Published in final edited form as:

Proteins. 2012 January ; 80(1): 221–238. doi:10.1002/prot.23192.

Surface-histogram, a new shape descriptor for protein-protein docking

Shengyin Gu^{*},

Department of Computer Science, University of California, Davis, CA 95616

Patrice Koehl,

Department of Computer Science and Genome Center, University of California, Davis, CA 95616

Joel Hass, and

Department of Mathematics, University of California, Davis, CA 95616

Nina Amenta

Department of Computer Science, University of California, Davis, CA 95616

Shengyin Gu: sgu@ucdavis.edu; Patrice Koehl: koehl@cs.ucdavis.edu; Joel Hass: hass@math.ucdavis.edu; Nina Amenta: abamenta@ucdavis.edu

Abstract

Determining the structure of protein-protein complexes remains a difficult and lengthy process, either by NMR or by X-ray crystallography. Several computational methods based on docking have been developed to support and even serve as possible alternatives to these experimental methods. In this paper, we introduce a new protein-protein docking algorithm, shDock, based on shape complementarity. We characterize the local geometry on each protein surface with a new shape descriptor, the surface-histogram. We measure the complementarity between two surface-histograms, one on each protein, using a modified Manhattan distance. When a match is found between two local protein surfaces, a model is generated for the protein complex, which is then scored by checking for collision between the two proteins. We have tested our algorithm on Version 3 of the ZDOCK protein-protein docking benchmark. We found that for 110 out of the 124 test cases of bound docking in the benchmark, our algorithm was able to generate a model in the top 3600 candidates for the protein complex within an RMSD of 2.5 Å from its native structure. For unbound docking predictions, we found a model within 2.5 Å in the top 3600 models in 54 out of 124 test cases. A comparison with other shape-based docking algorithms demonstrates that our approach gives significantly improved performance for both bound and unbound docking test cases.

Keywords

protein-protein docking; protein surface; shape descriptor; surface-histogram

1 Introduction

Proteins are ubiquitous and essential biomolecules that are involved in nearly all biological processes, such as cell signaling, immune responses, cell cycle control and gene expression. Most proteins do not function as a monomer: they interact instead with other proteins to form complexes that vary in size and complexity, from simple dimers to large nano-

^{*}Corresponding author; sgu@ucdavis.edu.

machines involving hundreds of individual proteins. Finding the structure of such complexes using NMR spectroscopy and/or X-ray crystallography is often difficult, time consuming and expensive. There are many structures of protein complexes in the current PDB; however most of them are from homo- or heterodimers and very little is known about much bigger complexes. While there is hope for progress in the experimental characterization of these large systems with the recent developments in cryo-EM spectroscopy ¹, it remains unlikely that there will be a high-throughput structural genomics effort on solving the structures of a large number of protein-protein complexes in the near future ². In response to these experimental difficulties, there has been a long-standing interest in developing computational methods for predicting the structure of a complex from its constituent proteins. These methods aim at solving the following problem, referred to as the *protein-protein docking problem*: given two proteins known to interact and whose unbound structures are known, find the conformation of their complex. This problem is not yet solved, as the laws that govern protein assembly have not yet been unraveled. Most attacks on the problem generate not one, but an ensemble of possible conformations for the protein complex of interest. These then need to be ranked and/or tested against experimental data.

There is an extensive literature on protein-protein docking techniques, reviewed in ²⁻⁶. Most of these follow a hierarchical algorithm ^{2,4,6} involving three steps, namely:

1. Generation of a large collection of candidate structures for the complex, usually starting from coarse representations of the proteins that are treated as if they were as rigid. Flexibility is handled indirectly by allowing for small overlaps of the protein surfaces in the candidate docked conformation.
2. Refinement of these coarse candidate conformations with addition of all atoms, usually followed by energy minimization.
3. Scoring of the refined structures followed by selection of the “best” candidate(s).

This strategy is used by most teams that participate in the blind prediction experiment for protein docking, Critical Assessment of Predicted Interactions (CAPRI) ⁷, who then select ten of their “best” candidates for submission. Our work focuses on the first step of this strategy, with the aim to produce and rank several thousand candidate conformations for the protein complex of interest.

Finding a good model for the structure of a protein complex based on the knowledge of the structures of each of the individual proteins requires a procedure for searching the conformational space that defines the position of one of the proteins (usually referred to as the *ligand*) with respect to the other (the *receptor*), as well as a method for scoring any potential solution. As the conformational space considered is large (of dimension at least 6, as it is defined by 3 rotations and 3 translations in the simple case of rigid docking), the search procedure needs to be computationally efficient without compromising quality. These requirements are not easy to meet simultaneously, and the search for a good strategy has been the subject of many developments over the last twenty years. Monte Carlo methods are often considered a good compromise between speed and quality of coverage for sampling and as such have been used for protein-protein docking. ICM-DISCO ⁸ and RosettaDock ⁹ for example, two of the state-of-the-art algorithms that participate in CAPRI, use rigid body Monte Carlo minimization in rotational and translational spaces in their initial exploration of the relative position of the two proteins. RosettaDock represents these proteins as backbones plus side-chain centroids, and scores the configurations using residue-scale interaction potentials that account for residue environment as well as residue-residue interactions. ICM-DISCO samples the rotational and translational degrees of freedom that define the position of one protein with respect to the other using a pseudo-Brownian Monte Carlo minimization technique. Its interaction energy is represented with grid potentials that include a van der

Waals potential, an electrostatic potential, a hydrogen-bonding potential and a hydrophobicity potential.

Other approaches for solving the sampling problem in docking rely on geometric techniques. Among those, the shape-based methods have attracted the most attention. The main idea is to encode the shapes of both proteins in a feature space that allows for fast identification of complementarity. The most popular method uses a 3D grid representation of the proteins and the Fast Fourier Transformation (FFT) to compute the scoring function that measures the fit between the two 3D grids. This approach was pioneered by Katchalski-Katzir¹⁰ and is now found in docking algorithms such as ZDOCK¹¹, PIPER¹², MolFit¹³ and others^{14–16}. In the initial versions of this method, the scoring function reflected only shape complementarity; more recent versions also include simple electrostatics and desolvation energy terms.

Other shape-based approaches are based on shape descriptors. A shape descriptor for a region on a surface is a vector that describes the geometric features of the region. Many shape descriptors have been proposed to identify complementary regions on two proteins. In PatchDock^{17,18}, the molecular surface of a protein is segmented into concave, convex and flat patches. The local shape descriptor is then constructed using two points in a patch and their corresponding normals; it includes Euclidean and geodesic distances between the points and the three angles formed by the pair of points and their normals. The patches are then matched using geometric hashing of their shape descriptors. Poses, implied by matching patches, are then clustered and scored. Other shape descriptors, such as spherical harmonics¹⁹, context shapes²⁰, 3D Zernike descriptors²¹ and spin images²² have also been used. The context shapes (or CS) algorithm places a sphere at a point on the surface, then emits many rays from the center of the sphere to capture local shape features. Each ray is represented by a bit string of 1's and 0's which correspond to the inside and outside of the surface, respectively. The set of binary vectors represents the surface at that point. Spherical harmonics were originally introduced as possible shape descriptors for protein surface matching by Ritchie and Kemp¹⁹. Correlations between the spherical harmonics expansion coefficients are used to accelerate the search for shape complementarity. The 3D Zernike descriptors²³ extend the spherical harmonics descriptors by including volumetric information, and they have been applied to docking in the algorithm LZerD²¹. Spin images are widely used shape descriptors in computer vision that have also been used to identify similar protein surfaces²².

In this paper, we introduce:

1. a new shape descriptor,
2. a distance measure between two shape descriptors,
3. an algorithm for fast descriptor matching, using the cover tree data structure,
4. a fast collision scoring procedure, and
5. a new docking algorithm based on 1–4.

The shape descriptor, which we refer to as the *surface histogram*, captures the local geometry around a set of two points with given normals on the surface of a protein. Matching two surface-histograms immediately gives a potential docking pose. This property is shared with some existing descriptors (those used in PatchDock, and shape contexts), while others (spin images, or descriptors based on the normals of spherical harmonics or Zernike moments) match points but are invariant under some group of rotations. This is a design trade-off: a descriptor which is not rotation invariant may suffer from false negatives, because of noise in the choice of rotation or normal direction. On the other hand, it quickly

eliminates many incorrect matches which might be retained by a descriptor invariant to such noise. We handle the possibility of false negatives by sampling the molecular surfaces densely with surface-histograms, so that there should be multiple matches attesting to a docking configuration.

Dense sampling in turn requires well-chosen data structures and algorithms. We use a modified Manhattan distance as a measure of complementarity between two surface-histograms, which allows us to find matches using a practical data structure for high-dimensional neighbor finding, the cover tree²⁶. Our algorithm then filters potential poses using a global collision detection procedure on the two proteins. Like most other geometric methods used for docking, we treat the proteins as rigid bodies and handle flexibility implicitly with a “soft dock” approach, by allowing a small amount of surface overlap when evaluating collision and ranking candidates. We have tested our approach on the bound and unbound test cases included in the ZDOCK Benchmark 3.0²⁴. A comparison with other shape-based docking algorithms, CS²⁰, ZDOCK¹¹, PatchDock^{17,18} and LZerD²¹, demonstrates the improved performance achieved with our approach. Particularly in bound docking, our method finds more cases with at least a near-native configuration than all other methods at various ranking thresholds (top 1, 5, 10, 100, 1000, 2000 and 3600).

2 Methods

2.1 Outline

We focus on finding the geometric complementarity of two proteins. We develop a local shape descriptor, called the *surface-histogram*. This local shape descriptor captures the shape of a region on the surface of a protein and stores this information into a vector of numbers. This encoding makes it easier and faster to identify complementary surface shapes on the two proteins, which we refer to as *matching*.

