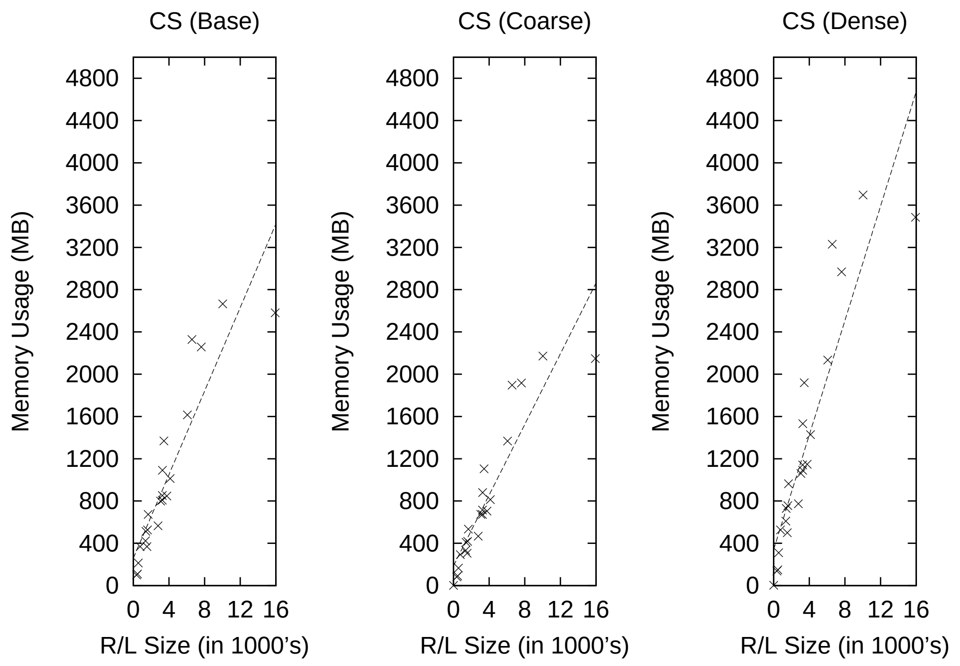
Figure 8(c) shows the memory usage trend for context shape generation, which clearly depends both on the size of the protein, as well as on the context ray sampling density. The memory usage for `CS(Base)`, with $\kappa = 1256$ rays, takes between 100.3MB–2580.8.5MB corresponding to the smallest and largest proteins. The memory consumption for `CS(Coarse)` with $\kappa = 630$ (half the



(a)



(b)



(c)

Figure 8: Memory usage for CS. (a) The size of the matching tables for the poses for different sampling densities (number of context rays κ). (b) The size of the 3D grid used to calculate the context shape surface layers. (c) The total memory usage for generating the context shapes for the receptor/ligand shown using different sampling densities for the context rays. `CS(Coarse)` has half as many, and `CS(Dense)` has twice as many rays compared to `CS(Base)`. Sub-plots (b) and (c) also show the linear regression line through the plotted points, indicating a clear linearly increasing trend of memory usage with increasing receptor/ligand (R/L) size.

number of rays), is 82.2MB–2147.5MB, and for `CS(Dense)` with $\kappa = 2518$ (twice the number of rays) is 137.5MB–3484.9MB. These results show a linear increase in the memory consumption with increasing protein size, as well as with increasing sampling density.

It is worth noting that we have not yet optimized the memory usage for CS. For example, when generating the context shapes, CS currently keeps them in memory and writes them to disk only after all of them have been generated. Furthermore, during the pose enumeration step, the entire set of context shapes from the receptor and ligand are read into memory. In fact, during context shape generation, only the grid needs to be kept in memory all the time, whereas each context shape can be immediately written to disk. Also, during pose matching, only the matching table needs to be in memory, and pairs of context shapes from the receptor and ligand can be read into memory as needed. However, the current implementation of CS, which is not memory optimized, requires considerably more memory as seen above.

