Protein-protein recognition using F2Dock

Muhibur Rasheed

January 23, 2011

Contents

Chapter 1

Introduction

1.1 Protein Docking/Virtual Screening

Several biological processes are mediated through proteins interacting and forming complexes. For example, the formation of multi-protein complexes is central to the rubric of molecular machines such as viral capsids, ribosomes, ion-channels. Protein-protein complexes are also critical to the immune targeting of infectious proteins, the specificity of enzymatic catalysis, and the precise hormonal control and physiological effect of organs. Atomistically detailed models of proteins and their computer simulations have become powerful assistive tools to wet lab experiments, for interpreting these complex interactions. The simulations of protein-protein interactions yield important clues for developing therapeutic interventions related to disease, such as cancer and metabolic disorders. In protein-protein docking, we seek to determine the best relative transformation and conformation of two proteins (e.g., a target and potential drug) that results in a stable complex reproducible in nature (if one exists).

1.2 Software suite

Performing and analyzing protein-protein docking experiments can be performing using a suite of software developed in the CVC: F^{2}Dock, *GB-rerank*, *MolSurf*, *TeχMol*, and the F^{2}Dock server.

F2Dock F^{2}Dock is the main molecular docking application. It efficiently scores conformations in the space of relative rotations and translations between two proteins and returns a sorted list of the top results.

GB-rerank *GB-rerank* is a post-process for the F^{2}Dock program for re-ranking the top results based on a (more computationally intensive) energy model including the Generalized Born solvation term.

MolSurf *MolSurf* is a suite tools for generating and manipulating molecular surfaces. Constructing molecular surfaces is an important preprocess to molecular docking since some terms in the energy model depend upon the molecular surface rather than just the atomic positions.

TexMol *TeχMol* is a graphical tool for visualizing protein models. It also contains an intuitive graphic interface to F2Dock and allows for immediate visualization of the results.

F2Dock Server The F^{2}Dock server is a program which passes information between the graphical *TeχMol* interface and the other software tools. This allows to the graphical client and computationally intensive tools to be run on separate computers (or for the latter, clusters of computers).

Chapter 2

Overview of F2Dock+GB-rerank

F2Dock uses an approximation of the *binding free energy* of the docked complexes to rank them. An empirical model of the total *free energy* of a system with explicit solvent interactions, can be modeled as-

$$E = \underbrace{E_{\mathrm{MM}} + E_{\mathrm{sol}}}_{\text{potential energy}} - TS,$$

where E_{MM} is the classical molecular mechanical energy of the solute, E_{sol} is the solvation energy, T is the system temperature, and S is the solute entropy.

The molecular mechanical energy is defined as follows [28].

$$E_{\mathrm{MM}} = \underbrace{E_d + E_\theta + E_\varphi}_{\text{bonded interactions}} + \underbrace{E_{\mathrm{vdw}} + E_{\mathrm{coul}}}_{\text{nonbonded interactions}}$$

The first three terms represent bonded interactions: covalent bonds (E_d), valence bonds (E_θ), and torsions around bounds (E_φ). The last two terms represent nonbonded interactions: Lennard-Jones potential for van der Waals forces (E_{vdw}), and the Coulomb potential for electrostatics (E_{coul}).

The solvation energy E_{sol} consists of the energy to form cavity in the solvent (E_{cav}), the solute-solvent van der Waals interaction energy ($E_{\mathrm{vdw(s\text{-}s)}}$), and the electrostatic potential energy change due to the solvation (also known as the polarization energy, E_{pol}) [15, 18, 20, 37, 38].

$$E_{\mathrm{sol}} = \underbrace{E_{\mathrm{cav}} + E_{\mathrm{vdw(s\text{-}s)}}}_{\text{nonploar}} + \underbrace{E_{\mathrm{pol}}}_{\text{polar}}$$

Now the binding free energy of a complex formed by two given molecules A and B is given by:

$$\Delta E = E_{A+B} - (E_A + E_B),$$

where E_A, E_B and E_{A+B} are the total free energy of molecule A, molecule B and the complex $A+B$, respectively. Now assuming that the solvent temperature T and the solute entropy S are constants, and the two molecules are mostly rigid (i.e., negligible conformational change upon binding), we have,

$$\Delta E \approx \Delta E_{\mathrm{vdw}} + \Delta E_{\mathrm{coul}} + \Delta E_{\mathrm{sol}}$$

Hence, for rigid-body docking ΔE has three major components.

- ΔE_{vdw} : The *Lennard-Jones* 12-6 dispersion-repulsion potential due to van der Waals forces is given by $\sum_{i\in A, j\in B}\left(\frac{a_{ij}}{r_{ij}^{12}} - \frac{b_{ij}}{r_{ij}^{6}}\right)$, where r_{ij} is the distance between two given atoms, and a_{ij} and b_{ij} are constants based on atom types. *Shape or curvature complementarity* along the docked interface is a related measure, and is considered one of the primary measures of docking quality.
- ΔE_{coul} : The long-range *electrostatic potential* is given by $\sum_{i\in A, j\in B}\frac{q_i q_j}{\epsilon(r_{ij})r_{ij}}$, where q_i and q_j are Coulombic charges, and $\epsilon(r_{ij})$ is a distance dependant dielectric constant. Inter-molecular hydrogen bonds and disulphide bonds are responsible for short-range electrostatic interactions.

- $\Delta E_{\rm sol}$: *Desolvation free energy* is defined as the change in energy due to the displacement of solvent molecules from the interface. For rigid molceules the non-polar part of $\Delta E_{\rm sol}$ can be approximated with a quantity proportional to the change in solvent accessible surface area upon binding. Existance of large hydrophobic area at the interface is often considered a favorable condition for docking. The polar part of $\Delta E_{\rm sol}$ measures how the electrostatic interaction between the solvent and the solute changes due to the change in the induced polarization in the solvent as a result of complex formation. This part can be approximated using *Generalized Born* (GB) theory [39].

In [9] we described a Non-equispaced Fast Fourier Transform (NFFT) based rigid-body protein-protein docking algorithm for efficiently performing the initial docking search (based on shape and electrostatics complementarity). In [5] we extended our NFFT-based docking algorithm to *F^2Dock* (F^2 = F̲ast F̲ourier) which included an adaptive search phase (both translational and rotational) for faster execution.

Finally, the latest version of *F^2Dock* [12] includes improved shape-complementarity and electrostatics functions as well as a new on-the-fly affinity function based on interface propensity and hydrophobicity. The current version uses uniform FFT, but exploits the sparsity of FFT grids for faster execution, and also restricts its search within a narrow band around the larger molecule. It also includes a clustering phase for penalizing docking poses that are structurally very similar to poses with better scores, and a set of efficient on-the-fly filters[1] based on Lennard-Jones potential, steric clashes, interface propensity, interface area, residue-residue contact preferences, antibody active sites, and glycine richness at the interface for enzymes. The filters are implemented using fast multipole type recursive spatial decomposition techniques [13]. A solvation energy based reranking program called *GB-rerank* has also been implemented which uses an approximation scheme, and so can be tuned to obtain any suitable speed-accuracy trade-off. Both *F^2Dock* and *GB-rerank* have been implemented as multithreaded programs for faster execution on multicore machines. Our molecular visualization software *TeχMol* serves as a front-end to *F^2Dock* in a client-server mode of execution. *F^2Dock* has been calibrated based on an extensive experimental study on the rigid-body complexes from ZDock Benchmark 2.0 [31].

2.1 Overview

Let A and B are two proteins with M_A and M_B atoms respectively and without loss of generality we assume that $M_A \geq M_B$, i.e., A is the larger of the two proteins.

Figure 2.1 gives a very high level overview of our docking algorithm which consists of two separate phases:

- Phase I (Input Processing using *MolSurf*): From equations to , it is clear that to evaluate the scoring terms for docking, one needs different accurate representations of the molecules. For example, electrostatics and VdW potentials are computed based on atom centers, radii and charges only. In other words, a primary structure representation is enough. However, for the shape complementarity and solvation energy computations, a quality smooth (SES) surface representation. Moreover, as will be further explained when we discuss the solvation energy based reranking, a discrete approximation of the polarization energy requires the sampling of Gaussian integration points on the smooth molecular surface.

- Phase II (Docking Search with *F^2Dock*): We perform exhaustive 6D search in discretized rotational and translational space. During this search molecule A is kept *stationary* while molecule B is assumed to be *moving* in the sense that relative transformations are applied to its coordinates. The rotational space is discretized using uniformly sampled Euler angles while the translational space is sampled on a uniform 3D grid. For each sampled rotation we first rotate molecule B around its geometric center, and then score all docking poses involving A and the rotated B where their relative translations are sampled from a uniform 3D grid. The scoring function includes shape complementarity, electrostatics and interface propensity, and scores for all grid points are computed simultaneously using FFT (Fast Fourier Transform) based 3D convolution.

[1]for penalizing potential false positives

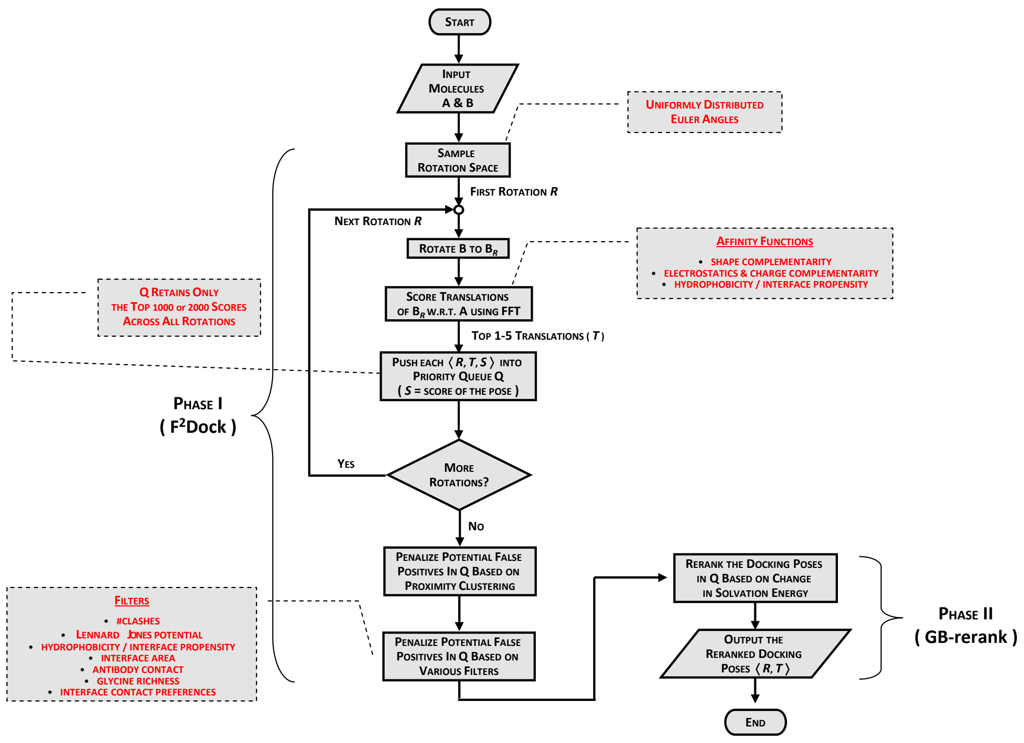

Start
Input Molecules A & B
Uniformly Distributed Euler Angles
Sample Rotation Space
First Rotation R
Next Rotation R
Rotate B to B_R
Affinity Functions
• shape complementarity
• electrostatics & charge complementarity
• hydrophobicity / interface propensity
Q Retains Only the Top 1000 or 2000 Scores Across All Rotations
Score Translations of B_R w.r.t. A using FFT
Top 1-5 Translations (T)
Push each ⟨ R, T, S ⟩ into Priority Queue Q (S = score of the pose)
Phase I (F²Dock)
Yes
More Rotations?
No
Penalize Potential False Positives In Q Based on Proximity Clustering
Filters
• #clashes
• Lennard Jones potential
• hydrophobicity / interface propensity
• interface area
• antibody contact
• glycine richness
• interface contact preferences
Penalize Potential False Positives In Q Based on Various Filters
Rerank the Docking Poses in Q Based on Change in Solvation Energy
Output the Reranked Docking Poses ⟨ R, T ⟩
Phase II (GB-rerank)
End

Figure 2.1: High level overview of rigid-body protein-protein docking using F^2*Dock* and *GB-rerank*.

The top several thousand poses from this FFT-based scoring phase are then examined for structural similarity[2], and a docking pose is penalized provided a structurally very similar docking pose with a better score already exists.

The resulting list is then passed through several filters in order to eliminate/penalize potential false positives. Results are filtered based on Lennard-Jones potential, number of steric clashes, interface area, interace propensity, residue-residue contact preferences, antibody active sites, and frequency of glycine residues at the interface for enzymes. Filters are implemented using fast multipole-type octree-based hierarchical spatial decomposition techniques [13] with a view to evaluating any given docking pose more accurately than FFT-based scoring but in a reasonable time so that they can still be used on-the-fly during docking.

- Phase III (Solvation Energy Based Reranking with *GB-rerank*): The ranked docking poses obtained from phase I are rescored and reranked based on the change in solvation energy caused by each pose. The polar part of the solvation energy is approximated using the surface-based formulation of Generalized Born (GB) energy [6], and implemented using a fast octree-based approximation scheme we describe in detail in [13]. Among the non-polar parts the dispersion energy is also approximated using octrees while the cavity forming is approximated by computing an approximate interface area of the two molecules using our fast linear-space *Dynamic Packing Grid* (DPG) data structure described in [4].

[2]based on how close the geometric centers of B are when A for the poses are superimposed

2.2 Input Preparation

In this section we briefly describe the different files types and how they are generated. Details of the file formats, the generation algorithm and software usage are presented in Chapter 4.

2.2.1 Primary structure/atomic representation

The primary structure representation, and the only input, used by *F^2Dock*is the PDB files of the receptor and the ligand. PDB files are the most widely used format to represent proteins, and these are easily and freely available. However, the PDB files sometimes are missing some atoms, specially Hydrogens, and often contains extra crysllazied water. So, the files are curated first.

The curated PDBs contains the atom centers, atom types, the residue, chain and secondary structure hierarchy etc. But it does not mention the radii and charges. We use PDB2PQR to generate a PQR file, which uses different force fields to compute the raii and charges. See Section 4.1.2 for details.

2.2.2 Molecular surface representation

Then we use our inhouse software *MolSurf*to generate a smooth molecular surface, improve the mesh quality and approximate the normals at the vertices. *MolSurf*provides many ways of generating and representing the SES and SAS (for example, sum of Gaussians, or a mesh with normals, or a signed distance function in a grid can be used). See Section 4.2 for details.

2.2.3 Quadrature points sampling and skin/core generation

*MolSurf*also provides interfaces for sampling the Gaussian quadrature points on the surface (details in Section 4.3.2). Finally, for the shape complementarity analysis, *F^2Dock*needs to identify the exposed and internal atoms of the ligand, and grow a layer of pseudoatoms as a skin for the receptor. Both these computations are also supported by *MolSurf*(details in Section 4.3.1).

2.3 Search

Pairwise protein-protein docking can be viewed as a search in the 6D relative translational (3D) and rotational (3D) space of the two proteins involved for a stable (energetically most favorable) complex. However, a more efficient and possibly adaptive sampling of this 6D search space is required to keep the scoring computations feasible. F2Dock splits the search space into a rotational and a translational part, i.e the space is $SO^3 \times R^3$. The following sections details how the spaces are subsampled.

2.3.1 Rotational Sampling.

The rotation space is sampled using uniformly distributed Euler angles as in [33, 30, 11]. Table 2.1 shows the different sampling intervals supported by *F^2Dock* along with the approximate size of the sample set in each case. The rotation samples were obtained from [32].

Sampling Interval	20°	15°	12°	10°	8°	6°	4°
Number of Samples (N_R)	1,900	4,400	8,600	14,900	29,000	68,800	232,000

Table 2.1: Rotational sampling based on uniformly distributed Euler angles [30, 11].

2.3.2 Exhaustive Translational Search.

As mentioned before, the translational search is performed on a 3D grid, and the various *affinity functions* used for the search are defined in such a way that FFT-based fast 3D convolution algorithms can be used for evaluating all grid points.

Figure 2.2 gives a high level overview of FFT-based scoring of the translational grid which involves two forward (forward FFT for the receptor is performed only once) and one inverse FFT computations.

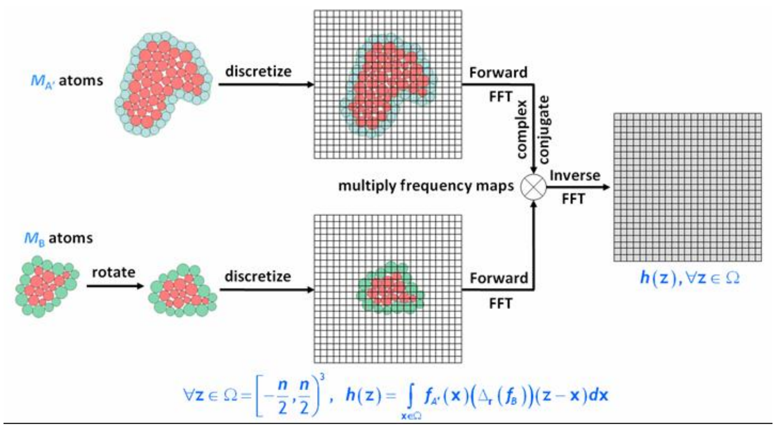


Figure 2.2: Fast Fourier Transform (FFT) based convolution for scoring of discrete translational space.

Current version of F^2Dock uses uniform FFT but exploits the sparsity of the input and the output grids for faster computation.

2.4 Score

A brief overview of the affinity functions used by F^2Dock (phase I) for FFT-based exhaustive translational search is as follows. Please refer to [12] for details.

2.4.1 Shape Complementarity

This is an improved version of the traditional double-skin layer based shape complementarity function [26]. Unlike the traditional model the receptor skin atoms do not touch the receptor surface, the weight assigned to a receptor core atom is a function of its depth from the surface, and the weight assigned to a receptor skin atom is a function of surface curvature. The position and thickness of the skin layers are carefully chosen for accuracy of predictions via the benchmark training set.

Let A' denote molecule A with its grown skin layer. If the weight assigned to an atom/pseudo-atom k (of molecule $P \in \{A', B\}$) is $c_k^{SC} = c_{k,Re}^{SC} + i \cdot c_{k,Im}^{SC}$, then the affinity function for shape complementarity is:

$$f_P^{SC}(\mathbf{x}) = \sum_{k \in P} \left(\sqrt{w_{ss}} \cdot c_{k,Re}^{SC} + i \cdot \sqrt{w_{cc}} \cdot c_{k,Im}^{SC}\right) \cdot g_{P,k}^{SC}(\mathbf{x}),$$

where, $g_{P,k}^{SC}(\mathbf{x}) = e^{-\beta\left(\frac{(\mathbf{x}-\mathbf{c}_k)^2}{r_k^2} - 1\right)}$ with blobbiness $\beta = 2.3$.

The overall shape complementarity score for translation $\mathbf{t}$ and rotation $\mathbf{r}$ is:

$$H_{A,B}^{SC}(\mathbf{t}, \mathbf{r}) = Re\left(F_{A,B}^{SC}(\mathbf{t}, \mathbf{r})\right) + \frac{w_{sc}}{\sqrt{w_{ss} \cdot w_{cc}}} \cdot Im\left(F_{A,B}^{SC}(\mathbf{t}, \mathbf{r})\right),$$

where, $F_{A,B}^{SC}(\mathbf{t}, \mathbf{r}) = \int_{\mathbf{x}} f_A^{SC}(\mathbf{x}) T_{\mathbf{t}}\left(\Delta_{\mathbf{r}}\left(f_B^{SC}(\mathbf{x})\right)\right)$ with T and Δ being the translation and the rotation operator, respectively.

For further details about the rasterization and weight assignment, refer to [12].

2.4.2 Electrostatics:

This is based on the approximate Coulombic interaction function introduced by Gabb et al. [17], but designed to reduce discretization errors on the grid. In the current version of F^2Dock, we define for $P \in A, B$,

$$g^E_{P,k}(\mathbf{x}) = e^{-\beta\left(\frac{(\mathbf{x}-\mathbf{c}_k)^2}{\gamma^2}-1\right)},$$

where, $\gamma > 0$ is a constant, and blobbiness $\beta = 2.3$. In our experiments the new definition of $g^E_{P,k}(\mathbf{x})$ with $\gamma = 3.4\mathring{A}$ performed much better than the original Gabb et al. formulation. Smoothing with the Gaussian function as above has the effect of reduced discretization error on the grid.