In the first step of our docking algorithm, shDock, we generate a dense sampling of surface-histograms on the surface of one of the two proteins, say protein Q , and store them for matching. We then randomly sample the other protein P and generate surface-histograms. For each surface-histogram from P , we find a list of the best matching surface-histograms from protein Q .

Matching surface-histograms indicates local geometric fit of two proteins; however it does not guarantee their global fit. Local protein surfaces are quite similar to each other. This similarity leads to a great many false positives. In the second step, we filter out false positives using a global collision detection test. We then rank the remaining configurations by a contact score which is an estimate of the contact surface area. The final candidate list contains the predicted conformations with the highest contact scores.

2.2 Surface-histogram construction

The first step in our procedure is to generate a triangular mesh on the surface of the protein of interest, preferably a uniform mesh. Then, to construct a surface-histogram, we pick two vertices, i and j , from this surface mesh, such that the distance l between i and j is between 8 Å and 8.2 Å. Note that we consider all vertex pairs at a fixed distance, rather than all vertex pairs, so that the number of surface-histograms we consider is not quadratic in the number of mesh vertices. The vertex coordinates, together with the unit surface normals at the vertices, are used to define a surface-histogram, as follows.

We begin by defining a local coordinate frame. The origin is defined as the middle point of i and j . We define the x -axis to point in the direction of the vector $j - i$. We let the average of the two vertex normals, n_i and n_j , determine the y -axis. Taking the average slightly reduces

the sensitivity to noise in the normals. We experimented with averaging all vertex normals in a local neighborhood, but found that it did not improve the quality of the matching based on the surface-histograms, while it did incur a cost in running time. Finally, the z -axis is determined by computing the cross product of the x and y axes vectors.

We then consider a rectangular 3D volume centered at the origin of the local frame and with edges parallel to the axes (see Figure 1). The size of the 3D volume is defined with two parameters: the bin size and the number of bins. We use a rectangular volume rather than a cubical volume, as the shape of the interface region between two proteins is often long and narrow; this is in contrast to earlier methods such as spin images¹⁷ and context shapes²⁰ that use symmetric regions. In all experiments reported below, we have used a bin size of 1.2 Å and 11, 10, 5 for the number of bins in the x , y , and z directions, respectively. These choices are based on computer experiments. The 3D volume intersects the mesh surface and all the bins that are cut by the mesh surface are marked.

Next we turn the 3D volume into a 2D image by projecting along the y -axis direction (see Figure 2). For each xz pixel in the 2D image, we define a voxel xyz as *inside* if there exists a voxel $xy'z$ that intersects the surface mesh, with y' being greater than y . For each xz , the 2D image value is then the number of inside voxels that correspond to that xz . This projection does not preserve the full 3D information when an xz position includes multiple y values at which the surface intersects the volume (the foldover issue). However, as the regions represented by surface-histograms are local, we believe that little information is lost in practice. This projection greatly reduces the size of the descriptor and enables faster matching.

Our surface-histograms bear some resemblance to the multi-scale surface descriptors of Cipriano *et al.*²⁵. Their surface descriptor is computed locally for a point i on the surface mesh and a circular region $N(i)$ on the surface around i . Each point in $N(i)$ is represented as a height field on the surface tangent plane at i . The main differences between the two descriptors are:

1. The multi-scale descriptor uses a circular region.
2. It constructs a smaller and less informative descriptor.
3. It does not define a local coordinate frame that can later be used to dock two regions whose descriptor matches.

2.3 Surface-histogram matching

Our goal in histogram matching is to identify shape complementarity between two proteins P and Q . When computing the surface-histograms, we reverse the surface normals on protein Q and leave the surface normals on protein P unchanged, so that by measuring the similarity of pairs of surface-histograms, one from each protein, we are measuring the local complementarity of the two surfaces.

To measure the similarity of two surface-histograms, we define a modified Manhattan distance which differentiates between surface *gaps* and *overlaps*. An overlap is an xz location in which the surfaces interpenetrate. A gap is a location in which the surfaces fail to meet. Overlaps are scored more negatively than gaps. Given two surface-histograms A from protein P and $\bar{B}$ (with surface normals reversed) from protein Q , the distance between them is:

$$d(A, \bar{B}) = d(\bar{B}, A) = \sum_{ij} d(a_{ij}, \bar{b}_{ij}) \quad (1)$$

where the summation extends over all bins ij in the 2D image, with the individual bin distance defined as:

$$d(a_{ij}, \bar{b}_{ij}) = \begin{cases} (a_{ij} - \bar{b}_{ij}) & \text{if } a_{ij} \geq \bar{b}_{ij} \quad (\text{overlap}) \\ c(\bar{b}_{ij} - a_{ij}) & \text{otherwise} \quad (\text{gap}) \end{cases} \quad (2)$$

The constant c is a number between 0 and 1 representing the penalty for gaps relative to overlaps. Through numerical experimentation, we find that $c = 0.1$ works well and this is what we use in the experimental data presented here.

Recall that each surface-histogram contains a local coordinate frame (an origin together with x , y , and z axes). By aligning the two coordinate frames, we find a transformation that locally aligns proteins P and Q . Thus surface-histogram matches immediately imply potential docking conformations.

2.4 Matching speed-up

Given two proteins, P and Q , we select the smaller of the two; let us assume this is Q . For each vertex q_i on the surface mesh Q_m of Q , we find all its neighbors q_j on Q_m such that the Euclidean distance $l(q_i, q_j)$ belongs to the interval $[8 \text{ \AA}, 8.2 \text{ \AA}]$. For each pair consisting of q_i and a neighbor q_j , we negate the normals, then use the vertex coordinates and the negated normals to create a surface-histogram. We store all the surface-histograms from Q . We then attempt to match surface-histograms from P with the stored surface-histograms from Q .

Brute-force matching of all surface-histograms from Q with all surface-histograms of P would be prohibitively expensive. Instead, we think of each 2D 11×5 surface-histogram as a point in an 11×5 dimensional space and use a cover tree²⁶, a data structure for high-dimensional nearest-neighbor searching, to perform fast matching. The time complexity of the nearest neighbor query using cover tree is proportional to the “intrinsic data dimension”, the dimension of the (possibly non-linear) subspace within which the data lies, rather than the dimension of the space itself. Protein surfaces are geometrically similar and therefore the intrinsic dimension of the surface-histogram space is much lower than the actual dimension based on the number of bins, i.e. less than $11 \times 5 = 55$ dimensions.

The correctness of the cover tree for answering nearest neighbor queries using standard Euclidean distance was proved in²⁶. We provide a proof in Appendix A that the construction of and the query on a cover tree is also valid using our modified Manhattan distance.

We then sample a percentage of mesh vertices from the partner protein P (we use 10% in our experiments). For each of the selected vertices, we find its neighbors that are between 8 \AA and 8.2 \AA away, and compute the corresponding surface-histograms. We then fix a number T , and for each surface-histogram computed, we find the T best matching surface-histograms from Q that satisfies a matching score threshold.

2.5 Collision filtering and contact score

After finding the top T matching surface-histograms, we filter the matching pairs $[p_i, p_j]$ and $[q_i, q_j]$ using a collision test. Large contact area between the two proteins is generally preferred, while big collisions between them are undesirable.

To set up the collision filtering, protein P is represented by a 3D volumetric grid with cubic grid cells of size 1.5 Å. Cells are marked as core and surface as follows. The grid cells occupied by the atoms (defined by atom centers and corresponding radii) are marked as core cells. To mark surface cells, we superimpose the mesh representation of the surface of P onto the grid. A cell is a surface cell if it contains a mesh vertex. Depending on the relative surface mesh resolution and 3D grid cell size, interpolation of vertices on the mesh is sometimes required to obtain an unbroken surface. To do so, we place extra points at the middle of each edge and at the centroid of each face of the surface mesh. Note that alternatively, a finer mesh representation of the surface of P could also be used to solve this problem.

To perform the collision filtering, we dock protein Q by aligning the local frame at $[p_i, p_j]$ with the one at $[q_i, q_j]$. This alignment transformation is applied to all mesh vertices on Q . There is a collision and we reject the pair $[p_i, p_j]$ and $[q_i, q_j]$, if any vertex on Q , after transformation, is contained in a core cell of P .

The surface cells on P containing transformed vertices of the surface mesh of Q are called *contact cells*. The number of such contact cells is expected to be high for a good match, as it would correspond to a large contact surface area between P and Q . We therefore define a contact score CS that is a function of the number of contact cells; this definition differs for the bound and unbound docking test cases, as presented below.

Finally, we rank all configurations that pass the collision filtering test according to their contact scores; the top scoring configurations are the candidate docked complexes.

2.6 Differences between bound and unbound docking cases

There are two main types of protein-protein docking experiments. In bound docking, the input is the structure of the complex formed by two proteins. The procedure is then to separate the two proteins and to reconstruct the complex; no conformational changes in the individual structures are involved. In unbound docking, the input is the structures of the two proteins in their native conformations and the goal is to predict the structure of the complex they form; conformational changes involving backbone and side-chains are to be expected. We treat these two problems slightly differently.

In the bound docking experiment, we know that there exists a very good fit between the two protein structures. In this case, we make our test for contacts very stringent and define the contact score CS as

$$CS = n_1 \quad (3)$$

where n_1 is the number of contact cells found in the surface layer of the 3D grid representing the protein P . We use one layer of surface cells as described in the previous section. We reject any configuration if there is a collision cell. In addition, we use a high resolution mesh to represent the surface of the proteins, with a resolution of 0.8 Å.