3.3 Discussion

To better understand the strengths and weaknesses of our new CS approach, we did an analysis of the receptor-ligand interface, for those cases where CS does particularly well, and for those cases where it fails.

Figure 9 shows four test cases (R-unbound/L-unbound) where CS does particularly well, i.e., it returns a hit with 2.5$\AA$ RMSD cut-off. For example, for 1BVN and 1PPE, CS returns a better ranked hit than ZDOCK(PSC), whereas PatchDock fails, and for 1HE1 and 1NCA, only CS returns a hit, whereas both ZDOCK and PatchDock fail. As we can observe in the figure, the best ranked hit matches very well with the crystallographic (unbound) conformation. For comparison, the rank 1 prediction (which is not necessarily a hit) is also shown. For these good cases, we can see that even these rank 1 predictions are in the vicinity of the actual conformation (except for 1NCA, in the other three cases the RMSD for the rank 1 prediction is less than 25$\AA$).

Figure 10 shows four test cases (R-unbound/L-unbound) where CS fails to return a hit with RMSD cut-off of 2.5$\AA$, but either ZDOCK or PatchDock do return a hit. If we relax the RMSD cut-off, then even for these cases CS does return a hit. For example, CS found a hit for 1BVK with RMSD 5.01$\AA$, for 1ML0 with RMSD 6.85$\AA$, for 1WEJ with RMSD 9.06$\AA$, and for 2MTA with RMSD 2.70$\AA$. These best ranked hits are shown in the figure. Also shown are the rank 1 predictions. In contrast to the good cases above, for these bad cases, the rank 1 predictions have very high interface RMSD (except for 2MTA, the other three rank 1 predictions have RMSD more than 40$\AA$).

To further understand our CS approach, we show in Table VII(a), the best RMSD predictions, i.e., those with the least interface RMSD values compared to the actual unbound ligand, obtained for each of the 84 test cases. As summarized in Table VII(b), if we let cut-off for a hit to be 10$\AA$, then in 65 out of the 84 cases, CS does return a hit. In fact, in 82 out of the 84 cases, CS finds a hit within 16$\AA$ of the actual ligand. This implies that in the vast majority of the cases, CS finds a hit in the vicinity of the actual pose.

We also analyzed those instances among the 84 test cases that are hard for all three methods,