The overall electrostatics score for translation $\mathbf{t}$ and rotation $\mathbf{r}$ is:

$$H^E_{A,B}(\mathbf{t},\mathbf{r}) = w_E \cdot Re\left(F^E_{A,B}(\mathbf{t},\mathbf{r})\right),$$

where, $F^E_{A,B}(\mathbf{t},\mathbf{r}) = \int_{\mathbf{x}} f^E_A(\mathbf{x}) T_{\mathbf{t}}\left(\Delta_{\mathbf{r}}\left(f^E_B(\mathbf{x})\right)\right)$, and w_E is the user-specified weight given to electrostatics interaction.

2.4.3 Interface Propensity:

The interfaces between the two molecules are scored based on statistical information on the relative frequencies (henceforth called, interface protepsity) of different residues in the interfaces of protein complexes [23, 24]. Let $IP(R)$ denote the natural logarithm of the interface propensity value of a residue R. Suppose $A + B_{\mathbf{t},\mathbf{r}}$ is a docking pose obtained by rotating molecule B by $\mathbf{r}$ and translating by $\mathbf{t}$. Let $iAtom^+_{\mathbf{t},\mathbf{r}}(P)$ and $iAtom^-_{\mathbf{t},\mathbf{r}}(P)$ denote the set of atoms in the interface of $P \in \{A, B\}$ in this docking pose that have positive and negative IP values, respectively. Also let $iAtom^s_{\mathbf{t},\mathbf{r}}(A,B) = iAtom^s_{\mathbf{t},\mathbf{r}}(A) + iAtom^s_{\mathbf{t},\mathbf{r}}(B)$, for $s \in \{+,-\}$. Then we assign the following interface propensity score to the docking pose:

$$IP\text{-}score_{\mathbf{t},\mathbf{r}}(A,B) = -\frac{\sum_{a \in iAtom^+_{\mathbf{t},\mathbf{r}}(A,B)} IP(a)}{\min\left(IP_\epsilon, \sum_{a \in iAtom^-_{\mathbf{t},\mathbf{r}}(A,B)} IP(a)\right)},$$

where $IP_\epsilon = \max_{IP(R)<0} IP(R)$.We also penalize for very small interfaces by setting $IP\text{-}score$ to a negative value when $|iAtom^+_{\mathbf{t},\mathbf{r}}(A,B)|$ is below a user-defined threshold.

2.5 Filter

The functions of the various filters used by F^2Dock are as follows. Again, the details can be found in [12].

2.5.1 Lennard-Jones Filter:

Penalize if the Lennard-Jones potential of a docking pose is above some user-defined threshold. We approximate the Lennard-Jones (LJ) potential between molecules A and $B_{\mathbf{t},\mathbf{r}}$ given by the following expression.

$$LJ(A, B_{\mathbf{t},\mathbf{r}}) = \sum_{i \in A, j \in B_{\mathbf{t},\mathbf{r}}} \left(\frac{a_{ij}}{r^{12}_{ij}} - \frac{b_{ij}}{r^6_{ij}}\right),$$

where r_{ij} is the distance between atoms $i \in A$ and $j \in B_{\mathbf{t},\mathbf{r}}$, constants a_{ij} and b_{ij} depend on the type (e.g., C, H, O, etc.) of the two atoms involved. For any fixed pair of atom types a_{ij} and b_{ij} are fixed, and are calculated from the Amber force field.

Observe that direct computation of $LJ(A,B)$ requires $\mathcal{O}(M_A M_B)$ time, where M_A (resp. M_B) is the number of atoms in molecule A (resp. B). However, since the terms in the summation diminish quickly with the increase of r_{ij}, one can evaluate $LJ_{\delta^-}(A,B) = \sum_{i \in A, j \in B, r_{ij} \leq \delta} lj(i,j)$ as an approximation of $LJ(A,B)$, where δ is a given distance cutoff.

Suppose $M_A > M_B$. Then one can evaluate $LJ_{\delta^-}(A,B)$ in $\mathcal{O}((M_A + M_B)\log M_B + m)$ time and $\mathcal{O}(M_A + M_B)$ space, where m is the total number of $\langle i \in A, j \in B\rangle$ pairs within distance δ. The trick is to use an octree [21] to store the atoms of B, and then use it to locate the atoms of B within distance δ from each atom of A (see [13] for details). Use of a 3D grid instead of an octree may result in $\Theta\left({M_B}^3\right)$ space usage in the worst case [13].

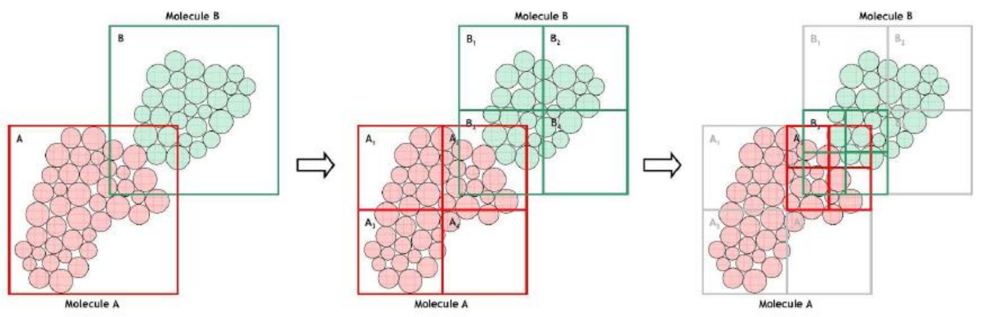


Figure 2.3: Approximation of LJ potential in 2D using quadtrees [16] (i.e., 2D variant of octrees): In the leftmost figure the bounding box of molecule A (resp. B) represents the root node of the quadtree storing A (resp. B). The smallest boxes in the middle and the rightmost figures represent quadtree nodes at levels 2 (i.e., children of the root) and 3, respectively. Let us assume for simplicity that if two nodes of the two quadtrees do not intersect they are far enough so that the LJ potential between their atoms can be approximated by treating them as pseudo-atoms. In the leftmost figure the two root nodes (nodes A and B) intersect, and so we move to their children nodes in the middle figure. In the middle figure only nodes A_2 and B_3 intersect, and so while the potential between the atoms of all other $\langle A_i, B_j \rangle$ pairs can be approximated, we need to move to the children of A_2 and B_3 in order to compute the potential between them (see the rightmost figure).

2.5.2 Fast $(1+\epsilon)$-Approximation of $LJ(A, B)$

Observe that $LJ(A,B) = LJ_{\delta^-}(A,B) + LJ_{\delta^+}(A,B)$, where $LJ_{\delta^+}(A,B) = \sum_{\ i \in A,\, j \in B,\, r_{ij} > \delta} \ lj(i,j)$. We outline below how to obtain an error-bounded approximation of $LJ(A,B)$ through a fast approximation of $LJ_{\delta^+}(A,B)$ in addition to the exact evaluation of $LJ_{\delta^-}(A,B)$. More precisely, given any user-defined constant $\epsilon > 0$, we will approximate $LJ(A,B)$ to within a $(1+\epsilon)$ factor of its exact value.

In the expression of $LJ(A,B)$, a_{ij} and b_{ij} are fixed for any fixed pair of atom types, and can be calculated from the Amber force field using well depths μ_{XY} and equivalence contact distances of homogeneous pairs $r_{eqm,XY}$, where $X = atomType(i \in A)$ and $Y = atomType(j \in B)$) [42, 34]. By definition, $a_{ij}/b_{ij} = r^6_{eqm,XY}/2$ (see [34]). We assume $X, Y \in \{\text{C, H, N, O, P, S}\}$.

Let $\mathcal{M}_X$ denote the subset of atoms of type X in molecule $\mathcal{M} \in \{A, B\}$. Then $LJ(A,B) = \sum_{X,Y \in \{\text{C, H, N, O, P, S}\}} LJ(A_X, B_Y)$, where, $LJ(A_X, B_Y) = LJ_{\delta^-_{XY}}(A_X, B_Y) + LJ_{\delta^+_{XY}}(A_X, B_Y)$, for some constant $\delta_{XY} \geq 0$ (to be defined later). We outline below how to approximate $LJ(A_X, B_Y)$ for a given pair $\langle X, Y \rangle$. We evaluate $LJ_{\delta^-_{XY}}(A_X, B_Y)$ exactly, and approximate $LJ_{\delta^+_{XY}}(A_X, B_Y)$ to within a factor of $(1+\epsilon)$ of its exact value.

Let $\delta_{XY} \geq (1/2 + 1/\epsilon)^{1/6} r_{eqm,XY}$. If we approximate each b_{ij}/r^6_{ij} with $r_{ij} > \delta_{XY}$ to within a factor of $1 + \epsilon/(2+\epsilon)$, simple algebraic manipulations show $|lj(i,j)| < \left[b_{ij}/r^6_{ij}\right]_{approx} < (1+\epsilon)|lj(i,j)|$.

In order to approximate $LJ(A_X, B_Y)$ as mentioned above, we construct two octrees $\mathcal{T}_{A_X}$ and $\mathcal{T}_{B_Y}$ from the atoms in A_X and B_Y, respectively, and compute a $(1+\epsilon)$-approximation of $LJ(A_X, B_Y)$ by simultaneous recursive traversals of $\mathcal{T}_{A_X}$ and $\mathcal{T}_{B_Y}$ starting from their root nodes. Suppose at some point we are at node x of $\mathcal{T}_{A_X}$ and node y of $\mathcal{T}_{B_Y}$. If both x and y are leaf nodes, potential between the atoms contained in x (say, $\mathcal{M}_x$) and y (say, $\mathcal{M}_y$) is computed exactly. Otherwise if x and y are far enough (i.e., at least δ_{XY} apart), and small enough[3] the potential between $\mathcal{M}_x$ and $\mathcal{M}_y$ is approximated by assuming that x and y are single pseudo atoms centered at the center of gravity of $\mathcal{M}_x$ and $\mathcal{M}_y$, respectively, and taking $|\mathcal{M}_x||\mathcal{M}_y|\left(b_{ij}/r^6_{xy}\right)$ as the approximated potential, where r_{xy} is the distance between the centers of the two pseudo atoms. Otherwise we subdivide x and/or y (i.e., move to their children), and approximate the potential recursively. Figure 2.3 explains the approach in 2D. The pseudocode is given in Figure 2.4.

See [13] for a proof of the upper bound on the approximation time.

[3] i.e., $r_{x,y} + (r_x + r_y) < (1 + \epsilon/(2+\epsilon))^{\frac{1}{6}} (r_{x,y} - (r_x + r_y))$, where, r_x (resp. r_y) is the radius of the smallest ball centered at the atom centers of x (resp. y) that encloses all atom centers of x (resp. y).

```
ApproxLJ( $x$, $y$ )
(Inputs are two octree nodes $x \in \mathcal{T}_{A_X}$ and $y \in \mathcal{T}_{B_Y}$, and the the output is a floating point number $V$ such that $U \leq V \leq (1+\epsilon) \cdot U$,
where $U = \sum_{i \in \mathcal{M}_x \wedge j \in \mathcal{M}_y} \left( a_{ij}/r_{ij}^{12} - b_{ij}/r_{ij}^{6} \right)$. By CHILD$(x)$ (resp. CHILD$(y)$) we denote the set of non-empty octree nodes
obtained by subdividing node $x$ (resp. $y$). We denote by $b_{XY}$ the value of the constant $b_{ij}$ for atom types $X$ and $Y$, and by $r_{x,y}$
the distance between the centers of $x$ and $y$.)

1. if LEAF$(x) \wedge$ LEAF$(y)$ then return $\sum_{i \in \mathcal{M}_x \wedge j \in \mathcal{M}_y} \left( \frac{a_{ij}}{r_{ij}^{12}} - \frac{b_{ij}}{r_{ij}^{6}} \right)$        {exact value}
2. else if $r_{x,y} - (r_x + r_y) > \delta_{XY} \;\wedge\; \frac{r_{x,y} + (r_x + r_y)}{r_{x,y} - (r_x + r_y)} < \left( 1 + \frac{\epsilon}{2+\epsilon} \right)^{\frac{1}{6}}$ then return $-\frac{M_x \cdot M_y \cdot b_{XY}}{(r_{x,y} - (r_x + r_y))^6}$        {approximation}
3.     else if LEAF$(x)$ return $\sum_{cy \in \text{CHILD}(y)}$ ApproxLJ( $x$, $cy$ )        {recursive approximation}
4.         else if LEAF$(y)$ return $\sum_{cx \in \text{CHILD}(x)}$ ApproxLJ( $cx$, $y$ )        {recursive approximation}
5.             else return $\sum_{cx \in \text{CHILD}(x) \wedge cy \in \text{CHILD}(y)}$ ApproxLJ( $cx$, $cy$ )        {recursive approximation}

ApproxLJ Ends
```

Figure 2.4: Recursive approximation of $\sum_{i \in \mathcal{M}_x \wedge j \in \mathcal{M}_y} \left(a_{ij}/r_{ij}^{12} - b_{ij}/r_{ij}^{6} \right)$ to within a factor of $1+\epsilon$. The initial call is ApproxLJ(ROOT$(\mathcal{T}_{A_X})$, ROOT$(\mathcal{T}_{B_Y})$) for the approximation of $\sum_{i \in A_X \wedge j \in B_Y} \left(a_{ij}/r_{ij}^{12} - b_{ij}/r_{ij}^{6} \right)$.

2.5.3 Clash Filter:

Penalizes all docking poses with the number of steric (atom-atom) collisions above some user-defined threshold. Two atoms $a \in A$ and $b \in B$ with van der Waals radii r_a and r_b, respectively, are said to be in a *clash* provided the distance between their centers is smaller than $\alpha(r_a + r_b)$, where α is a user-defined positive constant. F^{2}Dock counts the total number of atomic clashes between molecules A and $B_{\mathbf{t},\mathbf{r}}$ as follows.

$$Clash_{\mathbf{t},\mathbf{r}}(A, B) = |\{b | (b \in B_{\mathbf{t},\mathbf{r}}) \wedge \exists_{a \in A} (r_{a,b} < \alpha(r_a + r_b))\}|$$

Searching for clashes is sped up using spatial decomposition based on Octrees to isolate neighboring octree nodes and then DPG is used to locally search for clashed within the neighboring octree nodes only. Details of the technique is quite similar to the Lennard JOnes filter described above and can be found in [12].

2.5.4 Interface Propensity Filter:

Using statistical information from [24] it computes a score for each pose which from a high level can be viewed as the ratio of the interface area of the pose corresponding to residues that typically appear in high frequencies in protein interfaces to the interface area corresponding to residues that appear in low frequencies. A docking pose is penalized if this ratio is below a user-defined threshold.

2.5.5 Residue-Residue Contact Filter:

Penalizes potential false positives based on statistical information on residue-residue contact preferences described in [19]. Given a docking pose $A + B_{\mathbf{t},\mathbf{r}}$, we identify all residue-residue contacts at the interface of the two molecules using a fast algorithm similar to the one in clash filter. Two residues are considered to be in contact if the distance between their C_β atoms (C_α for Gly) is less than 6 Å. See [12] for further details.

2.5.6 Antibody-Antigen Contact Filter:

This filter uses statistical information on antibody-antigen contact preferences derived in [29, 1]. It is based on the observation that in each antibody each of the following three regions will make at least one antigen contact[4]: (1) either CDR-L1 or CDR-H1, (2) CDR-L3, and (3) CDR-H3. An atom a is considered to be in *close neighborhood* of another atom b provided the distance between the centers of a and b is

[4]a *contact* is defined as burial by at least 1 Å^2 change in solvent accessibility

at most $2(r_a + r_b)$. F^{2}Dock computes three quantities: $N_{L1 \cup H1}$, N_{L3} and N_{H3}, denoting the number of antigen atoms that are in the close neighborhood of any atom in the antibody regions CDR-L1/CDR-H1, CDR-L3 and CDR-H3, respectively. F^{2}Dock penalizes a docking pose provided $N_{L1 \cup H1} < \mu_{L1 \cup H1}$ or $N_{L3} < \mu_{L3}$ or $N_{H3} < \mu_{H3}$.

2.5.7 Glycine Filter:

This filter exploits the observation that enzyme active sites are rich in glycines, particularly *G-X-Y* and *Y-X-G* oligopeptides, where X and Y are polar and non-polar residues, respectively, and G is glycine [43]. F^{2}Dock counts such motifs on the interface and penalizes a docking pose if this count is below a user-specified threshold, and can also reward poses with higher *G-X-Y*/ *Y-X-G* frequency at the interface. Similar to the clash filter, the motifs are identified using an efficient algorithm based on DPG and octree based spatial decomposition.

2.5.8 Interface Area Filter:

Penalizes a docking pose if the interface area is outside some user-defined range. Weighted quadrature/integration points from the surface of each molecule (see Section 4.3.2). Sum of the weights of all such integration points is equal to the molecular surface area up to quadrature approximation error, and each integration point can be treated as the center of a small surface patch with its weight being the patch area. Hence we approximate the interface area as a sum of the weights of all integration points of one molecule within a user-specified cutoff distance from any integration point on the other molecule. We use our DPG data structure to compute this sum in time linear to the total number of atoms in the two molecules.

2.6 Energy based reranking

The solvation energy E_{sol} consists of three parts:

$$E_{\text{sol}} = \underbrace{E_{\text{cav}} + E_{\text{vdw(s-s)}}}_{\text{nonploar}} + \underbrace{E_{\text{pol}}}_{\text{polar}}$$

The first two terms are often modeled as [15, 40]

$$E_{\text{cav}} = pV + \sum_i \gamma_i A_i \quad \text{and} \quad E_{\text{vdw(s-s)}} = \rho_0 \sum_i \int_{\text{ex}} u_i^{(\text{att})}(\mathbf{x}_i, \mathbf{r}) d^3\mathbf{r}$$

where p is the solvent pressure, V is the molecular volume, A_i is the solvent accessible surface area of atom i and γ_i is its solvation parameter, ρ_0 is the bulk density, and $u_i^{(\text{att})}$ is the van der Waals dispersive component of the interaction between atom i and the solvent.

The last term, E_{pol}, can be approximated using *Generalized Born* (GB) theory [39].

$$E_{\text{pol}} = -\frac{\tau}{2} \sum_{i,j} \frac{q_i q_j}{\sqrt{r_{ij}^2 + R_i R_j e^{-\frac{r_{ij}^2}{4R_i R_j}}}}, \tag{2.1}$$

where $\tau = 1 - \frac{1}{\epsilon}$, and R_i is the effective Born radius of atom i.

GB-rerank approximates each of these terms as described in the following sections, and reranks the list of top docking poses produced by F^{2}Dock based on the resulting ΔE_{sol} values. In order to approximate ΔE_{sol}, *GB-rerank* precomputes the E_{sol} values for molecules A and B, and then computes E_{sol} for each docking pose. See [12] for details.

2.7 Dynamic Packing Grid Data Structure

The Dynamic Packing Grid (DPG) is a neighborhood data structure for maintaining and manipulating flexible molecules. It can be used to efficiently maintain the molecular surface and to compute binding affinities and other molecular properties.

Assuming that the centers of two different balls in the collection cannot come arbitrarily close to each other, on a RAM with w-bit words, the *DPG* data structure can report all balls intersecting a given ball or within $\mathcal{O}(r_{max})$ distance from a given point in $\mathcal{O}(\log\log w)$ time with high probability (w.h.p.), where r_{max} is the radius of the largest ball in the collection. It can also answer whether a given ball is exposed (i.e., lies on the union boundary) or buried within the same time bound. At any time the entire union boundary can be extracted from the data structure in $\mathcal{O}(m)$ time in the worst-case, where m is the number of atoms on the boundary.

2.7.1 Description (Layout) of the Packing Grid Data Structure

DPG represents the entire 3-space as a $2r_{max}$-grid, and maintain the non-empty grid-planes, grid-lines and grid-cells. Note that a grid component (i.e., cell, line or plane) is non-empty if it contains the center of at least one ball in M. The data structure can be described as a tree with 5 levels: 4 internal levels (levels 3, 2, 1 and 0) and an external level of leaves (see Figure 2.5). The description of each level follows (further details are available in ([2])).

The Leaf Level "Ball" Data Structure (DPG$_{-1}$). The data structure stores the center $c = (c_x, c_y, c_z)$ and the radius r of the given ball B. And other application specific details. For example, for surface maintenance, we compute the spherical arrangement on the surface of the ball and store the patches, for solvation energy computations, we store the Born radius etc.