For unbound docking we want to allow configurations with slight overlaps, to account for possible conformational changes implicitly. In the collision detection and contact scoring computation step, we therefore use two layers of surface cells on protein P . To generate the second surface layer, for each surface mesh vertex v on P , we compute a new vertex $v_2 = v - cs \times n_v$ where cs is the grid cell size and n_v is the unit normal associated with v pointing outwards. The 3D grid cells containing the set of vertices v_2 form the second surface layer. The contact score is then defined as:

$$CS = n_1 - n_2 \times 2 \quad (4)$$

where n_2 is the number of contact cells in the second surface layer. A contact cell in the second surface layer is undesirable and it would have been considered a collision for bound docking. In unbound docking, we penalize it by assigning a negative weight to reduce the total contact score. Note that in the unbound case, the cell size and the mesh resolution are reduced to 1.0 Å. In addition, we place additional points at the middle of each edge and at the centroid of each facet of the surface mesh to create an unbroken cell representation of the surface.

2.7 Docking test cases

We use the ZDOCK protein-protein docking benchmarks, both version 2.0²⁷ and 3.0²⁴. Benchmark 2.0 includes 84 cases of protein docking pairs while Benchmark 3.0 includes an additional 40 cases for a grand total of 124. These protein docking pairs cover different categories (enzyme/inhibitor or enzyme/substrate complexes, antibody/antigen, antigen/bound antibody and others) and are sorted into three difficulty levels (rigid-body, medium difficult and difficult). Each test case includes the PDB structures of a pair of proteins in their bound and unbound states. We conduct both bound and unbound docking experiments. All experiments are performed on Benchmark 3.0, except for comparison with published results from other docking methods, that are only available for Benchmark 2.0. While the difficulty level is meaningful for unbound test pairs, all bound pairs should be considered as rigid-body docking.

To compute surface-histograms on protein surfaces, we generate mesh surfaces from the PDB structures. We use the dms program²⁸ provided by the Resource for Biocomputing, Visualization, and Informatics (RBVI) center at UCSF to create the dotted molecular surface which is a discretization of the Connolly surface²⁹. We use the default parameters which give us about 5 points/Å² and the default “water” probe radius of 1.4 Å. We then use Geomagic Studio (<http://www.geomagic.com>), a commercially available meshing program, to turn the dotted molecular surfaces into nearly uniform surface meshes.

We use the uniform surface meshes to compute surface-histograms. For collision detection and contact score computation, we use both the PDB structures and the uniform surface meshes.

3 Results and Discussion

In this section, we validate first the effectiveness of surface-histograms matching. Next we show that our algorithm for protein-protein docking performs well by applying it to the ZDOCK Benchmark 3.0²⁴ on both bound and unbound test cases. Then we compare our results to those of four successful shape-based docking algorithms: CS, ZDOCK, PatchDock and LZerD, some of which have consistently been among the best docking algorithms in CAPRI challenges⁶. We also document the running time of our algorithm and show that it is reasonably fast. Finally, we analyze the cases in which our algorithm does not work well.

3.1 Effectiveness of surface-histograms for bound docking

We evaluate the effectiveness of surface-histograms in identifying complementary local regions with a ROC curve analysis³⁰. We use the bound test case labeled 1AKJ in the ZDOCK Benchmark as a representative example. It corresponds to the complex between the human proteins CD8 and the MHC molecule HLA-A2, which we label as P and Q , respectively. We use this complex as the test case in several of our experiments, for consistency.

We compute their respective surface meshes, P_m and Q_m , as described in the method section. A vertex v on surface mesh P_m is labeled an interface point if there is a vertex on mesh Q_m that is within 1 Å of v . We study all vertex pairs $[p_i, p_j]$ on mesh P_m where both p_i and p_j are interface vertices; we call them *interface pairs* on P . We define the true neighbors of $[p_i, p_j]$ as the interface pairs $[q_i, q_j]$ on Q_m where q_i and q_j are among the nearest points (we take the 7 nearest points) on Q_m to p_i and p_j respectively. For each interface pair $[p_i, p_j]$, we compute its surface-histogram, match it with the surface-histograms of all pairs $[q_i, q_j]$ on Q_m and sort all $[q_i, q_j]$ pairs according to their scores as defined by Equation 2, from the best to the worst. For varying thresholds, all pairs below the threshold are assumed possible matches (positive), and all above it unlikely matches (negative): the pairs that are true neighbors (according to the definition above) are called true positives (TP) while those that do not are false positives (FP). We plot the corresponding ROC curve (i.e. TP against FP as the threshold is varied) for each interface pair; and compute the area under the curve. The final area is normalized so that it is between [0,1]. If the surface-histograms successfully capture the local geometry of the surface of P and Q and our method successfully detects the complementarity between these surface-histograms, these ROC curves should have a large area below them.

Figure 3 shows all 109 interface pairs from 1AKJ reported in descending order of their ROC area. Of the 109, 69 pairs have a ROC area of 0.8 and above, and 32 pairs have a ROC area between 0.5 and 0.8. The average ROC area of all pairs is 0.816. A method producing random outcomes would lead to a ROC area of 0.5. This experiment shows that surface-histograms do capture relevant information about local shape complementarity.

3.2 Effectiveness of surface-histograms for unbound docking

While in the previous section we could measure the effectiveness of surface-histograms directly in the bound case through ROC analysis, the definition of interface pairs and true positives is not as straightforward in the unbound case. Instead, we analyze the contribution of the surface-histograms in the unbound case by comparing it with random sampling, again on the 1AKJ complex.

In this experiment, we compare our procedure of randomly choosing pairs of vertices on P and searching for similar surface-histograms on Q to a completely random process: randomly choose a vertex-pair, at the appropriate 8.0–8.2 Å distance on P and match it with a randomly chosen one from Q . We generate the same number of configurations with the two procedures, and compare the number of near-native configurations found. We define a configuration to be near-native if its interface RMSD to the bound structure is no more than 2.5 Å (RMSD definition detail is presented in section 3.3).

As a baseline, we find that on the bound test cases surface-histogram matching for the bound produces 9.9 times as many near-native configurations as the random sampling does; so this level of enrichment corresponds to the positive ROC analysis above. As one would expect, surface-histogram matching is less effective in the unbound case: it produces only 4.5 times as many near-native configurations as random sampling. Clearly, backbone and side-chain motions change the local surface shapes and make finding near-native configurations more

difficult. Nonetheless we see that the effectiveness of surface-histogram matching is only decreased by a bit more than a half. The complete data is presented in Table 1.

3.3 Bound and unbound docking experiments

In this section, we evaluate the entire docking algorithm, shDock, on the bound and unbound cases of the ZDOCK protein-protein docking benchmark. Initial-stage docking algorithms generally produce a few thousand candidate configurations. This number allows for refinement by a second stage algorithm in a reasonable time. Here we report results for the top 3600 candidates. These are ranked according to the contact scores defined by Equations 3 and 4, for bound and unbound docking, respectively. The number of candidates is set at 3600 for consistency with previous studies^{20,21}. We also report statistics on various ranking thresholds less than 3600.

Our algorithm performed well on both bound and unbound cases. In the following, RMSD is defined as the coordinate RMS computed over the C_{α} atoms in the interface of the complex after optimal superposition. The interface is defined as all residues with at least one atom located within 10 Å of the other protein. We define a hit (or a near-native configuration) to be a structure for which the RMSD to the bound structure is no more than 2.5 Å. These definitions are consistent with those given in previous studies^{11,20,21}. The *best RMSD prediction* is the candidate with the smallest RMSD among all selected conformations while the *best ranked hit* is the rank of the first hit, i.e. the first candidate with an RMSD below 2.5 Å. Note that the rank of this best ranked hit (which we refer to as the *best rank* below) can be larger than 3600.

In Figure 4, we show the best RMSD of the selected 3600 candidates for each of the 124 cases in Benchmark 3.0. Out of 124 bound cases, 110 (88.71%) have a best RMSD of 2.5 Å or less, 108 (87.10%) 2 Å or less, 105 (84.68%) 1 Å or less and 74 (59.68%) 0.5 Å or less. We find that (see Figure 5), 40 out of 124 cases (32.36%) have a best rank of 1, 57 (45.97%) of 5 or less, 63 (50.81%) of 10 or less, 87 (70.16%) of 100 or less, 103 (83.06%) of 1000 or less, 105 (84.68%) of 2000 or less and 110 (88.71%) of 3600 or less.

For unbound docking, 54 out of 124 cases (43.55%) have an RMSD of 2.5 Å or less, 35 cases (28.23%) of 2 Å or less and 4 cases (3.23%) of 1 Å or less (Figure 4). In addition, 2 cases (1.61%) out of 124 cases have a best rank of 10 or less, 9 cases (7.26%) of 100 or less, 30 cases (24.19%) of 1000 or less, 48 cases (38.71%) of 2000 or less and 54 (43.55%) of 3600 or less (Figure 5).

3.4 Comparison with other docking algorithms

We compare the performance of our docking algorithm, shDock, to those of four successful shape-based docking algorithms: CS, ZDOCK, PatchDock and LZerD. PatchDock and ZDOCK have consistently been among the best protein docking algorithms⁶ in recent CAPRI challenges. We show that our docking algorithm outperforms these four methods on bound docking test cases and that it outperforms CS and is competitive with PatchDock and LZerD on unbound test cases (with ZDOCK being consistently better performing).

Bound docking experiments—Bound docking results for the 84 test cases included in the ZDOCK Benchmark 2.0²⁷ (which is a subset of Benchmark 3.0) have been reported in the literature for CS, ZDOCK (version 2.1), PatchDock and LZerD (in Venkatraman *et al.*²¹, Additional File 2). For comparison, we have run our method shDock on the same test cases.

All five methods are compared on the same number of candidates for each test case (3600); these candidates are ranked according to the scoring scheme specific to the method.