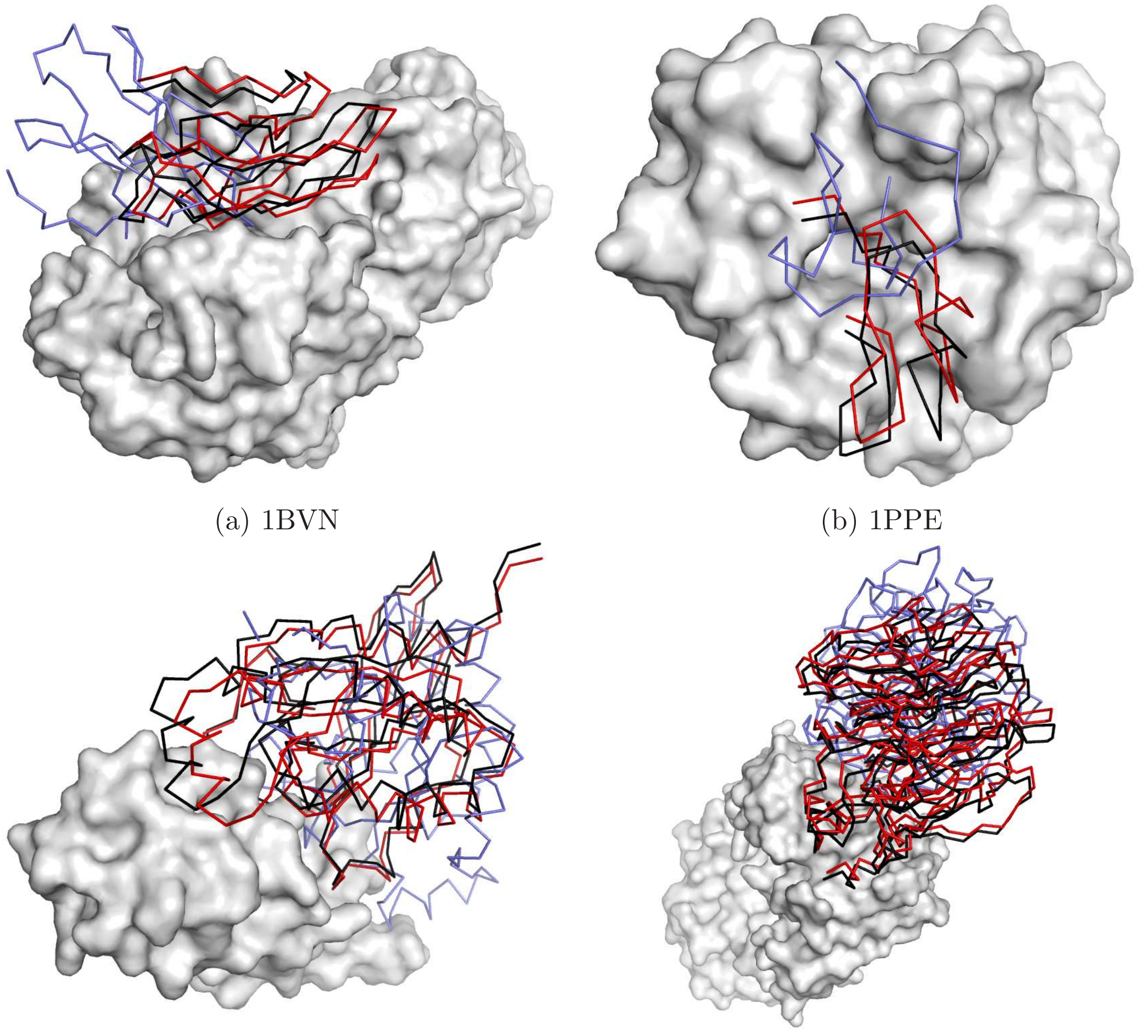
Table VII: ContextShape on R-unbound/L-unbound. (a) Best RMSD hits in the top 3600 predictions for all 84 Benchmark v2.4 test cases. (b) Summary of number of hits within a given RMSD cut-off value.

PDB	Rank	RMSD ($\AA$)	PDB	Rank	RMSD ($\AA$)
1A2K	1112	5.88	1I4D	652	7.37
1ACB	1449	8.87	1I9R	3367	3.38
1AHW	1810	1.55	1IB1	2806	12.01
1AK4	2584	9.86	1IBR	3303	7.46
1AKJ	3599	15.14	1IJK	2107	10.58
1ATN	2541	16.20	1IQD	2990	9.78
1AVX	3477	4.16	1JPS	1112	3.63
1AY7	3160	4.27	1K4C	984	7.81
1B6C	1856	2.90	1K5D	814	15.97
1BGX	1220	12.94	1KAC	2293	3.23
1BJ1	1893	1.93	1KKL	2247	4.14
1BUH	2372	2.58	1KLU	931	11.27
1BVK	532	5.01	1KTZ	3571	6.58
1BVN	34	2.41	1KXP	2591	2.73
1CGI	3261	4.38	1KXQ	2226	1.73
1D6R	1536	4.96	1M10	3472	10.74
1DE4	1933	13.46	1MAH	597	1.16
1DFJ	1291	3.51	1ML0	514	6.85
1DQJ	3452	7.37	1MLC	2650	1.87
1E6E	1568	6.09	1N2C	1309	11.20
1E6J	3231	2.66	1NCA	2746	1.19
1E96	2985	6.15	1NSN	1203	13.17
1EAW	94	2.29	1PPE	17	1.11
1EER	3023	19.43	1QA9	103	4.63
1EWY	2382	3.13	1QFW†	597	1.73
1EZU	301	4.88	1QFW‡	1207	1.82
1F34	161	2.72	1RLB	2223	6.10
1F51	3190	3.03	1SBB	2667	10.37
1FAK	3494	8.90	1TMQ	1092	1.54
1FC2	1732	7.19	1UDI	2469	2.14
1FQ1	3384	14.61	1VFB	228	2.46
1FQJ	937	8.24	1WEJ	2008	9.06
1FSK	275	1.33	1WQ1	159	3.92
1GCQ	1825	4.90	2BTF	1506	15.62
1GHQ	1327	7.72	2HMI	337	13.14
1GP2	1769	6.72	2JEL	2797	3.67
1GRN	3221	2.95	2MTA	1643	2.70
1H1V	3471	15.17	2PCC	3599	8.43
1HE1	1749	2.15	2SIC	1254	1.81
1HE8	2334	6.45	2SNI	1340	5.93
1HIA	1141	5.87	2VIS	3184	14.16
1I2M	1735	11.49	7CEI	2290	1.90