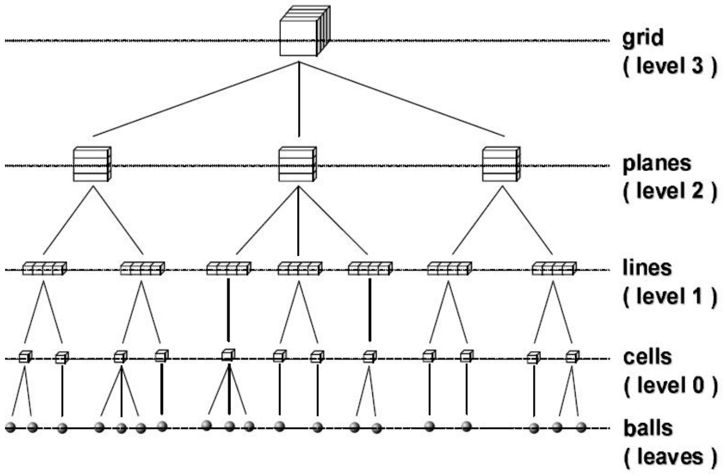


Figure 2.5: Hierarchical structure of DPG

The Level 0 "Grid-Cell" Data Structure (DPG$_0$). A grid-cell maintains a list of pointers to data structures of the $\mathcal{O}(1)$ balls whose centers lie inside the cell. Since we create "grid-cell" data structures only for non-empty grid-cells, there will be at most n (and possibly $\ll n$) such data structures.

The Level 1 "Grid-Line" Data Structure (DPG$_1$). We create a "grid-line" data structure for a (b, c)-line provided it contains at least one non-empty grid-cell. Each (a, b, c)-cell lying on this line is identified with the unique integer a, and the identifier of each such non-empty grid-cell is stored in an integer range search data structure RR described by Mortensen et al. ([35]). We augment RR to store the pointer to the corresponding "grid-cell" data structure with each identifier it stores.

The Level 2 "Grid-Plane" Data Structure (DPG$_2$). A "grid-plane" data structure is created for a c-plane provided it contains at least one non-empty grid-line. Similar to the "grid-line" data structure it identifies each non-empty (b, c)-line lying on the c-plane with the unique integer b, and stores the identifiers in a range reporting data structure RR.

The Level 3 "Grid" Data Structure (DPG$_3$). This data structure maintains the non-empty grid-planes in an integer range reporting data structure RR in a similar way where each c-plane is identified by the unique integer c.

2.7.2 Supported query/update

Let M be a collection of n balls with radii $r_1, \ldots, r_n$ and centers at $c_1, \ldots, c_n$. Let $r_{max} = \max_i \{r_i\}$ and let $d_{min} = \min_{i,j} \{d(c_i, c_j)\}$, where $d(c_i, c_j)$ is the Euclidean distance between c_i and c_j. DPG supports the following queries and functions (the time complexities are mentioned in Table 2.2)-

Queries

1. **Intersect**(c, r)**:** Returns all balls intersecting $B = (c, r)$.
2. **Range**(p, δ)**:** Returns all balls with centers within distance δ of point p. We assume that $\delta = O(r_{max})$.
3. **Exposed**(c, r)**:** Returns *true* if the ball $B = (c, r) \in M$ contributes to the boundary of the union of the balls in M.
4. **Surface**( )**:** Returns the outer boundary of the union of the balls in M. If there are multiple disjoint outer boundaries defined by M, the routine returns one of them.
5. **Add**(c, r)**:** Adds a new ball $B = (c, r)$ to the set M.
6. **Remove**(c, r)**:** Removes the ball $B = (c, r)$ from M.
7. **Move**(c_1, c_2, r)**:** Moves the ball with center c_1 and radius r to a new center c_2.

	Time Complexity (w.h.p.)	
Operations	Assuming $t_q = \mathcal{O}(\log\log w)$ $t_u = \mathcal{O}(\log w)$	Assuming $t_q = \mathcal{O}(\log\log n)$ $t_u = \mathcal{O}\left(\frac{\log n}{\log\log n}\right)$
Range, Intersect, Exposed	$\mathcal{O}(\log\log w)$	$\mathcal{O}(\log\log n)$
Add, Remove, Move	$\mathcal{O}(\log w)$	$\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$
Surface	$\mathcal{O}$ (#balls on surface) (worst-case)	
Assumptions: (i) RAM with w-bit Words, (ii) Collection of n Balls, (iii) $\delta = \mathcal{O}(r_{max})$ and, (iv) $r_{max} = \mathcal{O}$ (min. dist. between two balls)		

Table 2.2: Time complexities of the operations supported by the packing grid data structure.

See [4] for details of each update/query, surface maintenance and energetics computations.

Chapter 3

Software Implementation

3.1 F2Dock

The implementation of F2Dock follows a simple pattern. Each part of the docking pipeline discussed in Sections 2.3 to 2.5 have been implemented as separate dynamically linked libraries. A single controller class is also implemented which uses these libraries and follows the pathway shown in the flowchart in Figure 2.1.

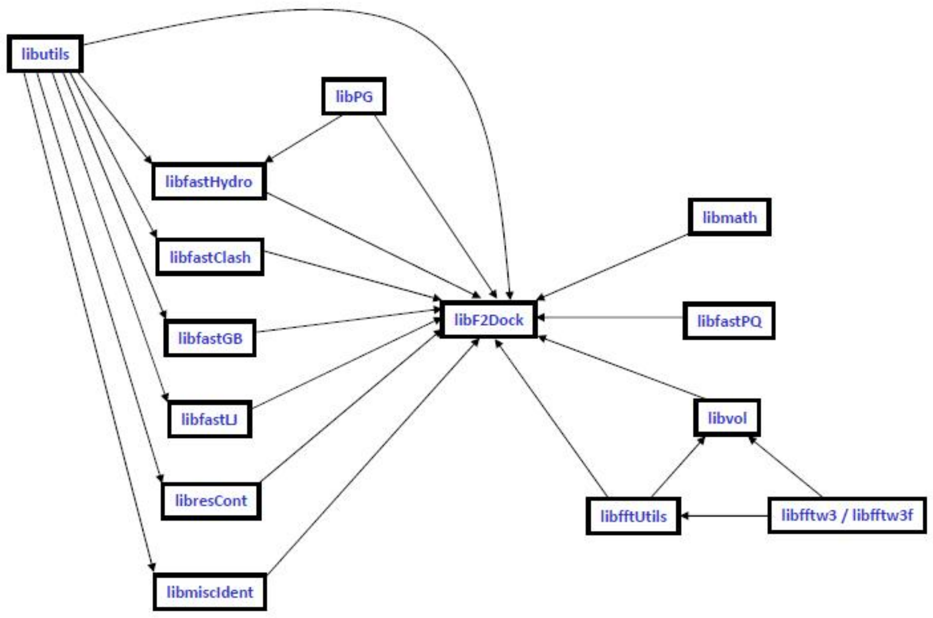


Figure 3.1: Dependency diagram of F2Dock. The overall pipeline is controlled by the library named libF2Dock. Individual components are implemented in the other corresponding libraries.

In Figure 3.1 we show the major libraries of F2Dock. A brief overview of the functions of these libraries are presented here-

3.1.1 libF2Dock

This library parses the parameters values passed to it as a file (see Section 5.1.1 for details), and based on the selected values, it decides which parts of the pipeline are to be eecuted. This library also parses the input files including pqr, pdb and f2d files (see Chapter 4) and populates internal data structures. After the pipeline is completed, this library also has the job of preparing and storing the output file.

3.1.2 libfftUtils, libfftw3, libvol

These libraries acts as wrappers to enable the libF2Dock to use the FFTW3 software package by preparing the FFTW related parameters and grid-values. These libraries also support inverse-FFT as well as fast multiplication routines for the FFT grids. In other words, the shape-complementarity and electrostatics scoring are carried out with the functions implmented in these libraries.

3.1.3 libDPG

This library implements the Dynamic Packing Grids data strucure for molecular surface maintenance and nearest neighbor queries. The library is heavily used in all the filtering computation including Lennard-Jones filter, Clash Filter, Hydrophobicity filter, Residue Contact filter as well as the hydrophobicity, solvation energy based scoring terms.

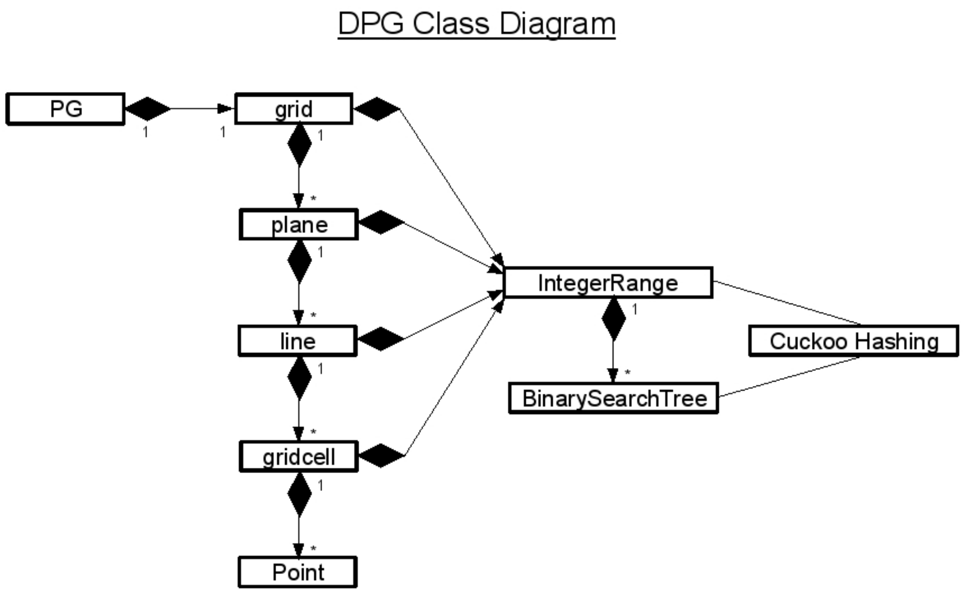


Figure 3.2: The class diagram for DPG

Figure 3.2 shows the class diagram of DPG. Note that it is possible to inherit the Point class to maintain application specific information. Also, this implementation builds on binary search trees, dynamic perfect hashing, and y-fast trees. However, instead of dynamic perfect hashing we used "cuckoo hashing" ([36]).

3.1.4 libfastClash

This library computes the number of clashes between two proteins in a given pose. Two atoms $a \in A$ and $b \in B$ with van der Waals radii r_a and r_b, respectively, are said to be in a *clash* provided the distance between their centers is smaller than $\alpha(r_a + r_b)$, where α is a user-defined positive constant. Clashes are identified by first applying hierarchical subdivision (oct-tree) to identify neighbor nodes of the tree and then libDPG is used locally to detect atom-atom clash.

3.1.5 libfastLJ

This library approximates the Lennard-Jones (LJ) potential between molecules A and $B_{\mathbf{t},\mathbf{r}}$ given by the following expression.

$$LJ(A, B_{\mathbf{t},\mathbf{r}}) = \sum_{i \in A, j \in B_{\mathbf{t},\mathbf{r}}} \left(\frac{a_{ij}}{r_{ij}^{12}} - \frac{b_{ij}}{r_{ij}^{6}} \right),$$

where r_{ij} is the distance between atoms $i \in A$ and $j \in B_{\mathbf{t},\mathbf{r}}$, constants a_{ij} and b_{ij} depend on the type (e.g., C, H, O, etc.) of the two atoms involved. A positive value indicates a electrostatic clash in a docked pose. However for unbound-unbound docking, it allows soft filtering to allow some clashes.

Similar to Clash filter, this also uses Oct-tree and libDPG to compute the clashes optimally in $O(n)$ time instead of $O(n^2)$.

3.1.6 libfastHydro

This library implements fast interface propensity computations optimized using spatial decomposition and nearest neighbor queries to detect residue-residue contacts on the interface. Details can be found in [12].

3.1.7 libfastGB

This library computes the difference of polarization energy between the proteins in isolation and in a docked pose. E_{pol} is approximated using *Generalized Born* (GB) theory [39].

$$E_{\text{pol}} = -\frac{\tau}{2}\sum_{i,j}\frac{q_i q_j}{\sqrt{r_{ij}^2 + R_i R_j e^{-\frac{r_{ij}^2}{4R_i R_j}}}}, \tag{3.1}$$

where $\tau = 1 - \frac{1}{\epsilon}$, and R_i is the effective Born radius of atom i given by Equation (3.2).

$$\frac{1}{R_i^3} = \frac{3}{4\pi}\int_{\text{ex}}\frac{1}{|\mathbf{r} - \mathbf{x}_i|^6}d^3\mathbf{r}, \quad i \in [1, M]. \tag{3.2}$$

However, we approximate the Born radii from the discrete approximation by applying the divergence theorem and Gaussian quadrature on Equation (3.2).

$$\frac{1}{{R_i}^3} \approx \frac{1}{4\pi}\sum_{k=1}^{N} w_k \frac{(\mathbf{r}_k - \mathbf{x}_i)\cdot\vec{\mathbf{n}}_k}{|\mathbf{r}_k - \mathbf{x}_i|^6}. \tag{3.3}$$

First all Born radii of the molecule/complex are computed using a fast approximation scheme described in [13]. A fast algorithm for estimating E_{pol} [13] from the approximated Born radii is also implemented.

3.1.8 libresCont

This filter uses statistical information on residue-residue contact preferences described in [19] in order to penalize potential false positives. These contact preferences were derived from a nonredundant set of 621 protein-protein interfaces of known high resolution structures. The implementation is similar to that of the interface propensity (hydrophobicity) filter.

3.1.9 libutils, libfastPQ, libmath

These libraries contain utility routines for basic mathematical operations, string comparisons and parsing, memory management etc.

3.2 GB-rerank

GB-rerank reranks a list of candidate docking poses based on approximate solvation free energy of binding. It coinsists of the libraries shown in Figure 3.3, and explained in the next few sections.

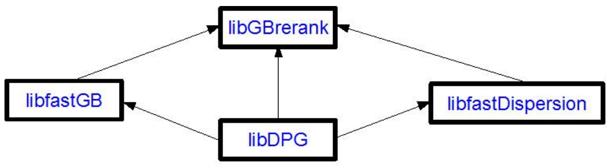


Figure 3.3: Dependency diagram of GB-rerank. Some libraries are shared with F2Dock.

3.2.1 libGBrerank

libGBrerank computes solvation energy as

$$E_{\text{sol}} = \underbrace{E_{\text{cav}} + E_{\text{vdw(s-s)}}}_{\text{nonploar}} + \underbrace{E_{\text{pol}}}_{\text{polar}}$$

It uses the $libfastGB$ explained in Section 3.1.7 to compute the polarization energy E_{pol} and a separate library named $libfastDespersion$ to compute the other two terms.

Finally it uses a weighted sum of the score of F2Dock and the solvation free energy to produce a score with which the candidate poses are reranked.

3.2.2 libfastGB

See Section 3.1.7.

3.2.3 libfastDispersion

The solute-solvent van der Waals interaction energy (also known as *dispersion energy*) is modeled by the following equation derived from the model of [15, 40]

$$E_{\text{vdw(s-s)}} \approx \rho_0 \frac{4\pi}{3} \sum_{i=1}^{M} \frac{1}{R_i^3} \tag{3.4}$$

where R_i is the discrete approximation of Born radii with the R^6 model. Hence the computation follows from the computation of fast Born radii (see Section 3.1.7).

To compute ΔE_{cav}, Instead of computing E_{cav} for A, B and each $A+B_{\mathbf{t},\mathbf{r}}$ separately, we approximate it with the buried surface area of $A + B_{\mathbf{t},\mathbf{r}}$ which is approximated using the same algorithm used for interface propensity filter in Section 3.1.6.

3.3 TexMol

TexMol is a software package for molecular computations and visualization.

It has built in support for visualizing proteins and RNA in a number of ways including Van der Waals surface, Lee Richards surface, Solvent excluded surface. It can open files in PDB, PQR, XYZ or XYZR format. TexMol is also capable of volume rendering.

TexMol also provides a front-end to many other in-house software including MolSurf, F2Dock, MolEnergy and Pocket/Tunnel extraction. MolSurf provides libraries for constructing hierarchical models of molecules, extracting/identifying interface atoms, computing molecular surface representation, meshing and mesh refinements etc. MolEnergy provides libraries for efficient computations of molecular potentials amd forces including electrostatics (Lennard-Jones), solvation energy in genralized Boltzman (GB) model, or the Poisson Boltzman (PB) model. And finally, it includes a front-end for F2Dock.

The complete TexMol package is too large to describe in the scope of this tutorial. Hence we shall only concentrate on TexMol's GUI support for F2Dock. Interested users can refer to [3] for details about the visualization pipeline of TexMol. Details about other supported software (MolSurf/MolEnergy), tutorials and link to publications can be found at $http://www.cs.utexas.edu/\ bajaj/cvc/software.shtml$.

3.3.1 Client/Server model of TexMol and F2Dock

TexMol does not include F2Dock as a library. Rather, F2Dock, GB-rerank, MolSurf as well as other required software are installed on a remote computing cluster. TexMol provides GUI to let users prepare different parameters related to surface generation, docking, reranking etc., selecting molecules (PDB/PQR) and submit jobs. TexMol communicates the parameters as well as the necessary input files to the a server. The server, named F2Dock server, recieves the job requests and executes the required softeware on the cluster. The folw is shown in Figure 3.4.

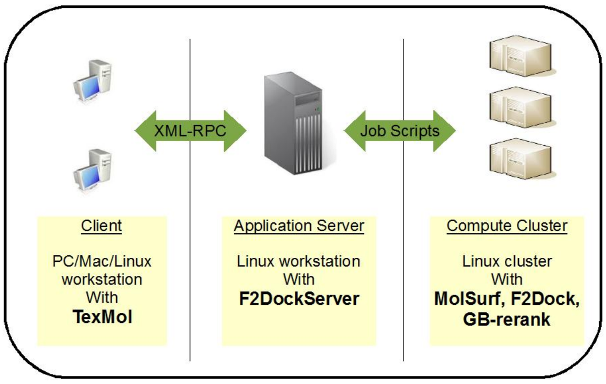

XML-RPC
Job Scripts
Client
PC/Mac/Linux workstation With TexMol
Application Server
Linux workstation With F2DockServer
Compute Cluster
Linux cluster With MolSurf, F2Dock, GB-rerank

Figure 3.4: Client-server communication between TexMol and F2Dock server

3.3.2 XML-RPC: the communication protocol

TexMol communicates with the F2Dock server using the XML-RPC protocol. The XML-RPC protocol encapsulates the data as well as the command in XML format and invokes a remote procedure call (RPC) to a designated network address and port. If another XML-RPC capable server installed at that address is programmed to listen to that port, then it receives the XML encoded message, executes the corresponding routine, and returns an appropriate response. We have chosen to use this protocol because it is available open source and also able to seamlessly communicate between software written in different languages. In other words, the F2Dock server has the ability to communicate with TexMol (written in C++) as well as a web server (written in java).

3.3.3 F2Dock support classes in TexMol

TexMol provides a number of GUI as well as command line interfaces for preapaing, submitting, monitoring and displaying docking jobs and results. See Sections 5.2, 6.2 and Chapter 4 for details about the usage of those UI. In this section we provide a very high level list of the classes and their functions.

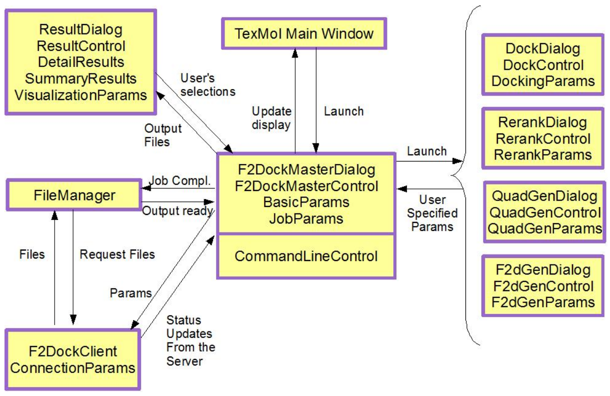


Figure 3.5: High level sketch of the F2Dock support classes in TexMol

Figure 3.5 shows the list of classes in the F2DockClient library of TexMol and the main comunication pattern between them. Note that, this is neither a standard class diagram, nor a data flow diagram. Please refer to the software documentation for further technical details.

The classes are designed based on the Model-View-Control (MVC) framework. There is a user interface class (View) for each of the major functions, namely- Job Management, Docking Parameter Selection, Reranking Parameter Selection, F2d/Quad Generation, Viewing the Results and finally Rendering the results. The data modified/used by each of these UI are defined in separate data classes (Model). The data classes provide functions to set up default values, update/retrieve values, check for consistency and validity etc. And finally, for each of the UI, a controller class (Control) is defined, which enables communication between the UI and the Data classes and also communicates with other control classes. The control class also ensures that a proper sequence of actions is followed, and prevents crashes/exceptions. However, there are also a few control classes which deal with XMLRPC communication support with the server.