For each method, we report the number of near-native conformations that have been found as well as the ranking of these hits among all non-native predictions. Table 2 shows the number of cases with at least one hit found by the five methods at different ranking thresholds: shDock always find a larger number of cases. We also observe that the RMSDs for the best ranked hits obtained with shDock are significantly smaller than those obtained with the other programs. Table 3 summarizes the pair-wise comparison of shDock with each of the other four methods with respect to ranking. We observe that our method produces a hit with a better rank in 40, 59, 55 and 48 cases than CS, ZDOCK (PSC), PatchDock and LZerD respectively. There are 14, 6, 9 and 10 cases for which shDock ties with CS, ZDOCK (PSC), PatchDock and LZerD respectively, and 5, 8, 8 and 7 cases where neither method finds a hit. There are 5 cases for which none of the five methods finds a hit.

Overall, these results show that shDock performs significantly better than the other four methods on these bound docking test cases. It is able to produce more cases with at least one hit in the top 3600 candidates and it is able to score these hits with high ranks among all candidates it generates. We provide full results on each individual test case in Appendix B.

Unbound docking experiments—Bound docking experiments are good model for docking cases with minimal structural flexibility. In blind prediction cases however, and in real docking problems, only the native structures of the proteins that bind to form a complex are known, and these structures may deform upon formation of the complex. Such situations are better modeled with unbound docking experiments. We have compared the quality of unbound predictions obtained with shDock with those of the competing four methods considered above, with one difference. While in the bound cases we are focusing mostly on geometry, it is expected that in unbound test cases, the score used to rank the conformations is robust enough to account implicitly for structural flexibility. We have implemented a modified version of the scoring function in shDock for unbound test cases (see Methods). In the latest version of ZDOCK, version 3.0.1³¹, a new physics-based scoring function has been implemented for similar reasons. We have therefore used this version of ZDOCK for comparison with the version of shDock tuned for unbound docking. We note that physics-based scoring has not yet been incorporated into shDock.

We present the analysis of unbound docking in two parts. First, we compare our method with CS and LZerD. These comparisons are based on published results for ZDOCK Benchmark 2.0^{20,21}. Second, we compare the results of shDock to those of ZDOCK and PatchDock for Benchmark 3.0.

For all 84 unbound test cases in ZDOCK Benchmark 2.0, we report the number of cases with at least one hit by each method (CS, LZerD and our method shDock) for various ranking thresholds (Table 4) as well as a pair-wise comparison of these methods with respect to their ability to rank these near-native conformations (Table 5). We find that our method consistently performs better than CS and is comparable with LZerD. (LZerD was already reported to perform better than CS²¹). As mentioned by Kihara and colleagues, CS seems better suited for bound docking test cases. We note that the average RMSD of the best ranked hits are similar for the three methods.

Next we compare shDock with ZDOCK and PatchDock on all 124 unbound test cases included in ZDOCK Benchmark 3.0. We find that our method and PatchDock are competitive, while ZDOCK performs systematically better (see Tables 5 and 6). Among the 124 test cases, there are 84, 52 and 54 cases with at least one hit in the top 3600 candidates

for ZDOCK (version 3.0.1), PatchDock and shDock respectively. ZDOCK (version 3.0.1) systematically finds more cases with at least one hit at all ranking threshold levels than PatchDock and shDock, the latter two performing similarly. Note that the average RMSD of the best ranked hits are similar. This indicates that ZDOCK does not generate better models, but rather that its scoring function is better suited for ranking these models and identifying the near-native among all candidates. This is consistent with previous analysis of the quality of docking algorithms. Ventatraman *et al.* reported that ZDOCK version 2.1 and LZerD had comparable performance on the unbound test cases of ZDOCK Benchmark 2.0²¹. We have shown that LZerD and shDock have similar performance on the same test cases. As version 3.0.1 of ZDOCK performs better than shDock, it appears that the improvement comes from the differences between versions 2.1 and 3.0.1, most likely from the introduction of the new physics-based scoring function³¹.

The full results for all unbound docking test cases for all five methods considered here are given in Appendix C.

3.5 Running time

It is important to consider the running time of an algorithm as it determines whether the method is usable in practice. In our algorithm, the first steps of computing and storing the surface-histograms are fast, as is the matching procedure (with the cover tree data structure we have implemented, see Methods). The bottlenecks are the steps that implement the filtering and the scoring of the many false positives generated. We have found that the running times for shDock are comparable (i.e. of the same order or magnitude) to the running times of the other methods considered in this study. Running times were measured on a desktop computer using an Intel i7 CPU 960 processor running at 3.20GHz, with 4 cores; each core has two threads, and the total amount of RAM available to the 8 threads is 12 GB. We ran 8 test cases simultaneously. For 124 unbound docking cases, we find the average running time for shDock to be 2758 seconds with the shortest and longest running times being 284 seconds and 14234 seconds respectively. For ZDOCK (version 3.0.1), the average, shortest and longest running times are 1160 seconds, 261 seconds and 8310 seconds respectively, while for PatchDock, the same running times are 1211 seconds, 25 seconds and 5174 seconds respectively. We did not run CS and LZerD; the running times reported in the literature for those two methods however are similar to those provided in this paper.

3.6 Docking challenges: contact area and conformational changes

We have shown above that shDock is a robust and better performing method for bound docking and a promising technique for unbound docking. Out of the 124 test cases included in the ZDOCK Benchmark 3.0, it is able to find a hit (i.e. a near-native prediction within 2.5 Å of the actual conformation of the complex) in 110 and 54 of 124 cases for bound and unbound docking, respectively.

We have examined the cases on which shDock fails, to identify steps in the procedure that may be improved. For bound docking, we observe that in failure cases the interface regions are small and sometimes have a hole (forming a complex with non-spherical topology). This is illustrated in Figure 6 for the test case 1AK4. This affects both the quality of the surface-histogram matching and the ranking based on the *contact score*. As an estimate of the relative size of the interface region, we define the contact ratio to be the contact score divided by the total number of surface cells in the collision detection grids on both proteins. In Figure 7, the RMSD of the best prediction found among the top 3600 candidates is plotted against this contact ratio. All test cases for which no hits were found in the top 3600

candidates have small contact ratios (less than 0.02). On the other hand, our method works well for all cases with contact ratios larger than 0.02.

In unbound docking, the shapes of the two proteins may change upon formation of the complex. This is especially important when it occurs in the regions that form the interface. We observe that the performance of shDock on the unbound docking test cases is related to the amount of interface conformational change. The ZDOCK Benchmark 3.0 provides a quantitative measure of the latter, defined as the coordinate RMS distance of the C_{α} atoms of the interface residues between their bound and unbound conformations. We find that there is a high correlation (0.723) between the best RMSD found with our unbound algorithm and the interface conformation change (see Figure 8).

3.7 Visualization of false positives

Like all first stage docking algorithms, shDock finds many false positives, even in the case of bound docking. We believe this is because there are indeed many configurations with geometric fit as good as that of the true positives, as well as large contact area. In Figure 9, we use Molsurfer³⁵ to visualize some of the false positive interfaces for bound docking on our example case 1AKJ. The overlaps and gaps are colored as red and blue respectively, while white indicates regions in which the molecular surfaces are close. Only the parts where the surfaces are within 3 Å of each other are shown. We see our predicted top three candidates all have larger contact area, particularly more white area, than the ground-truth configuration and the near-native configuration predicted (ranked 7th in the results with an RMSD of 0.68 Å). The large interface contact areas can also be inferred from Figures 10 and 11, which illustrate the relationship between the contact score and RMSD for the bound and unbound docking experiments related to 1AKJ, respectively.

This visualization also shows slightly more overlap in our best RMSD conformations than in native conformation. This suggests a direction for improvement, at least for bound docking.

4 Conclusion

Our docking algorithm is based on a shape descriptor, the surface-histogram, designed specifically for matching complementary molecular surfaces. Because protein-protein interface regions are often elongated rather than round, we compute surface-histograms over a rectangular rather than a round region, in contrast to descriptors such as spin images³² or context shapes²⁰. The surface-histogram is computed in a local coordinate system defined by the local geometry, so that it is rotationally and translationally invariant with respect to the input orientations of the proteins, and a pair of approximately complementary surface-histograms immediately imply a candidate docking conformation to test. The drawback of this approach is that we need two nearby points to define a local coordinate system, leading to a large number of surface-histograms.

Several features of the design also address computational efficiency. We represent local geometry as a height function on a 2D space; this compressed form of the 3D data allows for faster matching. We also design the surface-histogram so that the distance from perfect complementarity of a pair of histograms is greater for shapes that interpenetrate than for those that do not. While this distance function is non-Euclidean, we prove that it nonetheless can still be used with the cover tree data structure which allows fast matching.

In bound docking, the contact score calculation is particularly successful at filtering out false positives and assigning high ranks to true positives. Our algorithm outperforms the other four methods we have tested, namely CS, LZerD, ZDOCK and PatchDock. In unbound docking experiments however, conformational changes obscure the geometric fit and our

method performs less well than ZDOCK (version 3.0.1), though it compares well with the three other methods. Based on our results, the best initial stage unbound docking algorithm is clearly ZDOCK (version 3.0.1). Our method, shDock, uses purely geometric information, whereas other shape-based algorithms also consider the physical chemistry of the input proteins. In particular, ZDOCK (version 3.0.1) uses a physics-based scoring function, and we believe that this is a major factor in its superior performance in unbound docking. Incorporating such information into shDock, both in matching and in scoring, would greatly improve the results.

Dealing explicitly with the flexibility is another direction for improvement. We plan to build on our excellent results for bound docking by generating an ensemble of low-energy high-resolution structures for each input protein, as described in ³³, and perform docking of the two ensembles.

Although our algorithm is competitively fast already, it is also inherently highly parallelizable. GPU programming has been shown to improve the speed of many parallelizable applications by a factor of 5 to 425 ³⁴. In future work, we plan to implement our algorithm on GPU using CUDA.