(a)

RMSD cut-off	Number of Hits
5$\AA$	41
10$\AA$	65
15$\AA$	78
16$\AA$	82
20$\AA$	84

(b)

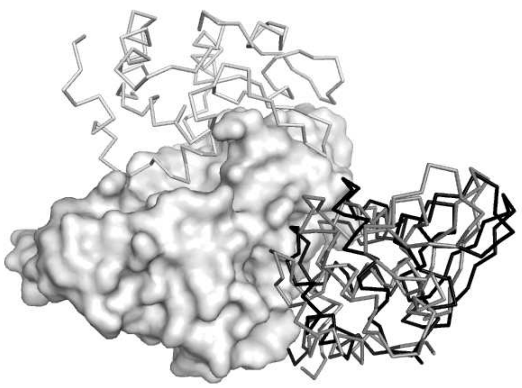


(a) 1BVN (b) 1PPE

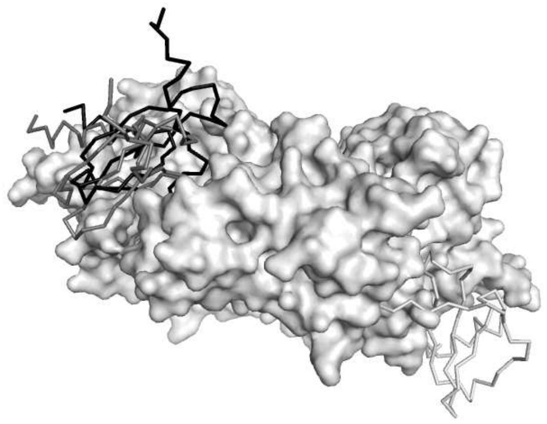
(c) 1HE1 (d) 1NCA

Figure 9: Good test cases for CS (R-unbound/L-unbound). Hits within 2.5$\mathring{A}$ RMSD were found for (a) 1BVN, (b) 1PPE, (c) 1HE1, (d) 1NCA. The unbound receptor surface is shown. The best ranked hit is shown in black, the original unbound ligand is shown in red, and the rank 1 prediction is shown in blue.

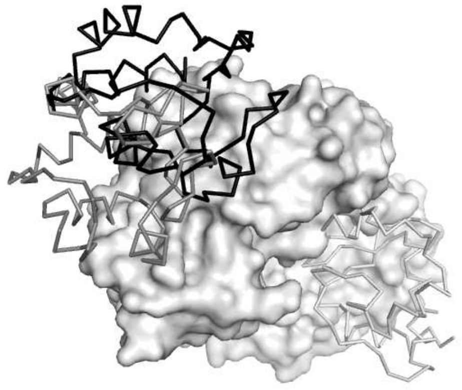
as shown in Figure 11. With a RMSD cut-off of 5$\mathring{A}$, with R-bound/L-bound, for 1GHQ, 2HMI, 2PCC, and 2VIS all three methods fail, whereas for 1FQ1 only PatchDock returns a hit, and for 1M10 only ZDOCK returns a hit. For R-unbound/L-unbound (with RMSD cutoff of 5$\mathring{A}$) all methods fail on 35 test cases (see Table V). Upon closer inspection, we find that these cases can be divided into three main classes. For example, 1GHQ and 2HMI, shown in Figure 11(a)-(b), have a very small interface (so do other proteins like 2VIS). 1M10 and 2BTF, shown in Figure 11(c)-(d), including other proteins like 1FQ1 and 1ATF, have relatively large cavities in the interface. 1DE4 and 1EER, shown in Figure 11(e)-(f), and other proteins like 1AKJ, 2PCC, and 1K5D, all have