Below we briefly discuss the tasks of the classes-

- **F2DockMasterDialog, F2DockMasterControl, JobParams, BasicParams:** The F2DockMasterDialog lets the user create, load, save and manage a number of jobs. The F2DockMasterControl class supports the operations of the dialog as well as acts as the launcher of other dialogs. It also acts as the conduit for communication between other classes and the XMLRPC client. The Job and Basic params classes are used to manage job and machine specific data.
- **DockDialog, DockControl, DockingParams:** DockDialog is used to display the default parameters to the user, letting the user to modify the parameters and specifying input/output file paths. The DockingControl class verifies that all necessary input have been received and notifies the F2DockMasterControl class. DockingParams are used to manage and verify the values of the parameters of F2Dock (see Section 5.1.2 for a list).
- **RerankDialog, RerankControl, RerankingParams:** Similar to docking.
- **F2dGenDialog, F2dGenControl, F2dGenParams:** Similar to docking.
- **QuadGenDialog, QuadGenControl, QuadGenParams:** Similar to docking.
- **ResultDialog, ResultControl, SummaryResults, DetailedResult:** THe Dialog displays the docking/reranking result as summaries and as detailed tabular lists. The dialog also displays pose specific statictics like docking scores, energies, interface area, residue contacts etc. If a pose is selected and a visualization mode is specified, it sends a update request to TexMol's main window via the F2DockMaster. The main window then displays the selected conformation. SummaryResults parses the docking/reranking output files, mines for summary and also prepares a list of DetailedResults.
- **F2DockClient, ConnectionParams** F2Dock client implements a XML-RPC client and connects to a remote server using the information from the ConnectionParams class. It sends the parameters and inputs files and retrieves servre responses. It also supports a multi-threaded 'Nagger' class which polls the server for updates regarding active jobs.
- **FileManager** FileManager retrieves and stores required output files from the server with the help of F2DockClient class.
- **CommandLineControl** This class is instantiated instead of the F2DockMasterControl when users use the command line version of TexMol. CommandLineControl takes the parameters from the command line and, based on the type of job, works with the corresponding control class, F2DockClient and FileManager to verify, prepare, submit and retrieve a job and its output.

Chapter 4

Input Preparation

4.1 Molecular representation

In this section we elaborate two schemes of representing molecules, namely the PDB file and the PQR file.

4.1.1 PDB

This is probably the most common and widely used textual representation of proteins and other large biomolecules. PDB is short for protein data bank, which maintains a database of solved protein structures in a uniform format.

Format

In the PDB file, there is an optional header containing details about the biochemical properties, classification etc. of the molecule. It also contains transformation/replication matrices to build the complete structure of the symmetric structure (and the structure of the assymetric unit is detailed in the body).

The most important part for the purpose of docking is the body of the file which consists of a hierarchical list of atoms grouped into residues which in turn are grouped into chains. Each atom is described by a single line of text.

Each line contains a unique serial number of the atom, its chain and residue number, the residue type, atom type and the position of the atom center in Cartesian (XYZ) coordinates. The width and position of each dataitem is fixed, which makes it easy to parse.

Preparation

For the purposes of docking, PDB files can be considered as input available at the outset, and hence there is no need to prepare the file as a whole. The PDB files are usually prepared by crystallographers and uploaded to the RSCB PDB database.

However, sometimes the PDB file may have missing atoms or residues which are later 'inserted' as part of a curation process. Also some steric clashes might be present initially or as a result of such insertions and hence residue locations (coordinates, not sequence) might need to be updated by looking up rotamer libraries. Another curation step is the removal of crystallized water (solvent) molecules from the structure.

4.1.2 PQR

PQR [14] files augment the PDB files by adding a radius and charge to each atom. Optionally, it is possible to perform some of the curations listed in the previous section as well as adding Hydrogen atoms to the residues.

Format

The PQR format leaves the original PDB format, in terms of the positions and width of each field, unchanged. It appends the radius and charge as two new columns at the end of each line. However, by default, the PQR file removes the chain ID and leaves the position blank. This can be addressed by using a '–chain' option while running the PDB2PQR software. But in that case, the resulting PQR file sorts the atoms based on their residue ids and may not maintain the same ordering of atoms as the PDB files.

Preparation

To prepare a PQR file from a PDB, one needs to install the PDB2PQR software. PDB2PQR is a software packaged written in Python and available at *http : //www.poissonboltzmann.org/pdb2pqr*

The above website contains all the details about installing and running PDB2PQR.

4.2 Molecular surface representation

Many of computations in docking/reranking including solvation energy, shape complementarity etc. depend on a smooth surface approximation of the proteins. Hence, the quality of the surface reprentation is highly important. We use our inhouse software package MolSurf to compute such high quality surfaces (RAW) and also approximate normals on them (RAWN).

4.2.1 RAW

The RAW file (with a file extension *.raw*) represents a surface mesh. The mesh is triangular by default, but it is also possible to produce quad or hex meshing. The mesh itself is a collection of points and a collection of simplices formed by connecting those points.

Format

The files starts by a single number n, representing the number of points on the mesh. This line is followed by n lines of three floating point values representing the Cartesian coordinates of a point on the surface.

The second part of the file starts with the number of simplices m followed by m lines, each line describing a single simplex. A simplex is a ordered list of vertices of the simplex. The vertices are ordered counterclockwise when seen from the outside. Each vertex is referred to using a single integer id between 0 and $n - 1$ corresponding to its position in the list of vertices.

Preparation

MolSurf includes a number of ways for generating a smooth surface from a PDB or PQR, including techniques based on the Adaptive Grids, the High order level set, basic Gaussian blurring etc. We refer the reader to the corresponding papers for further details of each method.

In the next section we detail the procedure of using MolSurf or TexMol to generate the RAW files.

However, the generated might contain 'bad' triangles or triangles containing angles which are too acute. These might cause problems in some calculations including normal approximation and Gaussian quadrature points sampling. So, MolSurf has additaional tools to improve the quality of any given mesh and produce a triangulation containing well behaved triangles only.

To prepare the surface, one needs to install MolSurf or TexMol. See the installation instructions in Sections 8.2.3 and 8.2.4. Though, we only mention MolSurf in the following examples, the Command is exactly the same for TexMol. All that needs to be changed in the name of the program.

Preparing the surface The command shown in Figure 4.1 can be used.

The command requires a path to the input PDB file, the path where the generated surface should be stored and an integer grid size g. The grid size specifies the the number of gridcells in a $g \times g \times g$ grid. Higher values of g would produce a better approximation of teh surface at the cost of a large number of vertices and simplices. We suggest using $g = 128$ for molecules of the size of an average protein.

```
MolSurf -surfaceUsingAdaptiveGrid <input pqr file> <output raw file> <grid size>
```

Figure 4.1: Preparing surface (RAW) using MolSurf

```
MolSurf -qualityImprove <input raw file> <output raw file>
```

Figure 4.2: Improving surface quality using MolSurf

Note that, it is possible to produce the surface using the other techniques like Gaussian blurring. Refer to TexMol's manual for details about such methods. We have only mentioned the ablve technique because it is the one used to prepare the dataset we have used with F2Dock.

Quality improvement The following command (Figure 4.2) accepts a mesh and produces a quality improved mesh.

4.2.2 RAWN

The RAWN file adds normals to the surface mesh by approximating outward pointing normals at each vertex of the mesh. Normals are neccessary while sampling Gauss quadrature points for solvation energy computaion (see Section 4.3.2) as well as for generating the extra skin layer outside the surface of the receptor (see section 4.3.1).

Format

The format is the same as the RAW format. The only difference is that each line describing a vertex contains six floating point values, the first three gives the position of the vertex and the other three represents the normal on that point.

Preparation

MolSurf (amd TexMol) provides tools to approximate the normals and produce a RAWN file. The following command (Figure 4.3) accepts a mesh and approximates normal on the vertices.

Example

In Figure 4.4 we show the VdW surfaces of the ligand and receptor of the complex 1A2K. And side by side we show the smooth solvent excluded surface (SES) generated by the methods described in this section.

4.3 Internal files for docking and reranking

In this section we detail some file formats representing either the molecule or the surface in some formats which are internally used by F2Dock or GB-rerank.

4.3.1 F2D

The F2D files is used to identify the skin and core regions of a protein, which are used to maximize skin-skin overlap in F2Dock's shape complementarity scoring (see Section 2.4 for details about this scoring term).

Given a receptor A and a ligand B- we define the skin and core regions as follows.

```
MolSurf -normals -average <input raw file> <output rawn file>
```

Figure 4.3: Approximating normal at mesh vertices using MolSurf

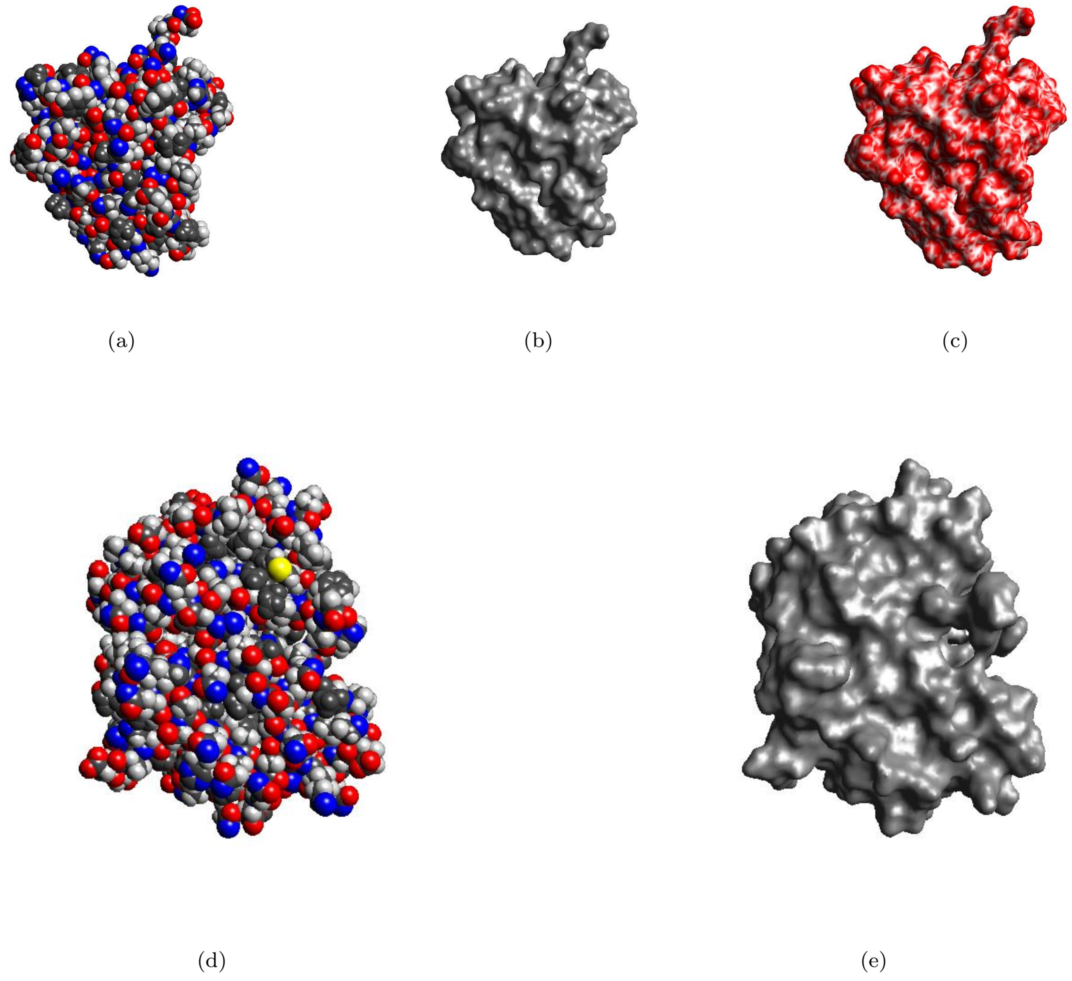


Figure 4.4: (a) The VdW surface rendered from the PQR file of the ligand of the complex with PDBID=1A2K. (b) The corresponding smooth surface (SES). (c) The surface colored by Mean curvature values. (d) The VdW surface rendered from the PQR file of the receptor of the complex with PDBID=1A2K. (e) The corresponding smooth surface.

- All atoms of B, which are partially exposed to the solvent (contributes to the SES) are marked as skin. The remaining atoms of B are marked as core atoms.
- All atoms of A are considered core atoms. A separate floating layer (slightly offset from the SES) of atoms are artifically added and are considered the skin.

Generating the F2D files involve the detection of surface atoms for the ligand and the population of the grown skin for the receptor. Both these functions are implemented as part of MolSurf.

The original version of F^2*Dock* used the traditional *double skin layer* approach for shape complementarity. Two *skin regions* are defined (see Figure 4.5): **(1)** the complementary region of A, defined by a *grown skin region*, by introducing a 1-layer of pseudo-atoms on the surface of A, each with the same radius which is chosen to make its size comparable to that of a solvent molecule, and **(2)** the *surface skin* of B, which consists of the surface atoms of B.

The current version of F^2*Dock* uses an improved version of the traditional double-skin layer approach described above which differs from the traditional approach in the following ways (see Figure 4.6). The receptor skin layer does not touch the receptor van der Waals surface (i.e., there is a gap between the skin and the core regions), and also the radius of those skin atoms differ from that in the traditional approach. In our experiments pseudo skin atoms of radius 1.1Å with their centers placed at a distance

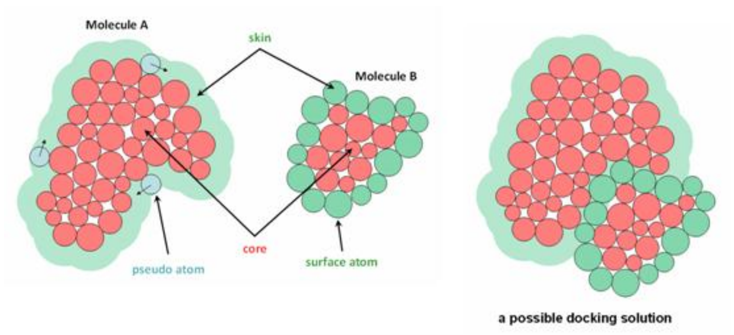
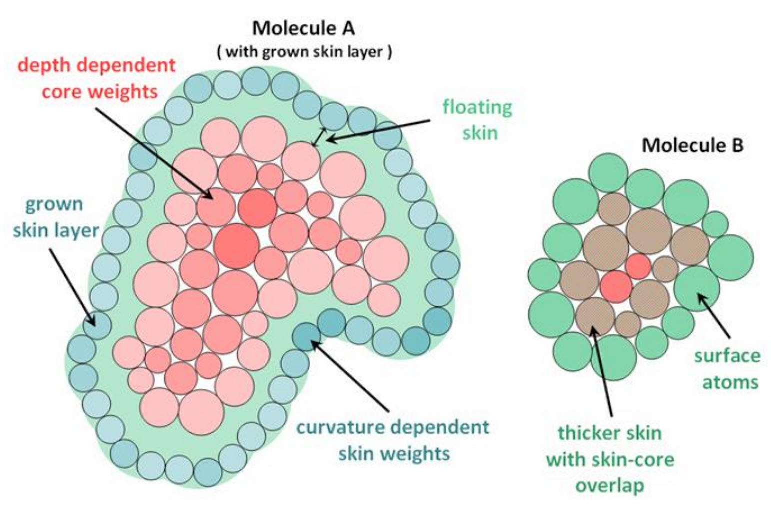


Figure 4.5: Traditional *double skin-layer* approach for shape complementarity [41, 22, 27, 10] which is also used in the first version of F^2*Dock* [5].



Molecule A
(with grown skin layer)
depth dependent core weights
floating skin
grown skin layer
curvature dependent skin weights
Molecule B
surface atoms
thicker skin with skin-core overlap

Figure 4.6: Skin and core definitions and weights in the current version of F^2*Dock*.

of 1.7Å from the receptor vdW surface (so there is a gap of 0.6Å) performed much better than the traditional approach where pseudo atoms of radius 1.4Å are placed touching the receptor vdW surface.

Format

The F2D file format is the same as a PQR file with an additional column added at the end. This additional column contains a characted I/E, where I stands for *internal* (i.e. core atom) and E stand for *external* (i.e. skin atom).

Additionally, the F2D file of the receptor contains extra (groen skin) atoms at the end. Their types are specified as 'HETATM'. The atom ID continues in increasing order from the highest atom ID of the original molecule. Similar rule is applied for the residue and chain IDs. The position (XYZ coordinates) are determined by the algorithm we discuss in the next section. The radii of all the grown atoms is the same and can be defined by the user while generating the F2D files.

Preparation

Growing the floating skin for receptor The function in Figure 4.7 of MolSurf generates the grown floating skin layer for a receptor-

Given a pqr file and a quality improved surface with normals (rawn), the function places pseudo-skin atoms $r + f$ distance away along the normal from each point on the surface, where r is the user-specified radius of the pseudo-atoms and f is the width of the floating band. It uses a neighborhood data structure

```
MolSurf -populateSASUsingMesh <pqr file> <rawn file> <xyz output file> <r> <f> <g> 0
```

Figure 4.7: Creating a floating skin layer outside the receptor surface using MolSurf

```
MolSurf -getSurfaceAtoms <input pqr file> <output xyz file>
```

Figure 4.8: Identifying the surface atoms of the ligand using MolSurf

```
MolSurf -generateF2d <pqr file> <xyz file> <f2d file> <receptor=1/ligand=0> 0
```

Figure 4.9: Identifying the surface atoms of the ligand using MolSurf

(DPG) to ensure that centers of any two of the added pseudoatoms are at least g distance apart from each other. The output is a list of positions for the pseudo-atoms.

Identifying the surface atoms of the ligand The function in Figure 4.8 of MolSurf identifies the surface atoms of a ligand.

This function takes the pqr file of the ligand and produces a list of atoms each marked as internal/external.

Creating F2d After the XYZ files have been produced by the either of above two schemes, generating the F2D file only requires consolidating the information present in the PQR and the XYZ files. The following function is available to do it.

Generating F2D using TexMol

TexMol also provides easy to use methods for generating F2D files directly from the PDB files. The user can optionally store the intermediate files (PQR, XYZ, RAWN etc.). TexMol submits the PDB files to a remote server which is running MolSurf and retrieves the outputs when they are ready. TexMol provides both a graphical and a command line interface for this.

Using TexMol in command line To produce f2d files from pdbs, the following command need to be executed.

```
./TexMol -f2dgen <>.inp
```

Figure 4.10: Generating f2d files using the command line interface of TexMol

The input file (which must have a .inp extension) contains exactly three lines.

```
staticMoleculePDB <>.pdb
movingMoleculePDB <>.pdb
resolution <int>
```

staticMoleculePDB and *movingMoleculePDB* specify the pdb file of the receptor and the ligand respectively. *resolution* specifies the size of the grid used and should be 64, 128 or 256. After completion, the pqr and f2d files are stored.

Using TexMol with graphical interface The interface shown in Figure 4.11 can be launched from the main window (see Section 5.2.1) of F2Dock in TexMol.

The user can select two pdb files (receptor and ligand) and a resolution (fine = 256x256x256, medium = 128x128x128 and coarse = 64x64x64), and choose to save the generated f2d and/or pqr files.

Example

In Figure 4.12 we show the VdW surfaces of the ligand of the complex 1A2K. The ligand's surface atoms and internal atoms are then detected by the methods decribed above to produce an F2D file. In the same figure we show the core (extracted from the F2D file) atoms of the ligand separately. The skin atoms are the remaining ones.