Acknowledgments

We gratefully acknowledge the support of NSF grant number CCF-0625744. P.K. acknowledges support from the NIH.

References

1. Baker M, Zhang J, Luttkie S, Chiu W. Cryo-EM of macromolecular assemblies at near-atomic resolution. *Nature Protocols*. 2010; 5:1697–1708.
2. Ritchie DW. Recent progress and future directions in protein-protein docking. *Current protein and peptide science*. 2008; 9:1–15. [PubMed: 18336319]
3. Halperin I, Ma B, Wolfson H, Nussinov R. Principles of docking: an overview of search algorithms and a guide to scoring functions. *Proteins: Struct Func Genet*. 2002; 47:409–443.
4. Smith GR, Sternberg MJ. Prediction of protein-protein interactions by docking methods. *Curr Opin Struct Biol*. 2002; 12:28–35. [PubMed: 11839486]
5. Méndez R, Leplae R, Maria LD, Wodak SJ. Assessment of blind predictions of protein-protein interactions: current status of docking methods. *Proteins: Struct Func Genet*. 2003; 52:51–67.
6. Vajda S, Kozakov D. Convergence and combination of methods in protein-protein docking. *Curr Opin Struct Biol*. 2009; 19:164–170. [PubMed: 19327983]
7. Janin J. Protein-protein docking tested in blind predictions: the CAPRI experiment. *Mol BioSyst*. 2010; 6:2351–2362. [PubMed: 20725658]
8. Fernández-Recio J, Totrov M, Abagyan R. ICM-DISCO Docking by Global Energy Optimization With Fully Flexible Side-Chains. *Proteins: Struct Func Genet*. 2003; 52:113–117.
9. Gray JJ, Moughon S, Wang C, Schueler-Furman O, Kuhlman B, Rohl CA, Baker D. Protein-protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations. *J Mol Biol*. 2003; 331:281–299. [PubMed: 12875852]
10. Katchalski-Katzir E, Shariv I, Eisenstein M, Friesem AA, Aflalo C, Vakser IA. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proc Natl Acad Sci (USA)*. 1992; 89:2195–2199. [PubMed: 1549581]
11. Chen R, Li L, Weng Z. ZDOCK: an initial-stage protein docking algorithm. *Proteins: Struct Func Bioinfo*. 2003; 52:80–87.
12. Kozakov D, Brenke R, Comeau SR, Vajda S. Piper: an FFT-based protein docking program with pairwise potentials. *Proteins: Struct Func Bioinfo*. 2006; 65:392–406.
13. Kowalsman N, Eisenstein M. Inherent limitations in protein-protein docking procedures. *Bioinformatics*. 2007; 23(4):421–426. [PubMed: 17040920]

14. Gabb HA, Jackson RM, Sternberg MJE. Modelling protein docking using shape complementarity, electrostatics and biochemical information. *J Mol Biol.* 1997; 272:106 – 120. [PubMed: 9299341]
15. Vasker IA. Protein docking for low-resolution structures. *Protein Eng.* 1995; 8:371–377. [PubMed: 7567922]
16. Mandell JG, Roberts VA, Pique ME, Kotlovyy V, Mitchell JC, Nelson E, Tsigelny I, Eyck LFT. Protein docking using continuum electrostatics and geometric fit. *Protein Eng.* 2001; 14:105–113. [PubMed: 11297668]
17. Duhovny, D.; Nussinov, R.; Wolfson, HJ. WABI '02: Proceedings of the Second International Workshop on Algorithms in Bioinformatics. Springer-Verlag; London, UK: 2002. Efficient Unbound Docking of Rigid Molecules; p. 185-200.
18. Schneidman-Duhovny D, Inbar Y, Nussinov R, Wolfson HJ. Geometry-based flexible and symmetric protein docking. *Proteins: Struct Func Bioinfo.* 2005; 60:224–231.
19. Ritchie D, Kemp G. Protein Docking Using Spherical Polar Fourier Correlations. *Proteins: Struct Func Genet.* 2000; 39:178–194.
20. Shentu Z, Hasan M, Bystroff C, Zaki M. Context shapes: Efficient complementary shape matching for protein-protein docking. *Proteins: Struct Func Bioinfo.* 2008; 70:1056–1073.
21. Venkatraman V, Yang Y, Sael L, Kihara D. Protein-protein docking using region-based 3D Zernike descriptors. *BMC Bioinformatics.* 2009; 10:407. [PubMed: 20003235]
22. Bock, ME.; Cortelazzo, GM.; Ferrari, C.; Guerra, C. Identifying Similar Surface Patches on Proteins Using a Spin-Image Surface Representation. In: Apostolico, A.; Crochemore, M.; Park, K., editors. *Combinatorial Pattern Matching*, volume 3537 of *Lecture Notes in Computer Science*. Springer; Berlin/Heidelberg: 2005. p. 417-428.
23. Canterakis, N. 3D Zernike moments and Zernike affine invariants for 3D image analysis and recognition. *Proc. 11th Scandinavian Conf. Image Anal.*; 1999. p. 85-93.
24. Hwang H, Pierce B, Mintseris J, Janin J, Weng Z. Protein-protein docking benchmark version 3.0. *Proteins: Struct Func Bioinfo.* 2008; 73:705–709.
25. Cipriano GGP Jr, Gleicher M. Multi-Scale surface descriptors. *IEEE Trans Vis Comput Graph.* 2009; 15:1201–1208. [PubMed: 19834190]
26. Beygelzimer, A.; Kakade, S.; Langford, J. Cover trees for nearest neighbor. *Proceedings of the 23rd international conference on Machine learning, ICML '06*; New York, NY, USA: ACM; 2006. p. 97-104.
27. Mintseris J, Wiehe K, Pierce B, Anderson R, Chen R, Janin J, Weng Z. Protein-protein docking benchmark 2.0: An update. *Proteins: Struct Func Bioinfo.* 2005; 60:214–216.
28. dms program. <http://www.cgl.ucsf.edu/chimera/docs/UsersGuide/midas/dms1.html>
29. Connolly M. Analytical molecular surface calculation. *J Appl Cryst.* 1983; 16:548–558.
30. Gribskov M, Robinson N. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Comput Chem.* 1996; 20:25–33. [PubMed: 16718863]
31. Pierce B, Hourai Y, Weng Z. ZDOCK 2.3.1 and ZDOCK 3.0.1: Using a New 3D Convolution Library to Enhance Docking Efficiency. submitted.
32. Bock ME, Garutti C, Guerra C. Discovery of similar regions on protein surfaces. *Journal of Computational Biology.* 2007; 14:285–299. [PubMed: 17563312]
33. Francis-Lyon P, Gu S, Hass J, Amenta N, Koehl P. Sampling the conformation of protein surface residues for flexible protein docking. *BMC Bioinformatics.* 2010; 11:575–588. [PubMed: 21092317]
34. CUDA, Supercomputing for the masses: Part 1. <http://www.drdobbs.com/high-performance-computing/207200659>
35. Gabdoulhine R, Wade R, Walther D. MolSurfer: a macromolecular interface navigator. *Nucl Acids Res.* 2003; 31:3349–3351. [PubMed: 12824324]

Appendix A: Proof of correctness for cover tree with modified Manhattan distance

In this section, we show that the cover tree data structure for nearest neighbor queries is correct using our modified Manhattan distance for surface-histogram matching.

Given a set S of n points and a query point q , the nearest neighbor query problem is to efficiently find a point $p \in S$ that minimizes the distance $d(p, q)$. A cover tree preprocesses the set of n points and stores them in a tree. Each node of the tree is associated with one of the input points. A point appears at most once at any level. In the following, we identify a node using its associated point. Let C_i be the set of points associated with all nodes at level i (i increases as we traverse the tree from root to leaves). A cover tree satisfies three properties:

1. (Nesting) $C_i \subset C_{i+1}$. This implies that once a point $p \in S$ appears in C_i , then every lower level in the tree also has a node associated with p .
2. (Covering) For every $p \in C_{i+1}$, there exists a $q \in C_i$ such that $d(p, q) < 2^{-i}$ and the node at level i associated with q is a parent of the node at level $i + 1$ associated with p .
3. (Separation) For all distinct $p, q \in C_i$, $d(p, q) > 2^{-i}$.

The query algorithm for finding the nearest neighbor starts at the root level and iteratively travels down the tree. At each level i , the algorithm finds a set Q_{i+1} of points at level $i + 1$ to search next. The set Q_{i+1} satisfies two conditions:

1. Each node v in Q_{i+1} must be a child of Q_i .
2. Let w be the node associated with the point closest to query point q among all the children of Q_i . Each node v in Q_{i+1} must satisfy Eq. (A.1)

$$d(q, v) \leq d(q, w) + 2^{-i} \quad (\text{A.1})$$

At root level, $Q_{\text{root}} = C_{\text{root}}$. When the algorithm reaches the leaf level, it selects the closest point in Q_{leaf} to q and reports it. More details of the algorithm and the proof of its correctness can be found in ²⁶. Note that, in contrast to our representation of the cover tree, i decreases as the tree is descended in this published proof.

To show the correctness of the cover tree for answering nearest neighbor queries using modified Manhattan distance, we first assume Eq. (A.2) holds for all k, v' , where k is a child node of v' . We show that this is sufficient to prove the correctness of the query algorithm above. Then we prove that Eq. (A.2) does indeed hold for our modified Manhattan distance.

$$d(q, v') \leq d(q, k) + d(v', k) \quad (\text{A.2})$$

A node is selected for Q_{i+1} , if and only if it satisfies Eq. (A.1). Let v' be a node that does not satisfy Eq. (A.1), that is $d(q, v') > d(q, w) + 2^{-i}$. From this and Eq. (A.2), we obtain $d(q, w) + 2^{-i} < d(q, v') \leq d(q, k) + d(v', k)$. In Eq. (A.2), k is a child node of v' . By the covering property, $d(v', k) \leq 2^{-i-1}$. This implies $d(q, w) < d(q, k)$, that is by not searching the children of v' , we do not miss the nearest neighbor of q .