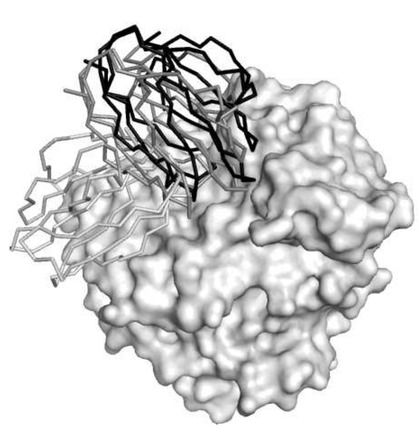
(a) 1BVK



(b) 1ML0



(c) 1WEJ



(d) 2MTA

Figure 10: Bad test cases for CS (R-unbound/L-unbound). No hits within 2.5$\mathring{A}$ RMSD were found for (a) 1BVK, (b) 1ML0, (c) 1WEJ, (d) 2MTA. The unbound receptor surface is shown. The best ranked hit is shown in black, the original unbound ligand is shown in grey, and the rank 1 prediction is shown in light grey.

enough differences between the bound and unbound conformations of the receptor and ligands, which confound purely geometric approaches.

Based on our analysis above, we can conclude that our CS approach almost always finds the largest complimentary surface, but this may be the wrong pose in some cases, since we do not consider electrostatics and hydrogen bonds (for bound cases), and we currently do not model flexibility (for unbound cases). The use of chemical properties, such as electrostatic potential, and modeling flexible docking, will most certainly improve the prediction accuracy, especially in the hard cases, and this is planned for future work.