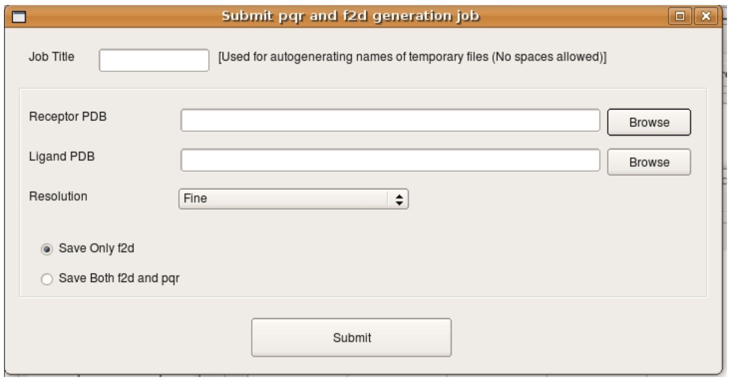

Submit pqr and f2d generation job
Job Title
[Used for autogenerating names of temporary files (No spaces allowed)]
Receptor PDB
Browse
Ligand PDB
Browse
Resolution
Fine
Save Only f2d
Save Both f2d and pqr
Submit

Figure 4.11: F2d Generation using the graphical interface of TexMol

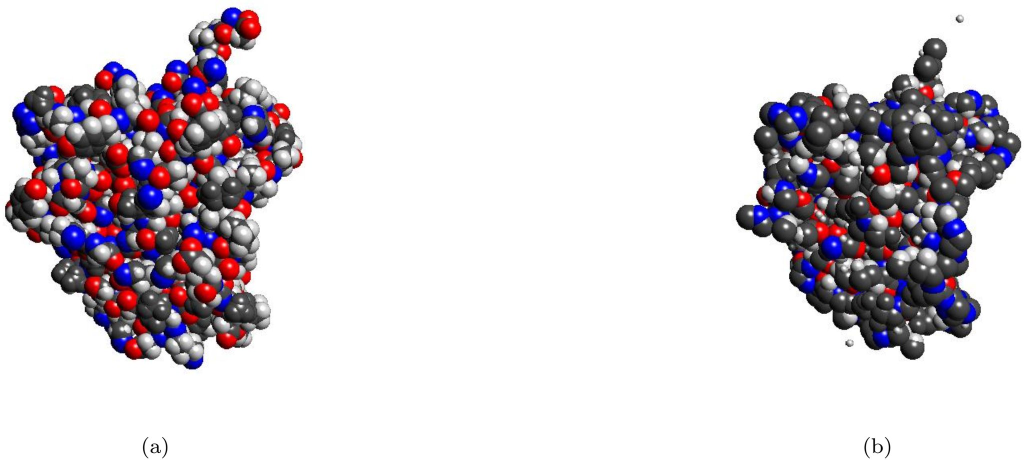


Figure 4.12: (a) The VdW surface rendered from the PQR file of the ligand of the complex with PDBID=1A2K. (b) The core atoms of the ligand

In Figure 4.13 we show the VdW surfaces of the receptor of the complex 1A2K. A floating layer of pseudoatoms are added outside the molecule to form a skin layer and the information is stored in the F2D file. The figure shows the original receptor molecule (Figure 4.13(a)) and with the grown skin which encapsulates all the original atoms (Figure 4.13(b)). Then Figure 4.13(c) shows the core and the skin together by clipping the skin at $x = 20$. Figure 4.13(d) shows a zoomed in view to highlight the empty band between the skin and the core.

4.3.2 QUAD

The QUAD files contains Gaussian integration (quadrature) points sampled on a smooth surface. See [7, 13] for details about the sampling procedure. The sampled points themselves are used for Born radii approximation and solvation enrgy computation in the Generalized Born model (see Section 2.6 for details).

Format

Each line of the quadrature points file contains the description of an integration point on the surface of the molecule. Each such line consists of 7 floating point numbers: the first three are the co-ordinates of the point followed by three floating point numbers giving the x, y and z components, respectively, of the unit outward surface normal at that point, while the last one is the weight given to that point.

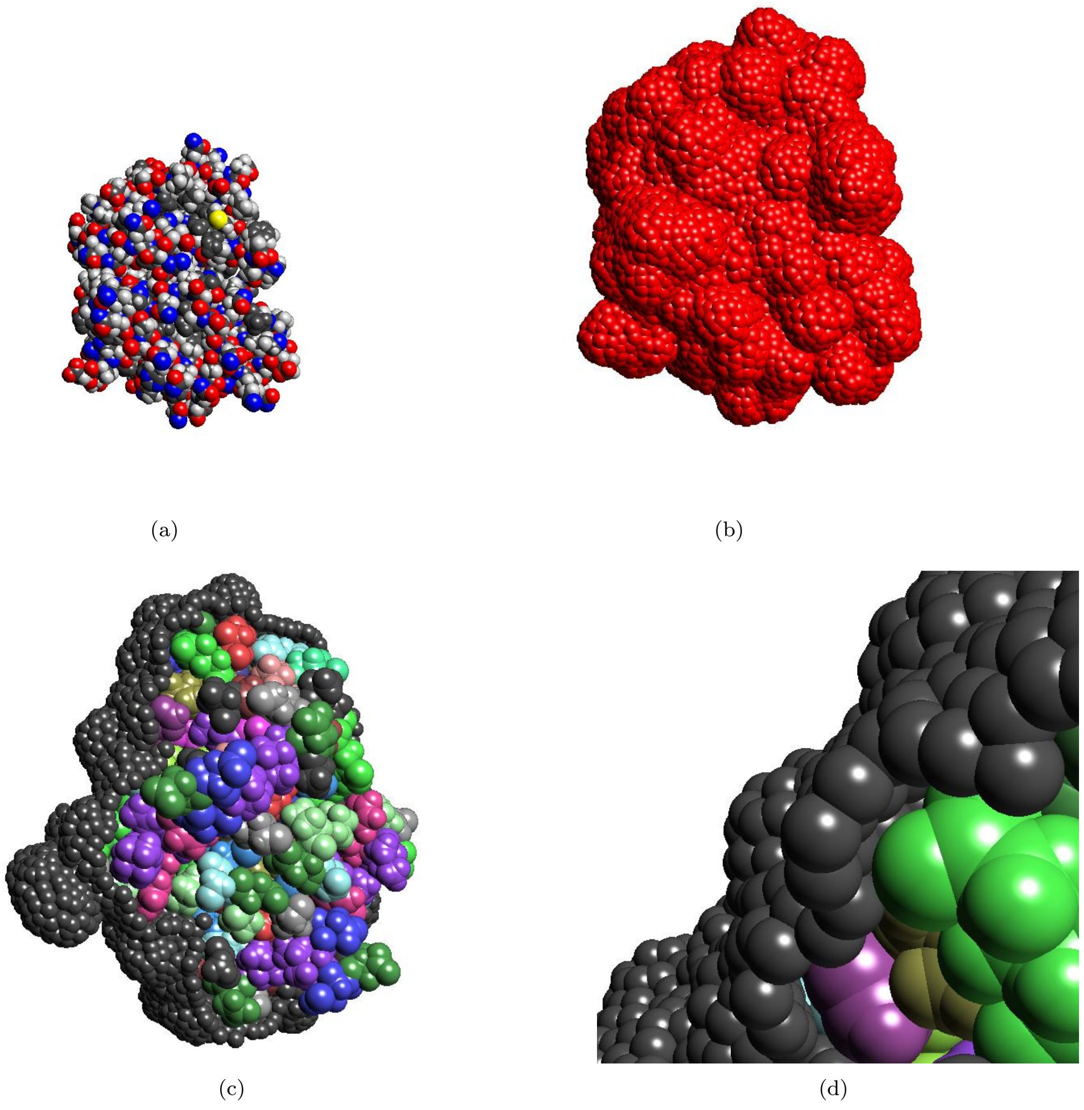


Figure 4.13: (a) The VdW surface rendered from the PQR file of the receptor of the complex with PDBID=1A2K. (b) The receptor with a grown skin layer. (c) The skin and core of the receptor (core colored by residue types for better visualization). (d) A close up view of the skin-core separation.

Preparation using MolSurf

The following command (Figure 4.14) accepts a mesh with normals and samples integration points.

Preparation using TexMol

Both the command line and the graphical interface for QUAD file generation in TexMol is quite similar to the F2D file generation.

Using TexMol in command line To produce quad files from pdbs, the following command need to be executed.

The input file format is the same as f2d generation. After completion, the pqr, rawn (surface) and quad files are stored.

```
MolSurf -aSpline -quad <input rawn file> gaussian 1 <output quad file>
```

Figure 4.14: Sampling Gaussian integration points on a surface mesh using MolSurf

```
./TexMol -f2dgen <>.inp
```

Figure 4.15: Generating quad files using the command line interface of TexMol

Using TexMol with graphical interface Same as F2d generation.

Example

In the example in Figure 4.16 we show the quadrature points generated by the above mentioned methods for the receptor of the 1A2K complex.

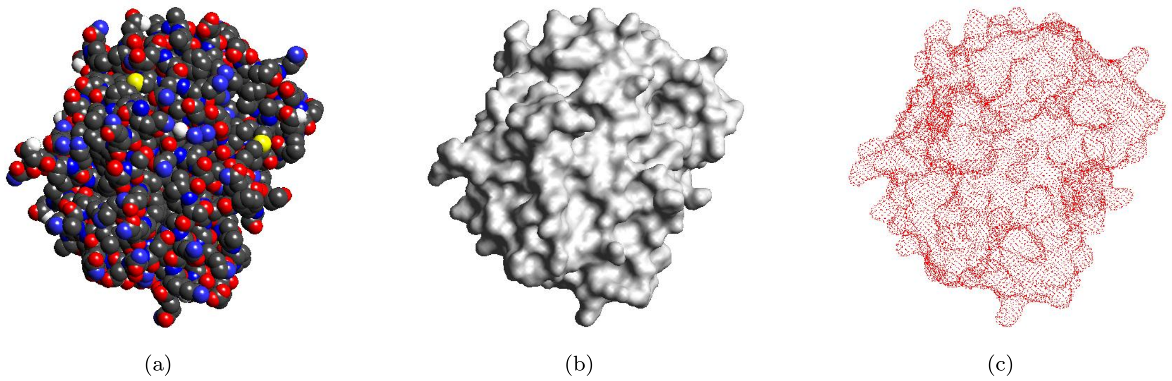


Figure 4.16: (a) The VdW surface rendered from the PQR file of the receptor of the complex with PDBID=1A2K. (b) The corresponding smooth surface (SES). (c) Gaussian integration points sampled on the smooth surface.

Chapter 5

Docking

The docking pipeline consisting search, score and filter (as explained in Section 2) are implemented in the F2Dock software. The basic version of this software has a command line interface only. Users can execute the program by passing it a file which lists the parameters and location of input files. However, we have also developed a user-friendly graphical user interface which can be used to interactively prepare the inputs and other parameters, run F2Dock, retrieve/store the results, and also analyze and visualize the results in detail. This graphical interface is part of TexMol.

This chapter describes both of these in detail.

5.1 Docking from the command line: F2Dock basic

Once the F2Dock executable is properly installed (see Section 8.2.1 for installation instructions), it can be executing simply by the following command-

The command would start executing F2Dock's pipeline and at the end produce a output file containing details of the predicted conformations.

The format of the parameter file, explanation of the parameters and the format and content of the output file are explained with examples in the next subsections.

5.1.1 The parameter file

The parameter file contains a sequence of lines. Each line specifies the value of a parameter. Each line starts by mentioning the name of the parameter and then provides a value for it.

A few parameters, like locations of input files, are mandatory and must be present in the parameter file (Section 5.1.2). Other parameters are optional, and can be used to assign non-default values to internal variables of F2Dock (Section 5.1.3). For example, F2Dock uses the clash filter by default, but it is possible to turn it off. It is also possible to tweak the weights of the scoring terms.

In the next two subsections, the types, default values and purpose of the parameters are explained. In the description, the following symbols are used to specify the types-

C:	single character/letter
S:	string of characters/letters
I:	integer
F:	floating point
B:	Boolean (true/false)

```
F2Dock <paramFileName>
```

Figure 5.1: Executing F2Dock

```
movingMoleculePQR data/PQR/1A2K_l_u.pqr
staticMoleculePQR data/PQR/1A2K_r_u.pqr
complexType A
outFile output/1A2K_dock.out
...
```

Figure 5.2: Example parameter file for F2Dock

5.1.2 List of mandatory parameters

These parameters must be specified to execute F2Dock. For each file name parameter below the name of the file must be specified with full path if it is not in the current folder. Details of these filetypes are explained in Section 4.

- *staticMolecule* (S): Name of the F2D file for the molecule that is to be held stationary during docking. Typically the static molecule is the larger of the two molcules to be docked.

- *staticMoleculePQR* (S): Name of the PQR file for the static molecule. Current version of F^2Dock only works on PQR files without chain information

- *staticMoleculeQUAD* (S): Name of the quadrature points file for the static molecule.

- *movingMolecule* (S): Name of the F2D file for the molecule that is to be moved around during docking. Typically the smaller of the two molcules to be docked is treated as the moving molecule for performance reasons.

- *movingMoleculePQR* (S): Name of the PQR file for the moving molecule.

- *movingMoleculeQUAD* (S): Name of the quadrature points file for the moving molecule.

- *outFile* (S): Name of the file to which F^2Dock will output the potential docking solutions (poses).

5.1.3 List of optional/advanced parameters

F2Dock have been trained to learn default values (for different type of complexes) for the parameters listed here. The values optimize the performance of F2Dock on ZDock 2.0 benchmark. However, we have still kept the parameters open to the advanced users.

Complex Type

F2Dock's default parameters were trained for four different complex types: antibody-antigen, enzyme-inhibitor or enzyme-substrate, neither antibody nor enzyme, and unknown. The type itself can be specified in the parameter file so that F2Dock can choose proper default values for other parameters.

- *complexType* (C): Type of the docked complex: *A* (antibody-antigen), *E* (enzyme-inhibitor or enzyme-substrate), *G* (neither A nor E), and *U* (unknown). The default value is *U* in which case F^2Dock tries to identify the complex type itself (very successfully for antibody-antigen, and with moderate success for enzyme-inhibitors and enzyme-substrates).

Output related parameters

- *numSolutions* (I): A positive integer giving the upper bound on the number of top docking poses F^2Dock should output. Default value is $20,000$.

- *rmsdAtoms* (S): Name of a text file containing the coordinates of the atoms of the moving molecule that must be included in the RMSD calculation. If this parameter is not specified RMSD values will not be calculated.

Parameter	complexType		
	A	*E*	*G*
peaksPerRotation	3	2	4

Table 5.1: Default values for *peaksPerRotation* based on complexType.

Parameter	complexType		
	A	*E*	*G*
skinSkinWeight	0.73	0.78	0.57
skinCoreWeight	-0.31	-0.08	-0.23
coreCoreWeight	31.00	5.00	5.00
curvatureWeightingRadius	4.5	6.0	4.5

Table 5.2: Default values for various shape complementarity related parameters based on complexType.

Search space parameters

These parameters control the sampling of the $R^3 \times SO^3$ search space. These can be modified to trade-off between speed and accuracy.

- *rotFile* (S): Name of a text file containing the rotations to be applied to the moving molecule during docking. Each line of the file contains one rotation specified by three Euler angles.

- *numRot* (I): The top *numRot* rotations from *rotFile* are used. If left unspecified all rotations will be used.

- *peaksPerRotation* (I): Only the top *peaksPerRotation* peaks will be retained for each rotation. Default values are given in Table 5.1.

- *gridSpacing* (F): A positive floating point value giving an upper bound on the spacing (in Å) between adjacent grid points in the 3D spatial FFT grid. F^{2}Dock consults the effGridFile file in order to compute a grid size so that the grid spacing does not exceed the upper bound provided by the user, and at the same time FFT can be computed as efficiently as possible. Thus numFreq parameter depends on this approximated grid spacing. The default upper bound is 1.2 Å.

- *numFreq* (I): A positive integer n specifying the number of frequencies n^3 to be used during FFT computations. If this parameter is not specified it is computed from the gridSpacing parameter.

- *sparseFFT* (B): If set to *true* the sparsity of the input and the output grids will be exploited for faster computation. Default value is *true*.

- *narrowBand* (B): If set to *true* only the positions of the moving molecule that lie within a narrow band around the static molecule will be considered for finding potential solutions, othewrise the entire 3D grid is searched. Default value is *true*.

Shape Complementarity

These parameters are used to control the weights of the shape complementarity terms and to specify the nature of the skins. See [12] for details.

- *skinSkinWeight* (F): A positive floating point value specifying the reward given to unit skin-skin overlap during shape-complementarity scoring. See Table 5.2 for default value.

- *skinCoreWeight* (F): Specifies the wight (reward or penalty) given to skin-core overlaps during shape-complementarity scoring. Default value is given in Table 5.2.

- *coreCoreWeight* (F): A positive floating point value specifying the penalty given to unit core-core overlaps during shape-complementarity scoring. See Table 5.2 for default value.

Parameter	complexType		
	A	*E*	*G*
elecWeight	0.72	0.15	0.72
elecKernelVoidRad	0.0	3.0	0.0
elecKernelValLow	4.0	1.0	4.0

Table 5.3: Default values for various electrostatics related parameters based on complexType.

- ***singleLayerLigandSkin* (B):** If set to *false* the core atoms adjacent to the skin atoms of the moving molecule are also considered part of the skin. Default value is *false.*

- ***curvatureWeightedStaticMol* (B):** If set to *true* the skin atoms of the static molecule are weighted based on curvature. Default value is *true.*

- ***curvatureWeightedMovingMol* (B):** If set to *true* the skin atoms of the moving molecule are weighted based on curvature. Default value is *false.*

- ***bandwidth* (F):** The width of the onion shell type bands of core atoms constructed for computing depth-dependant core weights. Default value is 2 Å.

- ***gradFactor* (F):** The weight of the core atoms in any given shell is a factor of *gradFactor* more than the weights of the core atoms in the shell just outside of it. Default value is 1.1.

Electrostatics

These parameters are used to control the electrostatics scoring.

- ***elecWeight* (F):** The weight given to the electrostatics score computed by F^{2}Dock. See Table 5.3 for default value.

- ***elecKernelVoidRad* (F):** Specifies the d_0 distance in computing the distance-dependant dielectric constant $E(\mathbf{x})$ using Equation 5.1 given below (a generalization of the Gabb et al. expression [17]). Default value is given in Table 5.3.

$$E(\mathbf{x}) = \begin{cases} 0 & \text{if } ||\mathbf{x}|| \leq d_0, \\ v_l & \text{if } d_0 < ||\mathbf{x}|| \leq d_l, \\ c_1||\mathbf{x}|| + c_2 & \text{if } d_l < ||\mathbf{x}|| \leq d_h, \\ v_h & \text{if } d_h < ||\mathbf{x}||, \end{cases} \tag{5.1}$$

where, $c_1 = \frac{v_h - v_l}{d_h - d_l}$, and $c_2 = v_l - d_l c_1$.

- ***elecKernelDistLow* (F):** Specifies the d_l distance in Equation 5.1. Default value is 6 Å.

- ***elecKernelDistHigh* (F):** Specifies the d_h distance in Equation 5.1. Default value is 8 Å.

- ***elecKernelValLow* (F):** Specifies the v_l value in Equation 5.1. See Table 5.3 for default value.

- ***elecKernelValHigh* (F):** Specifies the v_h value in Equation 5.1. Default value is 80.

- ***elecRadiusInGrids* (F):** A positive floating point value specifying the constant radius of each atom within which its charge is diffused using a Gaussian during electrostatics scoring. Default value is 2.9.

Hydrophobicity

- ***hydrophobicityWeight* (F):** Weight given to the ratio $r_h = \frac{\textit{interface hydrophobic score}}{\textit{interface hydrophilic score}}$ computed by F^{2}Dock. Default value is given in Table 5.4.

- ***hydrophobicityProductWeight* (F):** Weight given to the product $r_h \times$ *interface hydrophobic score* computed by F^{2}Dock. Default value is 0.001.

Parameter	complexType		
	A	*E*	*G*
hydrophobicityWeight	8.5	9.0	8.5
hydroRatioTolerance	8.0	8.0	10.0
hydroMinRatio	1.5	1.22	0.5
hydroRatioNumeratorLow	1.25	2.0	2.0
hydroRatioDenominatorLow	0.45	1.0	0.2
hydroRatioDenominatorHigh	2.5	7.0	6.0

Table 5.4: Default values for various hydrophobicity related parameters based on complexType.

- ***hydroRatioTolerance* (F):** A positive floating point value giving the upper bound on r_h. If r_h exceeds this value r_h is set to *hydroPenalty* (*hydroPenalty* $= -10$ in the current version). See Table 5.4 for default value.

- ***hydroMinRatio* (F):** A nonnegative floating point value giving the lower bound on r_h. If r_h is below this value r_h is set to *hydroPenalty.* Table 5.4 gives the default value.

- ***hydroRatioNumeratorLow* (F):** Lower bound on *interface hydrophobic score.* If this value is below the lower bound r_h is set to *hydroPenalty.* See Table 5.4 for default value.

- ***hydroRatioNumeratorHigh* (F):** Upper bound on *interface hydrophobic score.* If this value is above the upper bound r_h is set to *hydroPenalty.* Default value is 100.0.

- ***hydroRatioDenominatorLow* (F):** Lower bound on *interface hydrophilic score.* If this value is below the lower bound r_h is set to *hydroPenalty.* See Table 5.4 for default value.

- ***hydroRatioDenominatorHigh* (F):** Upper bound on *interface hydrophilic score.* If this value is above the upper bound r_h is set to *hydroPenalty.* Table 5.4 gives the default value.