Next we show Eq. (A.2) holds for our modified Manhattan distance measure. For cover tree building, we use the usual Manhattan distance. Since the usual Manhattan distance is a metric, and Eq. (A.2) is the triangle inequality, Eq. (A.2) holds. For cover tree querying, the distance of two surface-histograms is defined as the sum of the distances of corresponding bins. If Eq. (A.2) is satisfied for all bins, it is satisfied for the overall distance. We use Eq. (A.3) for the distance computation between corresponding bins. This is a special case of the Manhattan distance where we punish surface overlaps more than surface gaps:

$$d(a_{ij}, \overline{b_{ij}}) = d(\overline{b_{ij}}, a_{ij}) = \begin{cases} (a_{ij} - \overline{b_{ij}}) & \text{if } a_{ij} \geq \overline{b_{ij}} \quad (\text{overlap}) \\ c(\overline{b_{ij}} - a_{ij}) & \text{where } 0 < c < 1, \text{ if } a_{ij} < \overline{b_{ij}} \quad (\text{gap}) \end{cases} \quad (\text{A.3})$$

To prove that our distance measure Eq. (A.3) satisfies Eq. (A.4), we consider four cases. Recall, we build a cover tree for all surface-histogram pairs on Q (see Methods) where the vertex normals are reversed. Let $\overline{b_1}, \overline{b_2}$ and a be the corresponding bins of three surface-histograms, where $\overline{b_1}$ and $\overline{b_2}$ are from two surface-histograms in a cover tree, and a is from the query surface-histogram.

$$d(a, \overline{b_1}) \leq d(a, \overline{b_2}) + d(\overline{b_1}, \overline{b_2}) \quad (\text{A.4})$$

Case 1): $a \geq \overline{b_1}$ and $a \geq \overline{b_2}$: then $(a - \overline{b_1}) \leq (a - \overline{b_2}) + |\overline{b_1} - \overline{b_2}|$. This is the Manhattan distance, so the inequality holds.

Case 2): $a < \overline{b_1}$ and $a \geq \overline{b_2}$: then $c(\overline{b_1} - a) \leq (a - \overline{b_2}) + |\overline{b_1} - \overline{b_2}|$. Since $0 < c < 1$, the inequality holds.

Case 3): $a \geq \overline{b_1}$ and $a < \overline{b_2}$: then $\overline{b_1} < a < \overline{b_2}$.

$$\begin{aligned} (a - \overline{b_1}) &\leq c(\overline{b_2} - a) + (\overline{b_2} - \overline{b_1}) \\ (a - \overline{b_1}) - c\overline{b_2} + ca - \overline{b_2} + \overline{b_1} &\leq 0 \\ (1+c)a - (1+c)\overline{b_2} &\leq 0 \\ a &\leq \overline{b_2} \end{aligned} \quad (\text{A.5})$$

$a \leq \overline{b_2}$ is true, so the inequality holds.

Case 4): $a < \overline{b_1}$ and $a < \overline{b_2}$: then $c(\overline{b_1} - a) \leq c(\overline{b_2} - a) + |\overline{b_1} - \overline{b_2}|$. Since $c(\overline{b_1} - a) \leq c(\overline{b_2} - a) + c|\overline{b_1} - \overline{b_2}|$ and $0 < c < 1$, the inequality holds.

Appendix B: Comparison of CS, ZDOCK, LZerD, PatchDock and shDock for bound docking

1. The 84 test cases in Benchmark 2.0

See Table (B.1).

2. 40 additional test cases in Benchmark 3.0

See Table (B.2).

Appendix C: Comparison of CS, LZero, ZDOCK (v3.0.1), PatchDock and shDock on the unbound test cases in the protein-protein docking benchmark

See Table (C.1).

Figure 1. Surface-histogram construction

Top: Given two vertices i and j on a mesh surface, we compute the local coordinate frame: its origin is the midpoint between i and j , the axis vector $\vec{x}$ is a unit vector along the oriented line (i, j) pointing to j , the $\vec{y}$ axis is a unit vector along the direction given by the average of the normal vectors at i and j , and $\vec{z}$ completes $\vec{x}$ and $\vec{y}$ such that $(\vec{x}, \vec{y}, \vec{z})$ is an orthonormal basis. **Bottom:** A 3D volume is centered at the frame origin. The edges of the volume are parallel to the frame axes. This volume captures the local geometry of the surface of the protein in the neighborhood of i and j .

Figure 2. Surface-histogram compression

The 3D volume can be compressed along the y-axis direction. **Top**, the 3D volume before compression. Grey cells mark the surface in the volume. **Bottom**, the 2D image where the grey scale indicates the height of the histogram. The result is a 2D image where each bin is a histogram that describes the thickness of the corresponding column inside the protein.

Figure 3. Effectiveness of surface-histograms in detecting complementarity

For interface pairs on protein P of the 1AKJ complex we compute the performance of the surface-histograms to detect true neighboring pairs on protein Q , and report the results as ROC areas, with large values indicating good performance. Among 109 interface pairs, we find that 69 pairs have a ROC area of 0.8 and above and 32 pairs have a ROC area between 0.5 and 0.8. The average ROC area over all pairs is 0.816. Note that a method producing random outcomes would lead to a ROC area of 0.5.

Figure 4. Our best prediction (in terms of RMSD) for all test cases in ZDOCK Benchmark 3.0 The RMSDs for bound docking (blue squares) and for unbound docking (magenta dots) are plotted. To improve clarity, the two series are arranged in ascending order of their best RMSDs. (The sorting was done independently for bound and unbound docking; the x-axis therefore refers to the sorted test case number, which may correspond to two different proteins).

Figure 5. Distribution of the ranks of the first prediction within 2.5 Å of the native complex structure for all test cases in ZDOCK Benchmark 3.0

We regroup the test cases into 7 categories: those with a best rank of 1 ($x=1$), no more than 5 ($x=2$), 10 ($x=3$), 100 ($x=4$), 1000 ($x=5$), 2000 ($x=6$), and 3600 ($x=7$). The corresponding distributions are shown in cyan bars for bound docking and orange bars for unbound docking.

Figure 6. Problems in predicting the complex 1AK4

1AK4 corresponds to the complex formed by human cyclophilin A (magenta) with the N-terminal domain of HIV-1 capsid (green). shDOCK fails to find a near-native conformation (one with an RMSD better than 2.5 Å) in its top 3600 candidates for the bound docking test case; the best prediction among these candidates has an RMSD of 3.53 Å (blue). We observe that the contact surface area between the two proteins is small and that there is a “hole” at the interface.

Figure 7.
Contact ratio vs. RMSD of the best prediction (bound docking).

Figure 8.
The error of unbound docking prediction is correlated with the interface conformation change.

Figure 9. Interface visualization for 1AKJ bound docking

(a): the interface for the ground truth bound case. **(b):** the interface for the 7th predicted complex with an RMSD of 0.68 Å. **(c), (d) and (e):** the interfaces for the top three predicted complexes in bound docking arranged from left to right. The interfaces are visualized using MolSurfer³⁵, where red, white and blue represent overlap, close and gap respectively. We observe that our top predicted configurations all have larger contact area as well as larger white area, indicating better geometric fit, than the bound configuration and the near-native configuration.

Figure 10. Contact score vs. RMSD for bound 1AKJ docking

The single-circled dot and the double-circled dot represent the best ranked hit and the number one prediction respectively. The best ranked hit for 1AKJ bound docking is ranked the 7th with an RMSD of 0.68 Å. We observe that the approximated contact area is larger for the top ranked false positives than for the near-native configuration.

Figure 11. Contact score vs. RMSD for unbound 1AKJ docking

The single-circled dot and the double-circled dot represent the best ranked hit and the number one prediction respectively. The best ranked hit for 1AKJ unbound docking is ranked the 292th with an RMSD of 2.23 Å. We observe that the approximated contact area is larger for the top ranked false positives than for the near-native configuration.

Table 1

The number of near-native configurations and the total configurations after surface-histogram matching for the unbound test case 1AKJ.

	# Total cnfg. ^a	# Near-native cnfg.	Ratio to random ^b
Bound	88,430,065	1583	9.9
Unbound	25,982,245	213	4.5
Random sampling	25,982,245	47	1

^aThe total number of configurations is different between the bound and the unbound, because we use 0.8 and 1.0 mesh resolutions respectively.

^bThe ratio is the percentage of near-native configurations for each type to the percentage of near-native configurations in the random sampling.

Table 2

Number of cases with at least one hit in bound docking experiments.

	CS	ZDOCK (PSC)	PatchDock	LZerD	shDock
Rank=1 ^a	16	6	13	16	23
Rank≤5	30	11	15	20	37
Rank≤10	38	18	17	20	41
Rank≤100	53	37	28	39	57
Rank≤1000	64	48	39	56	67
Rank≤2000	69	54	41	58	69
Rank≤3600	71	55	42	60	74
Avg. RMSD (Å) ^b	1.64	1.73	1.53	1.17	0.69

^a For all 84 bound test cases in ZDOCK Benchmark 2.0, we report the number of cases with at least one hit found with our method (shDock) and four competing methods at different ranking levels, where a rank relates to the ordering provided by the scoring function of the method considered.

^b For each case that succeeds in generating a hit in the top 3600 candidates, we take the RMSD of the best ranked hit. For all of these, we compute the average RMSD (see text for details).

Table 3

Pair-wise comparison of shDock with four standard shape-based docking algorithms on bound docking test cases.