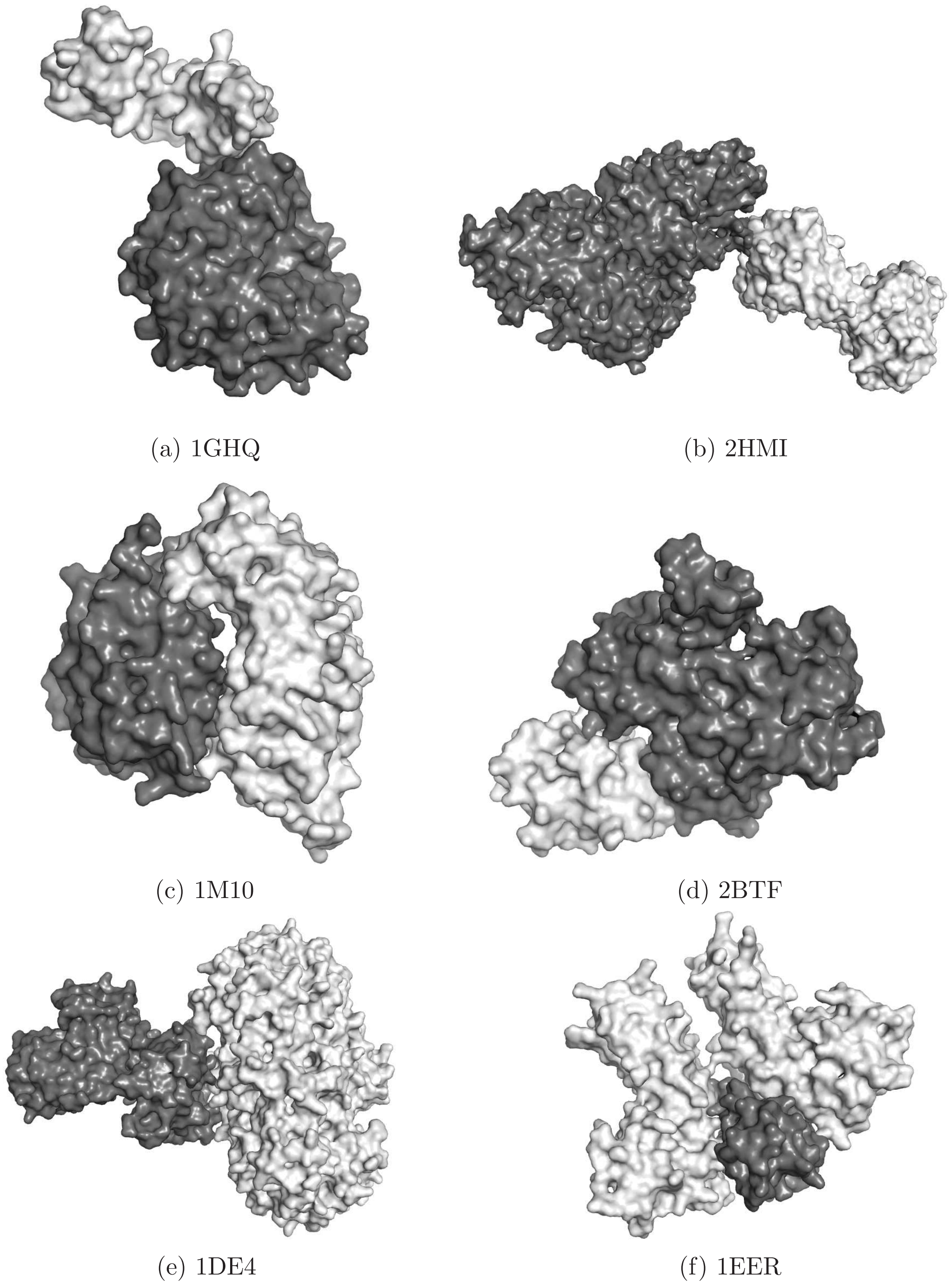


Figure 11: Hard Cases for Docking: (a)-(b) Small docking interface. (c)-(d) Voids in the interface. (e)-(f) Unbound conformations without a very good fit. The unbound receptor surface is shown in dark gray, whereas the unbound ligand surface is shown in light gray.

4 Conclusions

The success of our ContextShape method in correctly identifying docked poses shows that the assumptions that underly the method are correct for the most part. Specifically, we assumed that geometric complementarity of the interface surface plays a key role in protein docking, and furthermore that poses with superimposed surface points (PCPs) were the only important ones. We also assumed that the energy of a docked pose was best estimated as being proportional to the buried surface area, but only if the two shapes had little or no overlap. Success in the absence of any electrostatic considerations showed that shape complementarity plays a dominant role in determining binding specificity.

It is well-known that the conformation of a protein undergoes changes when it binds a ligand such as a substrate molecule or another protein [4]. This is often explained using the induced fit model [26]. The term unintentionally suggests a process in which binding happens first, and then a conformational change happens, improving the fit. That is, binding somehow induces a fit. This is how flexible docking is modeled in most cases [12, 18, 28, 35]. However, proteins exist in a dynamic equilibrium, constantly sampling slightly different conformational micro-states. The micro-state that exhibits the best fit for the ligand is the one that binds the ligand. This binding event effectively locks the protein into the bound conformation, shifting the dynamic equilibrium towards this micro-state. The result is the same, but in this view, the conformational sampling event occurs before binding. We believe, as do many others [1, 19], that modeling the conformational sampling can and should be done before, not after, binding. In short, the protein molecules can be modeled as an ensemble of conformational samples, and our context shapes approach can then be used on these ensembles to predict the good docking poses. We plan to extend our method to incorporate this in the future.