- ***twoWayHydrophobicity* (B):** If set to *false* only the hydrophobicity of the atoms of the static molecule is considered, otherwise both molecules are considered. Default value is *true.*

- ***useInterfacePropensity* (B):** If set to *true* interface propensity values from [24] are used, othewise hydrophobicity values from [8] are used. Default value is *true.*

- ***perResidueHydrophobicity* (B):** If set to *true* per residue hydrophobicity values from [8] are used, per atom hydrophobicity values [25] are used. Default value is *true.* If *useInterfacePropensity* is set to *true* this parameter is ignored (i.e., always assumed to be *true*).

- ***staticMolHydroDistCutoff* (F):** A non-negative floating point value giving the maximum distance from the surface of a core atom of the static molecule to the center of any skin atom in order to consider that core atom for hydrophobicity calculation. Default value is 4.0 Å.

Other scoring terms

- ***hbondWeight* (F):** The weight given to the hydrogen bonding score computed by F^{2}Dock. Default value is 0.0.

- ***hbondDistanceCutoff* (F):** A positive floating point value specifying the surface to surface distance upper bound between any pair of donor and acceptor atoms. The default value is 2 Å.

- ***simpleChargeWeight* (F):** Weight given to the simple charge complementarity score computed by F^{2}Dock. Default value is given in Table 5.5.

- ***simpleRadExt* (F):** The radius of each atom is extended by this value for simplified charge complementarity computation. Default value is 1.5 Å.

Parameter	complexType		
	A	*E*	*G*
simpleChargeWeight	0.1	5.5	2.0

Table 5.5: Default values for various simple charge complementarity related parameters based on complexType.

Parameter	complexType		
	A	*E*	*G*
clashTolerance	2	9	10
clashWeight	-30	-0.5	-0.5

Table 5.6: Default values for various clash filter related parameters based on complexType.

Clash Filter

- *applyClashFilter* (B): If set to *true* clash filter is applied. Default value is *true.*

- *eqmDistFrac* (F): Two atoms are considered to be in a clash if the distance between the atom centers is less than *eqmDistFrac* fraction smaller than the sum of their radii. Default value is 0.5.

- *clashTolerance* (I): Upper bound on the number of atomic clashes permitted. Default value is given in Table 5.6.

- *clashWeight* (F): Weight given to each clash when added to the total score. See Table 5.6 for default value.

Lennard-Jones Filter

- *applyVdWFilter* (B): If set to *true* Lennard-Jones filter is applied. Default value is *true.*

- *vdWCutoffLow* (F): If #clashes < *clashTolerance* / 2, poses with vdW potential ¿ *vdWCutoffLow* are penalized. Default value is 0.0.

- *vdWCutoffHigh* (F): If #clashes $\geq$ *clashTolerance* / 2, poses with vdW potential ¿ *vdWCutoffHigh* are penalized. Default value is given in Table 5.7.

- *vdWEqmRadScale* (F): All r_{eqm} values are multiplied by this factor. Default value is 0.3.

- *epsilonLJ* (F): Error control parameter ($\in (0, 1]$) for Lennard-Jones potential. Default value is 0.5.

- *useSSE* (B): Is set to *true* SSE (Streaming SIMD Instructions) will be used for faster execution. Default is *false.*

Hydrophobicity Filter

- *applyPseudoGsolFilter* (B): If set to *true* hydrophobicity filter will be applied. Default value is *true.*

- *pseudoGsolWeight* (F): Weight given to the value $\frac{(interface\ hydrophobic\ score)^2}{interface\ hydrophilic\ score}$ computed by the hydrophobicity filter which is then added to the total score. Default value is 0.0. This parameter is now redundant and will be removed.

Parameter	complexType		
	A	*E*	*G*
vdWCutoffHigh	0.0	20.0	5.0

Table 5.7: Default values for various Lennard-Jones filter related parameters based on complexType.

- ***useInterfacePropensity* (B):** Same as in Hyrophobicity scoring term.

- ***perResidueHydrophobicity* (B):** Same as in Hyrophobicity scoring term.

Dispersion Filter

- ***applyDispersionFilter* (B):** If set to *true* dispersion filter will be applied. Default value is *false*.

- ***dispersionEnergyLimit* (F):** If the computed dispersion energy has value larger than *dispersionEnergyLimit* it is set to *dispersionEnergyLimit*, and if it has value smaller than -*dispersionEnergyLimit* it is set to -*dispersionEnergyLimit*. Default value is 1000.0.

- ***dispersionMinAtomRadius* (F):** Smallest permitted atom radius during dispersion energy calculation. Default value is 0.1 Å.

- ***dispersionWeight* (F):** Weight given to the dispersion energy value computed which is then added to the total score. Default value is 0.0.

- ***epsilonBR* (F):** Error control parameter ($\in (0, 1]$) for dispersion energy approximation. Default value is 0.3.

Reranking

F2Dock has a built-in 'soft'-reranking procedure. It uses the scores of the filters and a coarse grained solvation energy computation to rerank the results. The parameters are-

- ***rerank* (B):** If set to *true* the output of F2Dock will be reranked based on various filters. Default value is *false*.

- ***numRerank* (I):** Number of top solutions to rerank. Default value is 2000.

- ***applyAntibodyFilter* (B):** If set to *true* antibody filter will be applied which filters based on the active sites on the antibody. Default value is *true*.

- ***applyEnzymeFilter* (B):** If set to *true* enzyme filter will be applied which filters based on the abundance of Glycine residues. Default value is *true*.

- ***applyResidueContactFilter* (B):** If set to *true* filtering will be done based on residue contact preferences (from Table III, page 94 of [19]). Default value is *true*.

- ***rerankerF2DockScoreWeight* (F):** Weight given to the original F2Dock score. Default value is 100.0.

- ***rerankerPseudoGsolWeight* (F):** Weight given to the value $\frac{(\textit{interface hydrophobic score})^2}{\textit{interface hydrophilic score}}$ computed by the hydrophobicity filter. Default value is 1.0.

Other params

- ***clusterTransRad* (F):** The radius of the level $i \in [1, 3]$ cluster is $i \cdot \textit{clusterTransRad}$. Default value is 1.2 Å.

- ***clusterTransSize* (F):** Maximum number of docking poses (i.e., geometric center of the moving molecule) in cluster $i \in [1, 3]$ is $i \cdot \textit{clusterTransSize} + c_i$, where $c_1 = 0, c_2 = 1$ and $c_3 = 3$. Default value is 1.

- ***numThreads* (I):** A positive integer specifying the number of concurrent threads to use. Default value is 4.

- ***scoreScaleUpFactor* (F):** F^2Dock scores are scaled up by this factor. Default value is $10,000$.

5.1.4 Making sense of the output file

The output file of F2Dock has some distinct parts, which are described in the following subsections.

The input header

The input header lists all the input parameters, both the defaults and the user-specified ones. The header starts with the tag "INPUT PARAMETERS" and looks like Figure 5.3.

```
# INPUT PARAMETERS:
#
# numThreads = 4
# breakDownScores = 0
# numberOfPositions = 20
# gridSize = 432
...
```

Figure 5.3: The input header in F2Dock's output file

The output header

The results of docking are reported in tabular format. Each row of the result corresponds to a candidate hit/conformation. The output header describes the format of table (i.e. the columns) by mentioning the columnheaders and the type of variable it represents. See Figure 5.4.

```
# OUTPUT FORMAT: 29
# COLNAME rank int
# COLNAME score float
# COLNAME shape float
# COLNAME ssr float
# COLNAME ccr float
# COLNAME scr float
# COLNAME elec float
# COLNAME hbond float
# COLNAME hydrophobicity float
# COLNAME smplcomp float
# COLNAME vdw float
# COLNAME clashes int
# COLNAME pgsol float
# COLNAME pgsolh float
# COLNAME deldispe float
# COLNAME mat1 float
# COLNAME mat2 float
# COLNAME mat3 float
# COLNAME mat4 float
# COLNAME mat5 float
# COLNAME mat6 float
# COLNAME mat7 float
# COLNAME mat8 float
# COLNAME mat9 float
# COLNAME mat10 float
# COLNAME mat11 float
# COLNAME mat12 float
# COLNAME conf int
# COLNAME rmsd float
```

Figure 5.4: The output header in F2Dock's output file

We explain each column below-

- rank = the rank of the hit (rank determined by score)

- score = total score (weighted sum of the terms)
- shape = score of shape-complementarity terms
- ssr = skin-skin overlap score
- scr = skin-core overlap score
- ccr = core-core overlap score
- elec = score of electrostatics term
- hbond = score of hydrogen bond term
- hydrophobicity = score of hydrophobicity term
- vdw = score of vdw filter
- clashes = number of clashes found in clash filter
- pgsol, pgsolh, deldispe = values found in interface-propensity filter
- mat1-mat12 = the transformation matrix that places the ligand to this specific conformation
- rmsd = the root mean squared distance between the locations of the interface atoms of the known solution (if specified using the rmsdAtoms parameter), and the locations of those same atoms in the current conformation

Detailed outputs These are N rows in the above mentioned format where N is the total number of solutions. The solutions are listed by decreasing order of their rank. The section starts with a "START PEAKS" tag.

```
# START PEAKS
20 -13.48773 192.29153 0.00000 0.00000 ...
```

Figure 5.5: Detailed list of candidate solutions in F2Dock's output file

The summary

The summary lists the number of hits ($rmsd < 5\mathring{A}$) in the specific ranges of the ranked outputs. And then also reports the highest ranked hit and the lowest rmsd hit.

```
# Hits in Range:
# [1, 1] --> 1
# [1, 10] --> 4
# [1, 100] --> 7
# [1, 1000] --> 11
# [1, 10000] --> 11
# [1, 100000] --> 11
# good peaks under 5 A: count = 2 highest rank = 1 min RMSD = 4.200000
#
# best peak: rmsd = 3.600000 rank = 11 score = 158.247224
```

Figure 5.6: Summary of docking in F2Dock's output file

5.1.5 Scripts

5.2 Docking via GUI: TexMol

TexMol provides a user-friendly interactive graphical interface to prepare, submit, manage docking jobs and analyze the results. The docking (and reranking) dialogs can be launched by selecting 'Dock' from the utilities menu of TexMol.

5.2.1 Job management

Once launched, the dialog shown in Figure 5.7 shows up. This dialog provides options to create new docking/reranking and pqr/f2d/rawn/quad file generations, save created jobs, load previously saved jobs and viewing available results.

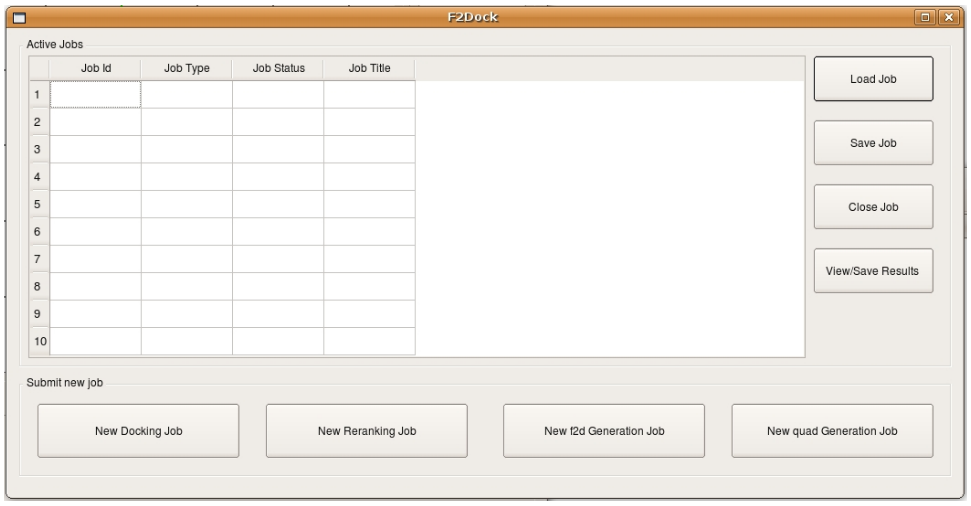


Figure 5.7: Job management

The most important content of this dialog is the tabular list of active jobs. The list displays the following for each job-

- **Job Id:** A unique integer id assigned by the server when the job was submitted.
- **Job Type:** The job could be of four possible types, namely *Docking*, *Reranking*, *F2d Generation* and *Quad generation*. The first two types are obvious, the last two types represent jobs for generating the input files (see Section 4) only.
- **Job Title:** It is a string given by the user to refer to the job. It is useful for keeping track of jobs if multiple jobs are submitted. If the user does not provide a title, TexMol generates a title basd on time and date during the job submission phase.
  - The title is used to store/load already created jobs. Each saved job corresponds to a file titled 'Jobtitle.job'.
  - The parameter list for docking (see Section 5.1.1), reranking etc. are created as 'Jobtitle.inp'.
- **Job Status:** This column reports the status of a job. The status can be one of the following-
  - **Submitted:** Once successfully created and submitted to the server, a job shows up on the list and its status is set to *Submitted*. TexMol polls the server in the background for all incomplete jobs in the list.
  - **Running:** The status is updated to *Running* when the job exits the queue on the server and actually starts executing.

  - **Complete:** The status changes to *Complete* when the execution ends. Once it is completed, the client (TexMol) automatically retrieves all required files/outputs from the server, stores them.
  - **Available:** The job status is changed to *Available* if all the required output is successfully retrived frm the server and stored. The results of a job can be viewed when it is in *Available* state.
  - **Error:** Indicates that there was an error on the server side. If the error happens due to connection problems, then it would be back to other non-error status, once the connection working again. However, if the status remains *Error* for an extended period of time, then it means that there must be something wrong with the input which is preventing F2Dock server to process the job properly. In such cases, the user is requested to recheck the inputs and resubmit.

- **Files:** Gives the name (and path) of the receptor pdb or pqr. This is mentioned only to provide a easy for the user.

The buttons on this interface are explained below-

- **Load/Save:** Jobs can be saved anytime irrespective of the state and loaded in the future. Closing a job would save it and remove it from the list. All these files are stored in the current directory. Once a saved job is loaded, the status of the job dictates which functions is available for it and whether TexMol would poll the server for updates. So, in effect users can simple submit a job, save it and close TexMol. They can launch TexMol in the future, load the job and see the results (if the server has processed the job already).

- **Job submission:** There are four buttons for submitting the four types of jobs. Each button launches the corresponding dialog for preparing parameters and submitting jobs. Each of these are described in separate sections, for example submitting docking jobs is described in Section 5.2.2.

- **Dislay Result:** This button is only active when an *Available* job is selected from the list. Once clicked, the result UI is launched which enables the user to see the summary of the results (Section 5.2.3), the details about each candidate solution (Section 5.2.4) and visualize the conformations (Section 5.2.5).

5.2.2 Prepare and submit a docking job

Figure 5.8 shows the first tab of the docking UI. Only the basic and essential options are exposed here. The users can select two pdbs and submit it immediately with default values for other parameters. Intermediate input files (pqr, f2d, rawn, quad) are autogenerated on the server. Note that, if an rmsd file is specified, f2dock can compute and report the rmsd values and identify the number of hits on its own.

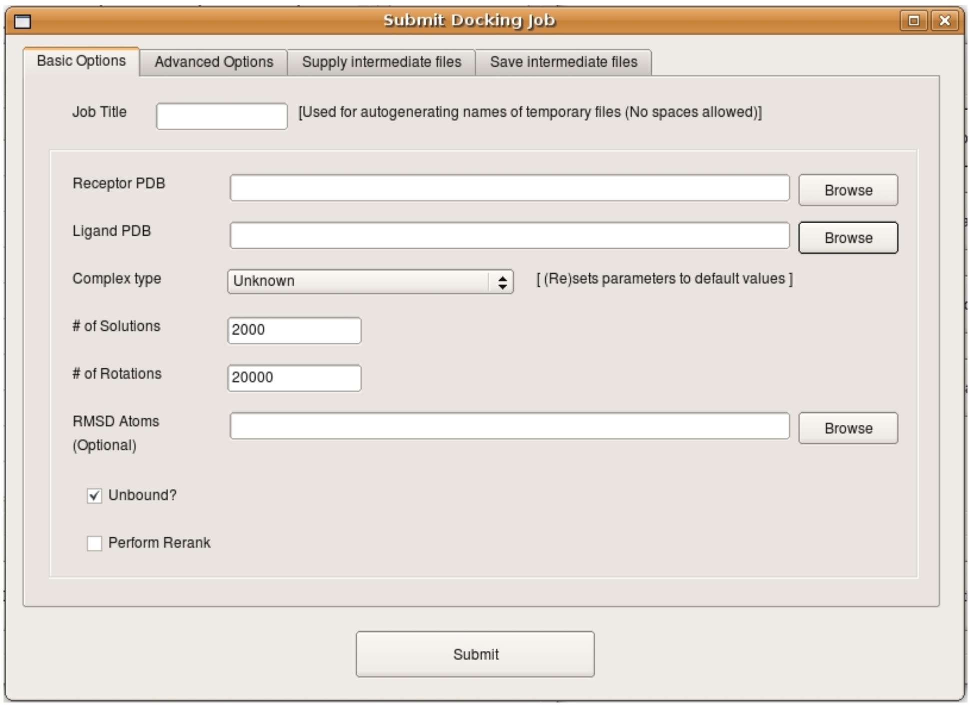
The remaining three tabs provide more advanced options for finetuning to parameters and also for making the jobs more efficient.

If the user intends to use the same pdb files for multiple docking jobs, then it is better to generate the intermediate files once and use them later to reduce the computation time.

In Figure 5.9, the third and fourth tabs are shown- which can be used to pass and save intermediate files respectively. If intermediate files are specified, then the server does not need to reproduce them and hence it saves time. This is specially useful if the user intends to run several docking jobs with different parameter values for the same complex. However, the user must be careful to ensure that the files are consistent with each other. Note that, the fourth tab is used for specifying which of in intermediate files (either user specified or generated at the server) need to be stored/retrieved once the docking job completes. These files are stored using the same path as the given pdbs and only the file type extension is modified.

Finally, the second tab (Figure 5.10) exposes some advanced parameters (see Section 5.1.3 for details about these parameters).

Note that sometimes it becomes tedious to select/change the values of parameters one by one. So, there is also an option to upload a user-defined .inp file directly.


Submit Docking Job

Basic Options | Advanced Options | Supply intermediate files | Save intermediate files

Job Title: [Used for autogenerating names of temporary files (No spaces allowed)]

Receptor PDB: Browse

Ligand PDB: Browse

Complex type: Unknown [(Re)sets parameters to default values]

# of Solutions: 2000

# of Rotations: 20000

RMSD Atoms (Optional): Browse

[✓] Unbound?

[ ] Perform Rerank

Submit

Figure 5.8: Basic Options for docking

5.2.3 Summarizing the result

Once the results are available, they are displayed in the UI shown in Figure 5.11. The table on the left lists all the results by mentioning the rank, score and rmsd (if available). If it was a reranking job, then the new rank and new score are also shown.

On the right panel, the summary of the result is displayed. It reports the number of hits in different ranges, the best hit, the topranked hit and the time.

5.2.4 Analyzing each conformation

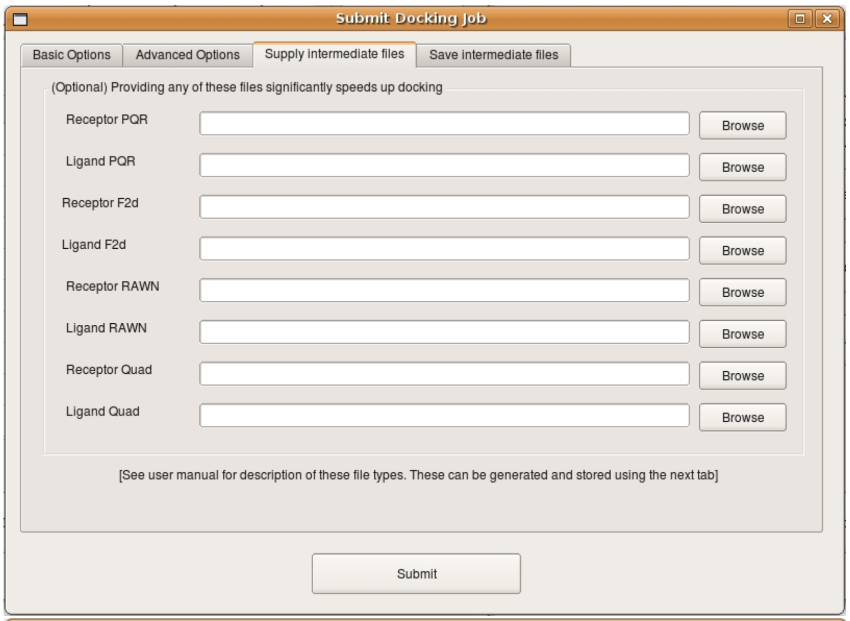
Individual results can be selected from the list on the left to see details about that result. These details are displayed in the tab shown in Figure 5.12. The details include breakdown of the scores of different scoring terms and filters and the transformation. The selected result can be stored as a transformed pdb of the ligand. This is useful if the user is interested in doing further informatics like binding affinity calculations, area/volume estimation etc.