	CS	ZDOCK (PSC) ^b	PatchDock	LZerD
shDock ranks better ^a	40	59	55	48
same rank	14	6	9	10
shDock ranks worse	25	11	12	19
Neither finds a hit	5	8	8	7

^aFor all 84 bound test cases in ZDOCK Benchmark 2.0, we compare the rank of the best-ranked hit found by shDock with the rank of the best-ranked hit found by each of the four other docking algorithms considered.

^bPSC refers to ZDOCK version 2.1.

Table 4

Number of cases with at least one hit in unbound docking experiments.

	Benchmark 2.0 (84 cases)		
	CS	LZerD	shDock
Rank $\leq$ 1000 ^a	12	29	22
Rank $\leq$ 2000	15	36	33
Rank $\leq$ 3600	18	38	41
Avg. RMSD (Å) ^b	2.03	1.87	1.89

^a For all 84 unbound test cases in ZDOCK Benchmark 2.0, we report the number of cases with at least one hit found with our method (shDock) and two competing methods at different ranking levels, where the rank indicates the ordering provided by the scoring function of the method considered.

^b For each case that succeeds in generating to generate at least one hit in the top 3600 candidates, we take the RMSD of the best ranked hit. For all of these, we compute the average RMSD.

Table 5

Pair-wise comparison of shDock with four standard shape-based docking algorithms on unbound docking test cases.

	Benchmark 2.0 (84 cases)		Benchmark 3.0 (124 cases)	
	CS	LZerD	ZDOCK	PatchDock
shDock ranks better ^a	30	20	15	32
same rank	0	0	0	0
shDock ranks worse	13	29	74	34
Neither finds a hit	41	35	35	58

^aFor all unbound test cases in ZDOCK Benchmark 2.0 and in ZDOCK Benchmark 3.0, we compare the rank of the best-ranked hit found by shDock with the rank of the best-ranked hit found by CS and LZerD, and by ZDCOK (version 3.0.1) and PatchDock, respectively.

Table 6

Number of cases with at least one hit in the unbound docking experiments in ZDOCK Benchmark 3.0.

	Benchmark 3.0 (124 cases)		
	ZDOCK (v3.0.1)	PatchDock	shDock
Rank $\leq$ 1000	68	32	30
Rank $\leq$ 2000	76	43	44
Rank $\leq$ 3600	84	52	54
Avg. RMSD ($\text{\AA}$)	1.92	1.79	1.87
# of configurations $\leq$ 2.5 $\text{\AA}$ ^a	619	140	207

^aThe total number of configurations generated with an RMSD of 2.5 $\text{\AA}$ or less for the top 3600 candidates for all cases in Benchmark 3.0.

Table B.1
Comparison of CS, ZDOCK (PSC), PatchDock, LZerD and shDock on all bound cases of Benchmark 2.0 ²⁷

We obtain the best-ranked hit and its associated RMSD for each case.

Complex	CS		ZDOCK (PSC)		PatchDock		LZerD		shDock	
	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD
1A2K	40	1.08	570	2.41	300	1.47	826	1.2	1	0.70
1ACB	8	2.32	6	0.82	10	1.6	15	1.5	140	1.31
1AHW	7	1.2	56	1.18	40	1.55	1	1.21	43	1.06
1AK4	2925	2.08	3471	1.14	-	-	1024	2.18	-	-
1AKJ	265	2.15	448	1.88	-	-	-	-	7	0.68
1ATN	49	2.1	558	1.15	-	-	-	-	11	0.43
1AVX	10	1.76	1	1.96	43	2.14	812	1.8	1	0.73
1AY7	193	1.23	46	1.68	24	2.07	46	1.31	11	0.56
1B6C	11	1.78	24	1.69	40	1.92	44	0.94	88	0.96
1BGX	1	1.96	-	-	-	-	-	-	-	-
1BJ1	1	1.05	3	1.42	-	-	58	0.73	4	0.55
1BUH	61	1.55	393	1.43	83	1.14	2378	0.82	130	0.40
1BVK	45	1.69	1087	1.43	131	2.12	-	-	259	0.45
1BVN	1	1.55	10	1.24	1	0.75	1	0.71	1	0.52
1CGI	1	1.37	1	1.12	1	1.08	1	0.61	1	0.41
1D6R	4	1.68	35	1.04	-	-	27	0.52	13	1.48
1DE4	13	1.21	452	1.62	-	-	-	-	4	0.32
1DQJ	67	1.65	19	2	83	1.71	391	0.67	1	0.36
1E6E	1	1.58	58	2.06	2	2.29	11	1.45	23	0.56
1E6J	1337	1.92	699	2.02	1706	1.43	104	0.62	46	0.52
1E96	1206	1.84	-	-	1767	1.44	-	-	87	0.68
1EAW	1	1.41	1	1.75	1	0.99	2	0.81	1	0.75
1EER	1	1.62	-	-	1	1.66	38	1.4	1	0.40
1EWY	518	2.26	-	-	139	1.42	141	0.86	-	-
1EZU	1	1.6	-	-	1	0.94	1	1.18	3	0.35
1F34	1	1.99	-	-	1	1.9	1	1.32	1	0.79
1F51	7	2.01	-	-	1	1.92	26	1.58	26	1.51

Complex	CS		ZDOCK (PSC)		PatchDock		LZerD		shDock	
	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD
1FAK	1997	1.7	-	-	-	-	-	-	1	0.59
1FC2	7	1.85	55	2.18	49	1.24	42	0.52	3	0.94
1FQ1	-	-	-	-	-	-	-	-	2	0.70
1FQJ	12	1.94	120	1.94	248	1.48	15	1.78	1	0.85
1FSK	9	2.06	19	1.7	218	1.57	454	0.94	1	0.52
1GCQ	2	1.26	382	1.81	-	-	119	0.44	1	0.44
1GP2	53	1.86	-	-	-	-	-	-	1	0.34
1GRN	1	1.84	7	2.26	3	1.45	1	1.08	1	0.22
1H1V	14	2.37	1510	2.4	-	-	-	-	818	1.19
1HE1	1	1.44	7	1.67	1	1.06	15	0.81	1	0.30
1H1A	2	1.07	1	1.7	14	1.19	3	1.5	3	0.71
1I2M	6	1.36	14	1.8	-	-	1	1.48	95	0.92
1I4D	104	1.42	793	2.08	167	1.05	695	1.68	38	0.57
1I9R	-	-	1271	2.04	-	-	174	2.11	3319	0.60
1IB1	2	1.48	-	-	-	-	5	2.04	1	0.57
1IBR	1	2.05	-	-	-	-	336	1.34	1	0.79
1IJK	-	-	-	-	-	-	115	0.9	2958	1.04
1IQD	14	1.19	55	1.83	-	-	1	0.35	28	0.37
1JPS	2	1.26	23	2.3	96	1.87	21	0.98	2	0.65
1K4C	5	0.88	30	1.16	337	1.53	274	0.88	5	0.26
1K5D	2	2.06	10	2.11	-	-	2245	2.03	1	0.75
1KAC	-	-	381	1.52	-	-	-	-	534	0.52
1KKL	226	1.67	-	-	-	-	946	0.91	26	0.78
1KLU	1108	1.8	-	-	-	-	-	-	178	0.54
1KTZ	2280	1.41	-	-	-	-	-	-	603	0.51
1KXP	3	2.17	-	-	-	-	-	-	1	0.97
1KXQ	229	1.51	30	1.6	29	1.63	47	0.83	4	0.21
1M10	-	-	33	2.23	-	-	-	-	33	0.63
1MAH	1	1.45	1	1.91	1	1.27	1	0.98	1	0.64
1ML0	569	1.91	75	1.94	7	0.58	157	1.48	52	0.41

Complex	CS		ZDOCK (PSC)		PatchDock		LZerD		shDock	
	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD
1MLC	30	1.15	1205	1.37	516	1.79	1378	2.49	10	0.67
1N2C	3	1.36	-	-	-	-	-	-	2	0.52
1NCA	3	1.77	20	1.48	-	-	354	1.21	8	0.46
1NSN	-	-	-	-	-	-	36	0.82	3305	2.20
1PPE	1	2.32	2	1.21	1	1.03	1	0.41	1	0.43
1QA9	972	1.3	-	-	-	-	-	-	-	-
1QFW	1247	2.21	16	2.46	-	-	14	2.1	1036	0.58
1RLB	311	1.63	-	-	3143	2.32	561	0.49	2021	2.22
1TMQ	1	2.32	8	1.79	1	1.52	1	0.65	4	0.69
1UDI	3	1.52	1	1.5	1	1.97	1	0.8	1	0.27
1VFB	8	1.5	-	-	-	-	37	1.74	1369	0.27
1WEJ	496	1.25	1120	1.11	-	-	707	2.4	202	0.55
1WQ1	1	1.14	4	2.04	1	0.84	1	1.77	-	-
2BTF	4	1.13	21	1.21	137	1.82	40	0.94	4	0.90
2JEL	56	1.4	532	1.77	282	1.65	1	0.79	10	0.31
2MTA	21	1.45	1447	2.26	115	1.71	92	0.88	81	0.66
2PCC	-	-	-	-	-	-	-	-	788	1.89
2QFW	38	2.13	54	1.84	-	-	1	1.02	986	0.61
2SIC	4	1.36	9	1.19	-	-	1	0.88	1	0.68
2SNI	2	1.27	4	2.5	13	2.1	4	1.27	2	0.51
2VIS	-	-	-	-	-	-	-	-	3471	0.59
7CEI	123	1.9	5	2.18	-	-	54	1.08	3	0.86

“-” indicates that the algorithm finds no hit in the top 3600 candidates. Bold font indicates the best rank among all methods. Cases for which no methods find a hit are excluded.

Table B.2

Our bound results on the 40 additional cases of Benchmark 3.0

We show the best-ranked hit and its associated RMSD for each case.