5 Acknowledgments

We thank the anonymous reviewers who gave valuable suggestions to improve the paper. This work was supported in part by NSF CAREER Award IIS-0092978, DOE Career Award DE-FG02-02ER25538, and NSF grants EIA-0103708, EMT-0432098, and DBI-0448072.

References

[1] R. Abagyan, M. Totrov, and D. Kuznetsov. ICM: A new method for protein modeling and design: Applications to docking and structure prediction from the distorted native conformation. *Journal of Computational Chemistry*, 15(5):488–506, 1994.

[2] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Res.*, 28(1):235–42, 2000.

[3] C. Bystroff. MASKER: Improved solvent excluded molecular surface area estimations using boolean masks. *Protein Eng*, 15:959–965, 2002.

[4] C. Bystroff and J. Kraut. Crystal structure of unliganded escherichia coli dihydrofolate reductase. ligand-induced conformational changes and cooperativity in binding. *Biochemistry*, 30(8):2227–39, 1991.

[5] C.J. Camacho and S. Vajda. Protein-protein association kinetics and protein docking. *Curr. Opin. Struct. Biol.*, 12:36–40, 2002.

[6] R. Chen, J. Mintseris, J. Janin, and Z. Weng. A protein-protein docking benchmark. *Proteins: Structure, Function and Genetics*, 52(1):88–91, 2003.

[7] R. Chen and Z. Weng. A novel shape complementarity scoring function for protein-protein docking. *Proteins: Structure, Function and Genetics*, 51(3):397–408, May 2003.

[8] R. Chen and Z. Weng. ZDOCK: An initial-stage protein-docking algorithm. *Proteins: Structure, Function and Genetics*, 52:80–87, 2003.

[9] M. L. Connolly. Shape complementarity at the hemoglobin α_1 β_1 subunit interface. *Biopolymers*, 25:1229–1247, 1986.

[10] M.L. Connolly. Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 221:709–713, 1983.

[11] D. Duhovny, R. Nussinov, and H. J. Wolfson. Efficient unbound docking of rigid molecules. In *2'nd Workshop on Algorithms in Bioinformatics*, pages 185–200, 2002.

[12] T.J.A. Ewing, S. Makino, A.G. Skillman, and I.D. Kuntz. DOCK 4.0: Search strategies for automated molecular docking of flexible molecule databases. *Journal of Computer-Aided Molecular Design*, 15(5):411–428, 2001.

[13] A. Fahmy and G. Wagner. TreeDock: A tool for protein docking based on minimizing van der walls energies. *Journal of American Chemical Society*, 124(7):1241–1250, 2002.

[14] D. Fischer, S.L. Lin, H.L. Wolfson, and R. Nussinov. A geometry-based suite of molecular docking processes. *J. Mol. Bio.*, 248:459–477, 1995.

[15] H.A. Gabb, R.M. Jackson, and M.J.E. Sternberg. Modelling protein docking using shape complementarity, electrostatics and biochemical information. *J. Mol. Bio.*, 272:106–120, 1997.

[16] I. Halperin, B. Ma, H. Wolfson, and R. Nussinov. Principles of docking: An overview of search algorithms and a guide to scoring functions. *Proteins: Structure, Function, and Genetics*, 47:409–443, 2002.

[17] E. Katchalski-Katzir, M. Eisenstein, A.A. Friesem, C. Aflalo, and I.A. Vakser. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proc. Nat. Acad. Sci. USA*, 89(6):2195–2199, March 1992.

[18] L. Li, R. Chen, and Z. Weng. RDOCK: Refinement of rigid-body protein docking predictions. *Proteins Structure Function and Genetics*, 53(3):693–707, 2003.

[19] D.M. Lorber, M.K. Udo, and B.K. Shoichet. Protein–protein docking with multiple residue conformations and residue substitutions. *Protein Science*, 11:1393–1408, 2002.

[20] J. G. Mandell, V. A. Roberts, M. E. Pique, V. Kotlovyi, J. C. Mitchell, E. Nelson, I. Tsigelny, and L. F. Ten Eyck. Protein docking using continuum electrostatics and geometric fit. *Proteins Engineering*, 14(2):105–113, 2001.