Interface statistics including interface area, interface planarity, interface distance/volume etc., statistics of residue-residue contact on the interface, presence of hydrophilic voids between interfaces etc. are detailed in the interface details tab.

5.2.5 Visualizing each conformation

Selected conformations are immediately rendered on TexMol's main interface. By default, all the selected conformations are displayed together in Union of balls mode of rendering. However, the user can control the rendering by changing several options from the visualization parameters tab.

Specifically, the user may select to render in either union of balls (NURBS based rendering) mode (Figure 5.13(a)), or the smooth molecular surface (Figure 5.13(b)). If the user selects the smooth molecular surface, then there is further options to color the surface by different potentials. The user is also able to select whether only one ligand position is displayed at a time, or several is displayed at the same time (which might be a better choice for comparison purposes).


Submit Docking Job

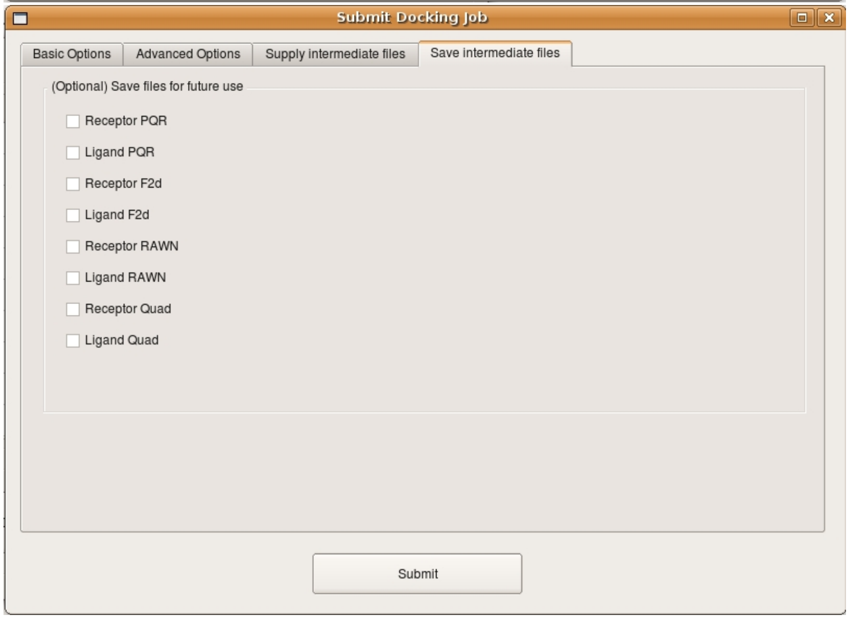
Basic Options | Advanced Options | Supply intermediate files | Save intermediate files

(Optional) Providing any of these files significantly speeds up docking

Receptor PQR [ ] Browse
Ligand PQR [ ] Browse
Receptor F2d [ ] Browse
Ligand F2d [ ] Browse
Receptor RAWN [ ] Browse
Ligand RAWN [ ] Browse
Receptor Quad [ ] Browse
Ligand Quad [ ] Browse

[See user manual for description of these file types. These can be generated and stored using the next tab]

Submit


Submit Docking Job

Basic Options | Advanced Options | Supply intermediate files | Save intermediate files

(Optional) Save files for future use

- [ ] Receptor PQR
- [ ] Ligand PQR
- [ ] Receptor F2d
- [ ] Ligand F2d
- [ ] Receptor RAWN
- [ ] Ligand RAWN
- [ ] Receptor Quad
- [ ] Ligand Quad

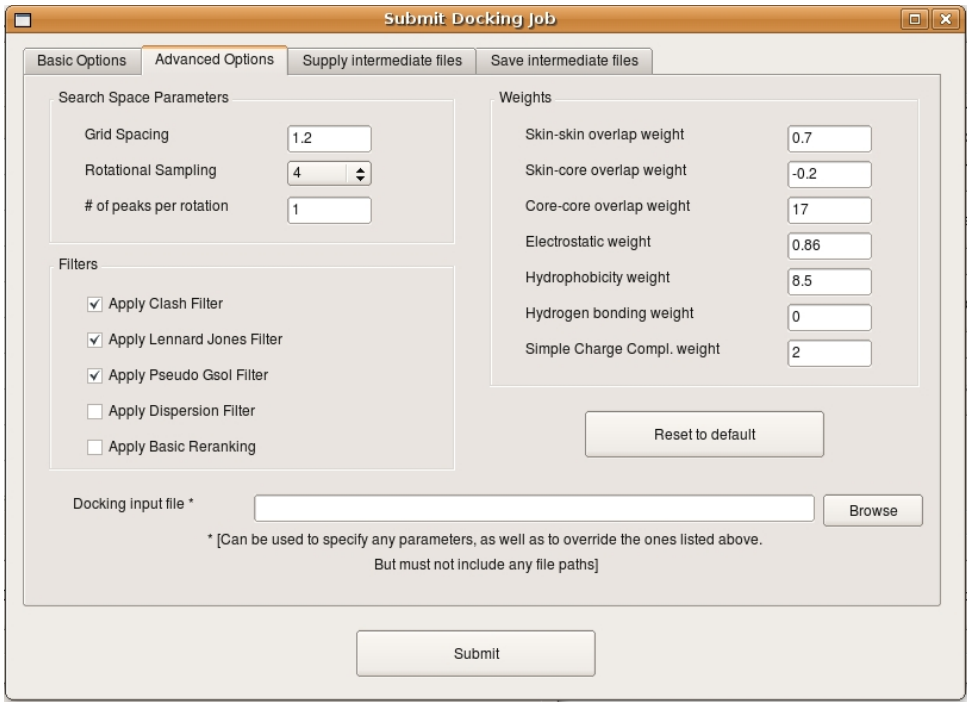
Submit

Figure 5.9: Intermediate file management. Top: passing intermediate files to the server. Bottom: choosing to store intermediate files so that they can be used in later docking runs

5.2.6 Performance tips

5.2.7 Command line version

TexMol also provides a command line interface which is useful to submit batch docking jobs to a remote server. The following command submits a remote docking job to the server.



Search Space Parameters

Grid Spacing	1.2
Rotational Sampling	4
# of peaks per rotation	1

Filters

- [✓] Apply Clash Filter
- [✓] Apply Lennard Jones Filter
- [✓] Apply Pseudo Gsol Filter
- [ ] Apply Dispersion Filter
- [ ] Apply Basic Reranking

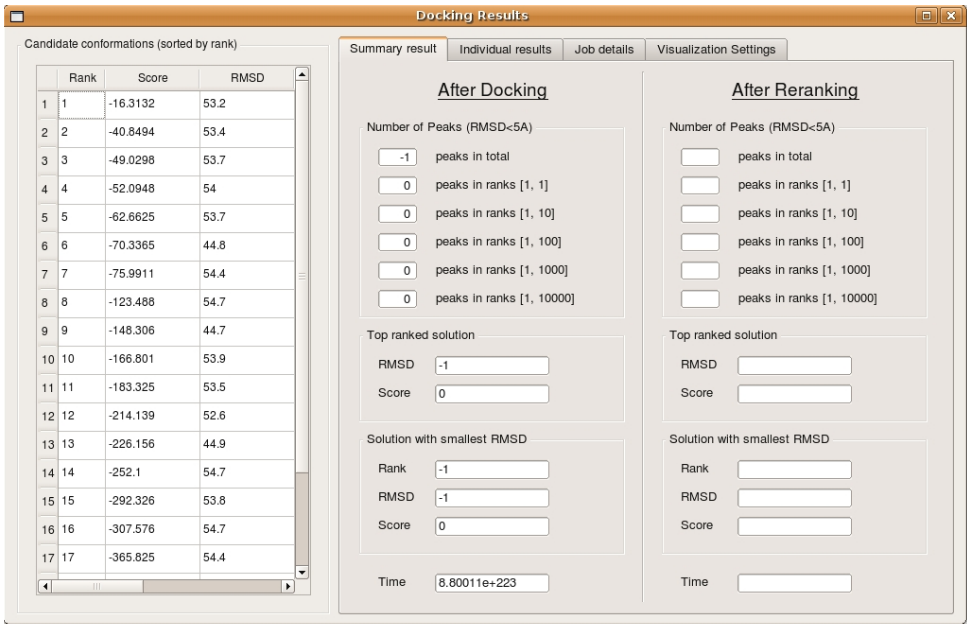
Weights

Skin-skin overlap weight	0.7
Skin-core overlap weight	-0.2
Core-core overlap weight	17
Electrostatic weight	0.86
Hydrophobicity weight	8.5
Hydrogen bonding weight	0
Simple Charge Compl. weight	2

Docking input file *

* [Can be used to specify any parameters, as well as to override the ones listed above.
But must not include any file paths]

Figure 5.10: Advanced Options for docking

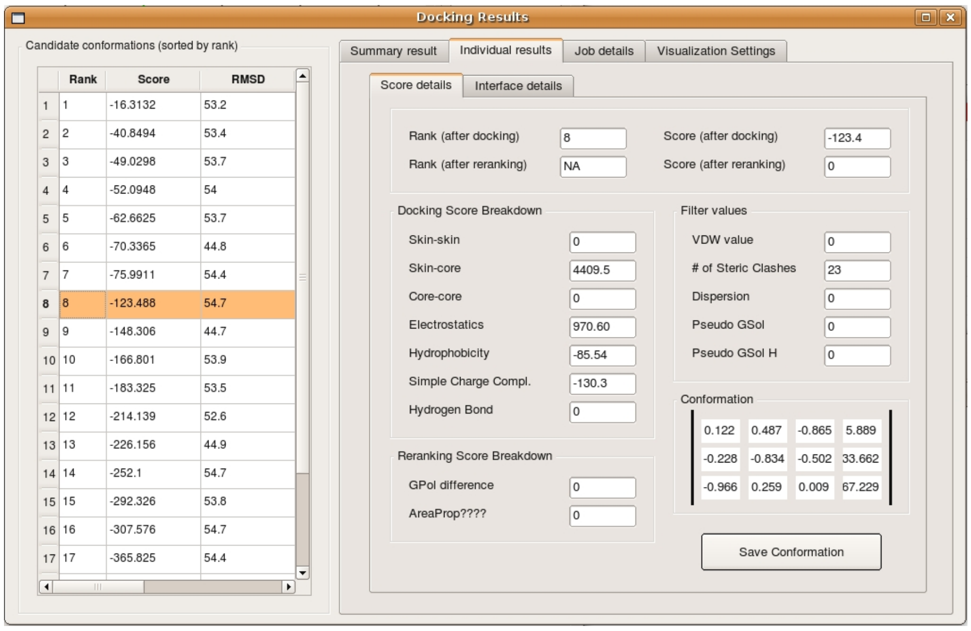


Candidate conformations (sorted by rank)

	Rank	Score	RMSD
1	1	-16.3132	53.2
2	2	-40.8494	53.4
3	3	-49.0298	53.7
4	4	-52.0948	54
5	5	-62.6625	53.7
6	6	-70.3365	44.8
7	7	-75.9911	54.4
8	8	-123.488	54.7
9	9	-148.306	44.7
10	10	-166.801	53.9
11	11	-183.325	53.5
12	12	-214.139	52.6
13	13	-226.156	44.9
14	14	-252.1	54.7
15	15	-292.326	53.8
16	16	-307.576	54.7
17	17	-365.825	54.4

	After Docking	After Reranking
Number of Peaks (RMSD<5A)		
peaks in total	-1	
peaks in ranks [1, 1]	0	
peaks in ranks [1, 10]	0	
peaks in ranks [1, 100]	0	
peaks in ranks [1, 1000]	0	
peaks in ranks [1, 10000]	0	
Top ranked solution		
RMSD	-1	
Score	0	
Solution with smallest RMSD		
Rank	-1	
RMSD	-1	
Score	0	
Time	8.80011e+223	

Figure 5.11: Summary of docking result



Candidate conformations (sorted by rank)

	Rank	Score	RMSD
1	1	-16.3132	53.2
2	2	-40.8494	53.4
3	3	-49.0298	53.7
4	4	-52.0948	54
5	5	-62.6625	53.7
6	6	-70.3365	44.8
7	7	-75.9911	54.4
8	8	-123.488	54.7
9	9	-148.306	44.7
10	10	-166.801	53.9
11	11	-183.325	53.5
12	12	-214.139	52.6
13	13	-226.156	44.9
14	14	-252.1	54.7
15	15	-292.326	53.8
16	16	-307.576	54.7
17	17	-365.825	54.4

Rank (after docking)	8	Score (after docking)	-123.4
Rank (after reranking)	NA	Score (after reranking)	0

Docking Score Breakdown

Skin-skin	0
Skin-core	4409.5
Core-core	0
Electrostatics	970.60
Hydrophobicity	-85.54
Simple Charge Compl.	-130.3
Hydrogen Bond	0

Reranking Score Breakdown

GPol difference	0
AreaProp????	0

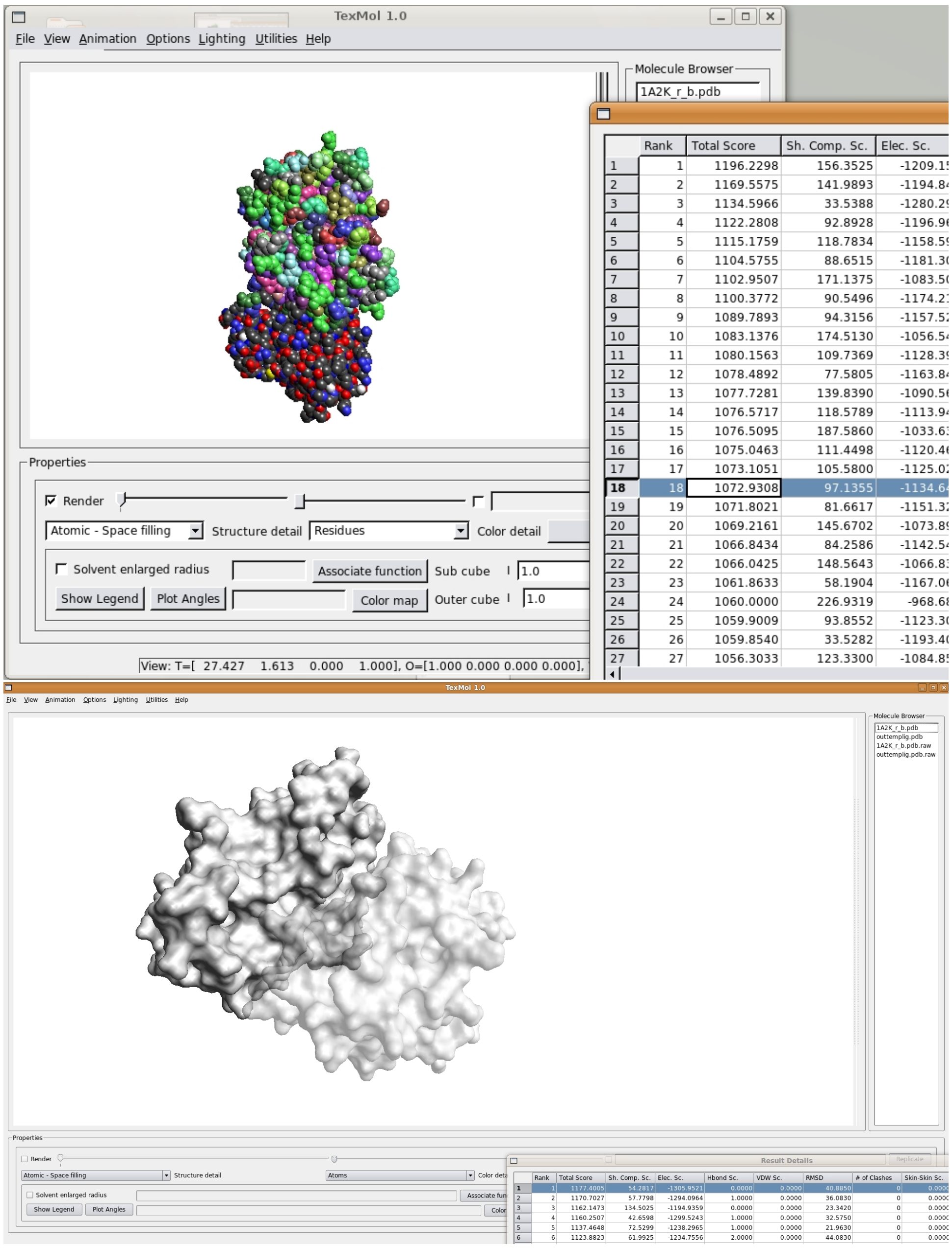
Filter values

VDW value	0
# of Steric Clashes	23
Dispersion	0
Pseudo GSol	0
Pseudo GSol H	0

Conformation

$$\begin{vmatrix} 0.122 & 0.487 & -0.865 & 5.889 \\ -0.228 & -0.834 & -0.502 & 33.662 \\ -0.966 & 0.259 & 0.009 & 67.229 \end{vmatrix}$$

Figure 5.12: Details of individual results



Molecule Browser: 1A2K_r_b.pdb

Atomic - Space filling | Structure detail: Residues | Solvent enlarged radius | Sub cube 1.0 | Outer cube 1.0

View: T=[27.427 1.613 0.000 1.000], O=[1.000 0.000 0.000 0.000],

	Rank	Total Score	Sh. Comp. Sc.	Elec. Sc.
1	1	1196.2298	156.3525	-1209.1
2	2	1169.5575	141.9893	-1194.8
3	3	1134.5966	33.5388	-1280.2
4	4	1122.2808	92.8928	-1196.9
5	5	1115.1759	118.7834	-1158.5
6	6	1104.5755	88.6515	-1181.3
7	7	1102.9507	171.1375	-1083.5
8	8	1100.3772	90.5496	-1174.2
9	9	1089.7893	94.3156	-1157.5
10	10	1083.1376	174.5130	-1056.5
11	11	1080.1563	109.7369	-1128.3
12	12	1078.4892	77.5805	-1163.8
13	13	1077.7281	139.8390	-1090.5
14	14	1076.5717	118.5789	-1113.9
15	15	1076.5095	187.5860	-1033.6
16	16	1075.0463	111.4498	-1120.4
17	17	1073.1051	105.5800	-1125.0
18	18	1072.9308	97.1355	-1134.6
19	19	1071.8021	81.6617	-1151.3
20	20	1069.2161	145.6702	-1073.8
21	21	1066.8434	84.2586	-1142.5
22	22	1066.0425	148.5643	-1066.8
23	23	1061.8633	58.1904	-1167.0
24	24	1060.0000	226.9319	-968.6
25	25	1059.9009	93.8552	-1123.3
26	26	1059.8540	33.5282	-1193.4
27	27	1056.3033	123.3300	-1084.8

Molecule Browser: 1A2K_r_b.pdb, outtemplig.pdb, 1A2K_r_b.pdb.raw, outtemplig.pdb.raw

Atomic - Space filling | Structure detail: Atoms | Solvent enlarged radius

Result Details

	Rank	Total Score	Sh. Comp. Sc.	Elec. Sc.	Hbond Sc.	VDW Sc.	RMSD	# of Clashes	Skin-Skin Sc.
1	1	1177.4005	54.2817	-1305.9521	0.0000	0.0000	40.8850	0	0.0000
2	2	1170.7027	57.7798	-1294.0964	1.0000	0.0000	36.0830	0	0.0000
3	3	1162.1473	134.5025	-1194.9359	0.0000	0.0000	23.3420	0	0.0000
4	4	1160.2507	42.6598	-1299.5243	1.0000	0.0000	32.5750	0	0.0000
5	5	1137.4648	72.5299	-1238.2965	1.0000	0.0000	21.9630	0	0.0000
6	6	1123.8823	61.9925	-1234.7556	2.0000	0.0000	44.0830	0	0.0000

Figure 5.13: Rendering selected conformations. (a) Union of Balls. (b) Smooth molecular surface.

```
./TexMol -dock <parameter file name>
```

Figure 5.14: Submitting remote docking job

Chapter 6

Reranking

Binding free energy based reranking (as explained in Section 2) is implemented in the GB-rerank software. Along with a command line interface suitable for batch processing, we have developed a user-friendly graphical user interface as part of TexMol. This chapter describes both of these in detail.