Complex	Rank	RMSD	Complex	Rank	RMSD	Complex	Rank	RMSD
1AZS	84	0.32	1R0R	1	0.55	2C0L	15	0.50
1BKD	1	0.60	1R8S	1	0.13	2CFH	1	0.37
1E4K	242	0.58	1S1Q	38	0.56	2FD6	300	0.40
1EFN	1	0.66	1T6B	47	0.38	2H7V	341	0.80
1GLA	-	-	1XD3	1	0.63	2HLE	2	0.25
1GPW	3	0.58	1XQS	1	0.26	2HQS	1	0.82
1IRA	1	0.86	1Y64	-	-	2HRK	7	0.49
1J2J	7	0.36	1YVB	83	0.37	2I25	40	0.89
1JMO	1	0.33	1Z0K	1	0.34	2NZ8	1	0.42
1K74	124	1.28	1Z5Y	25	0.83	2O8V	-	-
1N8O	1	0.55	1ZHI	506	0.87	2O0B	160	0.62
1NW9	1	0.62	2AJF	-	-	2OT3	1	0.44
1OPH	33	0.54	2B42	1	0.51	2UUY	2	0.46
1PXV	1	0.36						

“-” indicates that the algorithm finds no hit in the top 3600 candidates.

Table C.1
Comparison of CS, LZerD, ZDOCK (v3.0.1), PatchDock and shDock on the unbound cases
 We obtain the best-ranked hit and its associated RMSD for each case.

Complex	CS			LZerD			shDock			ZDOCK (v3.0.1)			PatchDock		
	Rank	RMSD		Rank	RMSD		Rank	RMSD		Rank	RMSD		Rank	RMSD	
1A2K	-	-	-	-	-	-	237	2.45	-	900	2.45	-	-	-	-
1AHW	402	2.46	5	1.34	1419	2.05	1304	2.21	168	1.3					
1AK4	-	-	-	-	-	-	-	-	2195	2.01	-	-	-	-	-
1AKJ	-	-	-	-	-	-	292	2.23	356	1.99	-	-	-	-	-
1AVX	-	-	786	2.41	1199	2.5	22	1.85	2053	2.22					
1AY7	-	-	1884	1.98	733	1.56	946	2.05	679	1.14					
1AZS	NA	NA	NA	NA	1794	2.26	6	1.25	-	-	-	-	-	-	-
1B6C	-	-	1001	2.41	-	-	17	2.26	201	2.14					
1BJ1	1893	1.93	298	1.86	2052	1.58	1	1.35	-	-	-	-	-	-	-
1BUH	-	-	-	-	391	1.78	45	1.92	625	2.37					
1BVK	-	-	-	-	-	-	253	1.89	-	-	-	-	-	-	-
1BVN	34	2.41	27	2.32	82	2.15	2	1.85	110	1.68					
1CGI	-	-	-	-	-	-	148	2.32	-	-	-	-	-	-	-
1D6R	-	-	2619	2.24	-	-	-	-	-	-	-	-	-	-	-
1DFJ	-	-	-	-	-	-	1	2.11	-	-	-	-	-	-	-
1DQJ	-	-	-	-	-	-	2432	2.29	-	-	-	-	-	-	-
1E6E	-	-	52	2.13	1014	1.52	15	2.1	38	2.12					
1E6J	-	-	439	2.18	3065	2.49	8	2.39	3483	2.29					
1E96	-	-	216	2.14	3526	2.5	444	2.19	-	-	-	-	-	-	-
1EAW	94	2.29	20	2.42	324	2.07	39	1.51	59	2.1					
1EFN	NA	NA	NA	NA	-	-	2619	1.99	-	-	-	-	-	-	-
1EWY	-	-	349	2.36	175	2.15	11	2.49	88	2.46					
1EZU	-	-	824	1.21	784	2.24	1173	2.19	-	-	-	-	-	-	-
1F34	-	-	-	-	1528	2.14	668	1.48	30	1.57					
1F51	-	-	3545	1.58	3561	2.12	19	2.1	650	2.03					
1FQJ	-	-	-	-	-	-	-	-	3004	2.46					
1FSK	20	1.57	15	2.4	47	0.62	1	0.76	420	2.08					

Complex	CS		LZerD		shDock		ZDOCK (v3.0.1)		PatchDock	
	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD
1GCQ	-	-	-	-	1787	2.21	3123	2.35	-	-
1GLA	NA	NA	NA	NA	-	-	1455	2.39	-	-
1GP2	-	-	-	-	-	-	26	2.44	-	-
1GPW	NA	NA	NA	NA	117	1.14	72	2.5	1	2.02
1GRN	-	-	1407	2.18	1724	1.61	1055	1.98	831	1.54
1HE1	1029	2.17	267	1.98	3107	1.41	1344	1.94	33	2.16
1HE8	-	-	-	-	646	2.27	-	-	-	-
1I2M	-	-	-	-	-	-	44	2.4	-	-
1I4D	-	-	-	-	-	-	669	2.23	-	-
1I9R	-	-	95	2.39	302	2.48	-	-	-	-
1IJK	-	-	-	-	1639	2.42	446	0.98	-	-
1IQD	-	-	41	1.2	-	-	2	2.31	3228	2.12
1I2J	NA	NA	NA	NA	211	2.14	-	-	248	1.6
1IPS	-	-	292	0.9	469	1.56	1141	0.87	1185	1.89
1K4C	-	-	1188	1.43	-	-	2364	1.69	-	-
1K5D	-	-	-	-	-	-	2055	2.44	-	-
1K74	NA	NA	NA	NA	33	1.35	5	1.17	48	1.3
1KAC	-	-	655	2.18	138	2.38	1538	1.97	-	-
1KXP	-	-	-	-	3	1.7	1	1.83	37	2.49
1KXQ	2226	1.73	73	1.68	30	1.41	1	0.95	11	1.5
1MAH	597	1.16	92	0.87	2252	2.13	1	1.19	1184	0.83
1ML0	-	-	559	2.38	303	1.87	1	2.32	450	1.59
1MLC	18	2.28	1834	1.16	1027	0.96	87	1.23	847	0.98
1N8O	NA	NA	NA	NA	-	-	2	1.7	1501	1.27
1NCA	26	1.79	-	-	600	0.85	188	0.76	575	1.45
1NSN	-	-	945	2.29	1364	2.03	2646	1.96	1254	1.76
1PPE	2	2.31	1	0.83	8	1.86	1	0.88	12	1.51
1QA9	-	-	1381	2.19	1264	2.16	-	-	3039	1.86
1QFW	597	1.73	108	1.24	759	1.08	326	1.19	1457	1.85
1R0R	NA	NA	NA	NA	870	1.62	407	2.09	290	1.74

Complex	CS		LZerD		shDock		ZDOCK (v3.0.1)		PatchDock	
	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD
1RLB	-	-	-	-	-	-	5	2.47	-	-
1SIQ	NA	NA	NA	NA	-	-	479	2.28	341	1.45
1T6B	NA	NA	NA	NA	-	-	58	2.45	-	-
1TMQ	783	1.68	50	1.45	186	1.18	11	1.83	3	1.16
1UDI	2469	2.14	59	2.36	-	-	60	1.92	261	1.55
1VFB	228	2.46	1303	1.69	207	2.23	1240	2.45	1541	2.48
1WEJ	-	-	-	-	-	-	155	0.71	2152	1.25
1WQ1	-	-	141	1.87	-	-	15	2.16	-	-
1XD3	NA	NA	NA	NA	-	-	19	2.26	2999	2.41
1XQS	NA	NA	NA	NA	-	-	26	2.46	-	-
1YVB	NA	NA	NA	NA	235	2.43	2	1.86	38	2.09
1Z5Y	NA	NA	NA	NA	-	-	64	2.03	-	-
1ZHI	NA	NA	NA	NA	-	-	11	1.98	-	-
2AJF	NA	NA	NA	NA	-	-	685	2.36	-	-
2B42	NA	NA	NA	NA	-	-	1	0.9	31	1.62
2BTF	-	-	-	-	-	-	177	2.07	-	-
2CFH	NA	NA	NA	NA	75	1.75	2	1.81	2	1.86
2FD6	NA	NA	NA	NA	1553	1.51	1	1.47	2517	1.66
2H7V	NA	NA	NA	NA	-	-	7	2.01	-	-
2HLE	NA	NA	NA	NA	620	2.32	186	2.28	70	2.37
2HMI	-	-	-	-	-	-	237	2.5	-	-
2HQS	NA	NA	NA	NA	60	1.74	2046	2.22	1045	2.28
2HRK	NA	NA	NA	NA	1386	2.45	425	2.47	-	-
2I25	NA	NA	NA	NA	-	-	4	2.05	1393	2.49
2JEL	-	-	133	2.49	-	-	5	1.69	1142	0.95
2MTA	-	-	606	1.64	2423	2.11	170	2.38	1086	0.83
2OOB	NA	NA	NA	NA	2267	1.1	-	-	-	-
2PCC	-	-	-	-	-	-	944	2.12	-	-
2QFW	33	2.32	68	1.55	-	-	12	2.08	1018	1.68
2SIC	1077	2.28	12	2.04	35	1.94	4	0.53	113	1.24

Complex	CS		LZerD		shDock		ZDOCK (v3.0.1)		PatchDock	
	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD	Rank	RMSD
2SNI	-	-	-	-	-	-	440	1.66	-	-
2UTY	NA	NA	NA	NA	2025	1.75	526	2.15	2046	1.83
2VIS	-	-	-	-	3027	1.45	638	2.49	-	-
7CEI	2290	1.9	-	-	1515	2.05	1	2.07	241	2.49

“.” indicates that the algorithm finds no hit in the top 3600 candidates. “NA” indicates the case is new in Benchmark 3.0 and it was not present in Benchmark 2.0. Cases where no method finds a hit are excluded.