[21] R. Mendez, R. Leplae, M. F. Lensink, and S. J. Wodak. Assessment of capri predictions in rounds 3-5 shows progress in docking procedures. *Proteins: Structure, Function and Genetics*, 60:150–169, 2005.

[22] R. Mendez, R. Leplae, L.D. Maria, and S.J. Wodak. Assessment of blind predictions of protein-protein interactions: current status of docking methods. *Proteins: Structure, Function and Genetics*, 52:51–67, 2003.

[23] J. Mintseris, K. Wiehe, B. Pierce, R. Anderson, R. Chen, J. Janin, and Z. Weng. Protein-protein docking benchmark 2.0: An update. *Proteins: Structure, Function and Genetics*, 60(2):214–216, 2005.

[24] R. Nussinov and H.J. Wolfson. Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques. *Proc. Nat. Acad. Sci. USA*, 88:10495–99, December 1991.

[25] P.N. Palma, L. Krippahl, J.E. Wampler, and J.G. Moura. Bigger: a new (soft) docking algorithm for predicting protein interactions. *Proteins: Structure, Function and Genetics*, 39(4):372–84, 2000.

[26] J.M. Rini, U. Schulze-Gahmen, and I.A. Wilson. Structural evidence for induced fit as a mechanism for antibody-antigen recognition. *Science*, 255(5047):959, 1992.

[27] D.W. Ritchie and G.J.L. Kemp. Protein docking using spherical polar fourier correlations. *Proteins: Structure, Function and Genetics*, 39(2):178–94, 2000.

[28] B. Sandak, H.J. Wolfson, and R. Nussinov. Flexible docking allowing induced fit in proteins: Insights from an open to closed conformational isomers. *Proteins Structure Function and Genetics*, 32(2):159–174, 1998.

[29] M.F. Sanner, A.J. Olson, and J.-C. Spehner. Fast and robust computation of molecular surfaces. In *11th ACM Symposium on Computational Geometry*, 1995.

[30] D. Schneidman-Duhovny, Y. Inbar, V. Polak, M. Shatsky, I. Halperin, H. Benyamini, A. Barzilai, O. Dror, N. Haspel, R. Nussinov, and H. J. Wolfson. Taking geometry to its edge: fast unbound rigid (and hinge-bent) docking. *Proteins: Structure, Function and Genetics*, 52(1):107–12, 2003.

[31] G.R. Smith, M.J.E. Sternberg, and P.A. Bates. The relationship between the flexibility of proteins and their conformational states on forming protein–protein complexes with an application to protein–protein docking. *J. Mol. Biol.*, 347(5):1077–101, 2005.

[32] I.A. Vakser. Protein docking for low-resolution structures. *Protein Engineering*, 8(4):371–7, 1995.

[33] A. Via, F. Ferre, B. Brannetti, and M. Helmer-Citterich. Protein surface similarities: a survey of methods to describe and compare protein surfaces. *Cellular and Molecular Life Sciences*, 57:1970–1977, 2000.

[34] C. Wang, N. Karpowich, J.F. Hunt, M. Rance, and A.G. Palmer. Dynamics of atp-binding cassette contribute to allosteric control, nucleotide binding and energy transduction in abc transporters. *J. Mol. Biol.*, 342(2):525–537, 2004.

[35] J. Wang, P.A. Kollman, and I.D. Kuntz. Flexible ligand docking: a multistep strategy approach. *Proteins*, 36(1):1–19, 1999.

[36] H.J. Wolfson and I. Rigoutsos. Geometric Hashing: An overview. *IEEE Computational Science & Engineering*, 4(4):10–21, 1997.

[37] A. Zanzoni, L. Montecchi-Palazzi, M. Quondam, G. Ausiello, M. Helmer-Citterich, and G. Cesareni. MINT: a molecular interaction database. *FEBS Letters*, 513(1):135–140, 2002.