6.1 Reranking from the command line: GB-rerank

Once the GB-rerank executable is properly installed (see Section 8.2.2 for installation instructions), it can be executing simply by the following command-

This procedure takes in the output of a docking software and produces a reranked list of the conformations. The format of the parameter file, explanation of the parameters and the format and content of the output file are explained with examples in the next subsections.

Currently, GB-rerank only accepts docking output files which conform to the format of F2Dock's output file (see Section 5.1.4 for details of the format). However, we have developed scripts which can convert output files of other popular docking software into the format accepted by GB-rerank.

6.1.1 The parameter file

Similar to F2Dock, the parameter file of GB-rerank contains a sequence of lines. Each line specifies the value of a parameter. Each line starts by mentioning the name of the parameter and then provides a value for it.

A few parameters, like locations of input files (PDB/PQR, QUAD), and docking output file are mandatory and must be present in the parameter file. Other parameters are optional, and can be used to assign non-default values to internal variables.

6.1.2 List of mandatory parameters

- *staticMoleculePQR* (S): Name of the PQR file for the static molecule. Current version of F^{2}Dock only works on PQR files without chain information

- *staticMoleculeQUAD* (S): Name of the quadrature points file for the static molecule.

- *movingMoleculePQR* (S): Name of the PQR file for the moving molecule.

- *movingMoleculeQUAD* (S): Name of the quadrature points file for the moving molecule.

- *dockingOutputFile* (S): Name of the file to which F2Dock wrote the output containing the potential docking solutions (poses).

```
GB-rerank <paramFileName>
```

Figure 6.1: Executing GB-rerank

```
movingMoleculePQR data/PQR/1A2K_l_u.pqr
staticMoleculePQR data/PQR/1A2K_r_u.pqr
numRerank 2000
dockingOutputFile output/1A2K_dock.out
rerankingOutputFile output/1A2K_rerank.out
...
```

Figure 6.2: Example parameter file for GB-rerank

- *rerankingOutputFile* (S): Name of the file to which GB-rerank will write the reranked list of poses.

6.1.3 List of optional/advanced parameters

- *numRerank* (I): Number of top solutions to rerank. Default value is 2000.

- *rerankerF2DockScoreWeight* (F): Weight given to the original F^2Dock score. Default value is 100.0.

- *spectrum* (S): The spectrum parameter is a string of the form: "a1:b1-a2:b2-a3:b3..." which says that the top a1 results in the reranked output must not have original F2Dock rank larger than b1, the next a2 results must not have F2Dock rank larger than b2, and so on. By default this "spectrum" parameter is empty (i.e., not set).

6.1.4 Making sense of the output file

The output file of GB-rerank has exactly the same distinct parts as F2Dock (described in Section 5.1.4). However it adds four additional columns in the detailed list of conformations and sorts the conformations based on their new ranks.

The additional columns are-

- **new_rank:** rank of this docking result after GB based reranking
- **new_score:** new score computed from delgpol and original F2Dock score
- **delgpol:** reduction in polarization energy in kcal/mol
- **areaprop:** approximation to constant * surfaceArea

6.1.5 Scripts

6.2 Reranking via GUI: TexMol

TexMol provides a user-friendly interactive graphical interface to prepare, submit, manage reranking jobs and analyze the results. The dialogs can be launched by selecting 'Dock' from the utilities menu of TexMol.

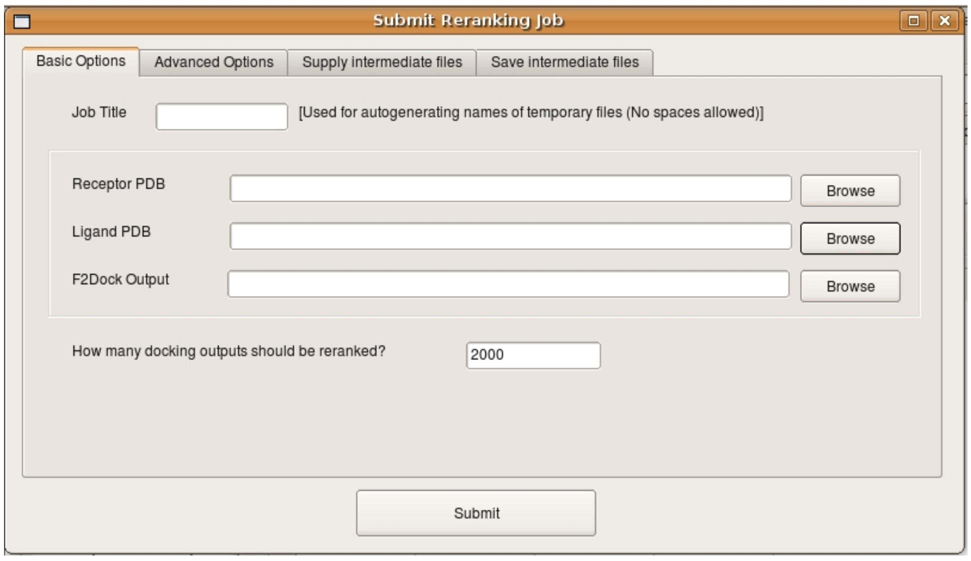
6.2.1 Job management

See Section 5.2.1

6.2.2 Prepare and submit a reranking job

Figure 5.8 shows the first tab of the reranking UI. Only the basic and essential options are exposed here. The users can select two pdbs, and a docking output file and submit it immediately with default values for other parameters. Intermediate input files (pqr, quad) are autogenerated on the server.

In Figure 6.4, the third and fourth tabs are shown- which can be used to pass and save intermediate files respectively. If intermediate files are specified, then the server does not need to reproduce them and



Job Title [Used for autogenerating names of temporary files (No spaces allowed)]

Receptor PDB

Ligand PDB

F2Dock Output

How many docking outputs should be reranked? 2000

Figure 6.3: Basic options for reranking

hence it saves time. This is specially useful if the user intends to run several docking jobs with different parameter values for the same complex. However, the user must be careful to ensure that the files are consistent with each other. Note that, the fourth tab is used for specifying which of in intermediate files (either user specified or generated at the server) need to be stored/retrieved once the docking job completes. These files are stored using the same path as the given pdbs and only the file type extension is modified.

Finally, the second tab (Figure 6.5) exposes the optional parameters (see Section 6.1.3 for details about these parameters).

6.2.3 Summarizing the result

See Section 5.2.3 for details about the summary results. If the result UI is launched for a reranking job, then on the right panel the number of hits in different ranges, the best hit, the topranked hit and the time before and after reranking is mentioned.

6.2.4 Analyzing each conformation

Same as presented in Section 5.2.4.

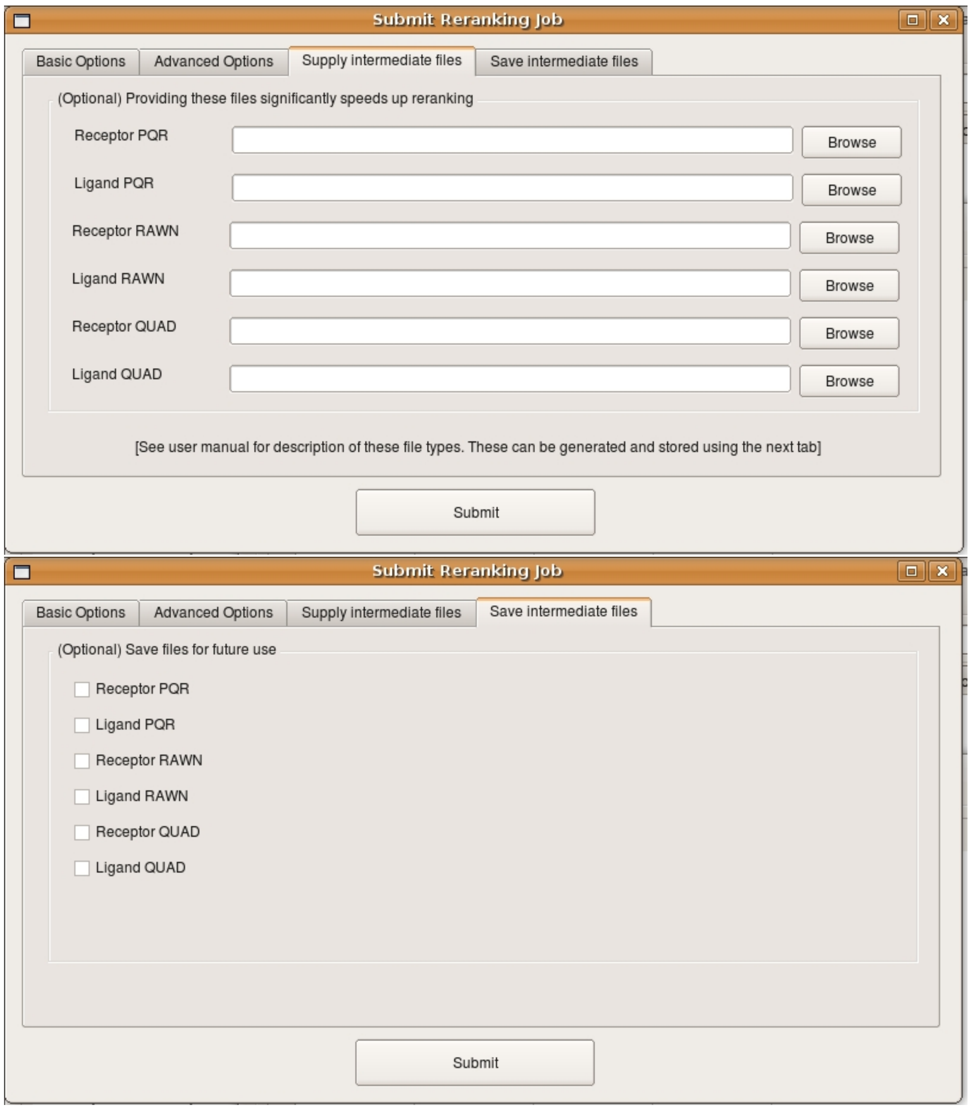
6.2.5 Visualizing each conformation

Same as presented in Section 5.2.4.

6.2.6 Performance tips

6.2.7 Command line version

TexMol also provides a command line interface which is useful to submit batch reranking jobs to a remote server. The following command submits a remote reranking job to the server.



Submit Reranking Job

Basic Options | Advanced Options | Supply intermediate files | Save intermediate files

(Optional) Providing these files significantly speeds up reranking

Receptor PQR [Browse]

Ligand PQR [Browse]

Receptor RAWN [Browse]

Ligand RAWN [Browse]

Receptor QUAD [Browse]

Ligand QUAD [Browse]

[See user manual for description of these file types. These can be generated and stored using the next tab]

Submit

Submit Reranking Job

Basic Options | Advanced Options | Supply intermediate files | Save intermediate files

(Optional) Save files for future use

☐ Receptor PQR

☐ Ligand PQR

☐ Receptor RAWN

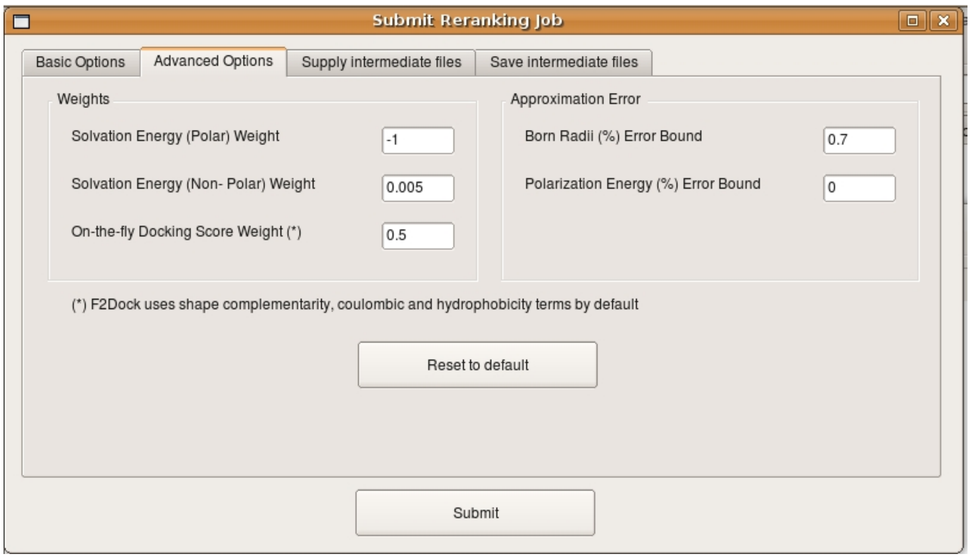
☐ Ligand RAWN

☐ Receptor QUAD

☐ Ligand QUAD

Submit

Figure 6.4: Intermediate file management



Solvation Energy (Polar) Weight: -1
Solvation Energy (Non- Polar) Weight: 0.005
On-the-fly Docking Score Weight (*): 0.5
Born Radii (%) Error Bound: 0.7
Polarization Energy (%) Error Bound: 0

(*) F2Dock uses shape complementarity, coulombic and hydrophobicity terms by default

Figure 6.5: Advanced Options for reranking

```
./TexMol -rerank <parameter file name>
```

Figure 6.6: Submitting remote reranking job

Chapter 7

Availability

7.1 Software

7.1.1 License

7.1.2 Download instructions

7.2 Dataset

7.2.1 License

7.2.2 Download instructions

Chapter 8

Installation

8.1 Requirements

8.1.1 Supported platforms

All software listed below has been built in Linux (Ubuntu 10.04.1) and Mac (OS X 10.5) operating systems.

8.1.2 Software dependencies

All the software listed below uses the CMake build system, `http://www.cmake.org/`. This is commonly part of many linux installations but may need to be downloaded and installed on other platforms.

MolSurf `MolSurf` requires the OpenGL libraries (due to some underlying data structures which are shared with *TeχMol*).

Additionally some optional components of `MolSurf` (which are not needed for *F^2Dock* depend on the CGAL library, `http://www.cgal.org/`.

TexMol Building *TeχMol* from source requires Qt (version 4, `http://qt.nokia.com/products/`), the boost libraries (`http://www.boost.org/`), OpenGL libraries, and Cg (NVIDIAs C for Graphics toolkit, `http://developer.nvidia.com/page/cg_main.html`). Most of these are already part of a standard linux installation.

Some optional components for performing energy computations in *TeχMol* have additional dependencies, NFFT (non-uniform fast Fourier transform, `http://www-user.tu-chemnitz.de/~potts/nfft/`), and PETSc (`http://www.mcs.anl.gov/petsc/petsc-as/`). These components are not necessary for running *F^2Dock*.

F2Dock Server Building the *F^2Dock* server from source depends upon the FFTW library, `http://www.fftw.org/`.

8.2 Install

8.2.1 F2Dock

8.2.2 GB-rerank

8.2.3 MolSurf

8.2.4 TexMol

8.2.5 F2Dock Server

Chapter 9

Comments

9.1 Known issues

9.2 Work in progress

Bibliography

[1] Antibody-antigen contacts. http://www.bioinf.org.uk/abs/allContacts.html.

[2] C. Bajaj, R. Chowdhury, and M. Rasheed. A dynamic data structure for flexible molecular maintenance and informatics. Technical Report TR-10-31, ICES, UT Austin, July 2010.

[3] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. TexMol: Interactive visual exploration of large flexible multi-component molecular complexes. In *Proc. of the Annual IEEE Visualization Conference*, pages 243–250, Austin, Texas, 2004.

[4] Chandrajit Bajaj, Rezaul A. Chowdhury, and Muhibur Rasheed. A dynamic data structure for flexible molecular maintenance and informatics (accepted). *Bioinformatics*, 2010. Preliminary version appeared in the *Proceedings of the ACM Symposium on Solid and Physical Modeling, 2009.*

[5] Chandrajit Bajaj, Rezaul A. Chowdhury, and Vinay Siddavanahalli. F2Dock: Fast Fourier Protein-Protein Docking (Accepted). *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2009.

[6] Chandrajit Bajaj and Wenqi Zhao. Fast molecular solvation energetics and forces computation. *SIAM Journal on Scientific Computing*, 31(6):4524–4552, 2010.

[7] Chandrajit Bajaj and Wenqi Zhao. Fast molecular solvation energetics and forces computation. *SIAM Journal on Scientific Computing*, 31(6):4524–4552, 2010.

[8] S.D. Black and D.R. Mould. Development of hydrophobicity parameters to analyze proteins which bear post- or cotranslational modifications. *Anal. Biochem.*, 193:72–82, 1991.

[9] Julio Castrillon-Candas, Vinay Siddavanahalli, and Chandrajit Bajaj. Nonequispaced fourier transforms for protein-protein docking. ICES Report 05-44, The University of Texas at Austin, Austin TX USA, October 2005.

[10] Rong Chen, Li Li, and Zhiping Weng. Zdock: An initial-stage protein-docking algorithm. *Proteins: Structure, Function, and Genetics, Special Issue: CAPRI - Critical Assessment of PRedicted Interactions . Issue Edited by Jol Janin*, 52(1):80–87, May 2003.

[11] Rong Chen and Zhiping Weng. A novel shape complementarity scoring function for protein-protein docking. *Proteins: Structure, Function, and Genetics*, 51(3):397–408, March 2003.

[12] Rezaul Chowdhury, Donald Keidel, Maysam Moussalem, Muhibur Rasheed, Arthur Olson2, Michel Sanner, and Chandrajit Bajaj. Protein-protein docking with F2Dock 2.0 and GB-rerank. *Submitted to Journal*, 2011.

[13] Rezaul Alam Chowdhury and Chandrajit Bajaj. Algorithms for faster molecular energetics, forces and interfaces. ICES report 10-32, Institute for Computational Engineering & Science, The University of Texas at Austin, Austin, TX, USA 78712., August 2010.

[14] T. Dolinsky, J. Nielsen, J.A. McCammon, and N. Baker. Pdb2pqr: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations. *Nucleic Acids Research*, 32:665–667, 2004.

[15] D. Eisenberg and A. Mclachlan. Solvation energy in protein folding and binding. *Nature (London)*, 319:199–203, 1986.

[16] Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.

[17] H. A. Gabb, R. M. Jackson, and M. J. E. Sternberg. Modelling protein docking using shape complementarity,electrostatics and biochemical information. *Journal of Molecular Biology*, 272(1):106–120, 1997.

[18] M. Gilson, M. Davis, B. Luty, and J.A. McCammon. Computation of electrostatic forces on solvated molecules using the Poisson-Boltzmann equation. *J. Phys. Chem.*, 97:3591–3600, 1993.

[19] Fabian Glaser, David M. Steinberg, Ilya A. Vakser, and Nir Ben-Tal. Residue frequencies and pairing preferences at proteinprotein interfaces. *PROTEINS: Structure, Function, and Genetics*, 43:89–102, 2001.

[20] R. Hermann. Theory of hydrophobic bonding. II. Correlation of hydrocarbon solubility in water with solvent cavity surface area. *J. Phys. Chem.*, 76:2754–2759, 1972.

[21] Chris L. Jackins and Steven L. Tanimoto. Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing*, 14(3):249–270, 1980.

[22] Fan Jianga and Sung-Hou Kim. "soft docking": Matching of molecular surface cubes. *Journal of Molecular Biology*, 219(1):79–102, May 1991.

[23] Susan Jones and Janet M. Thornton. Principles of protein-protein interactions. *Proceedings of the National Academy of Sciences of the United States of America*, 93(1):13–20, 1996.

[24] Susan Jones and Janet M. Thornton. Analysis of protein-protein interaction sites using surface patches. *Journal of Molecular Biology*, 272(1):121–132, 1997.

[25] L. Kapcha and P. Rossky. Personal Communication, University of Texas at Austin.

[26] Ephraim Katchalski-Katzir, Isaac Shariv, Miriam Eisenstein, Asher A. Friesem, Claude Aflalo, and Ilya A. Vakser. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences of the United States of America*, 89(6):2195–2199, 1992.

[27] Ephraim Katchalski-Katzir, Isaac Shariv, Miriam Eisenstein, Asher A. Friesem, Claude Aflalo, and Ilya A. Vakser. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences of the United States of America*, 89(6):2195–2199, March 1992.

[28] M. Levitt, M. Hirshberg, R. Sharon, and V. Daggett. Potential energy function and parameters for simulations of the molecular dynamics of proteins and nucleic acids in solution. *Comp. Phys. Comm.*, 91:215–231, 1995.

[29] R.M. MacCallum, A.C.R. Martin, and J.M. Thornton. Antibody-antigen interactions: contact analysis and binding site topography. *Journal of Molecular Biology*, 262(5):732–745, 1996.

[30] Jeffrey G. Mandell1, Victoria A. Roberts, Michael E. Pique, Vladimir Kotlovyi, Julie C. Mitchell, Erik Nelson, Igor Tsigelny, and Lynn F. Ten Eyck. Protein docking using continuum electrostatics and geometric fit. *Protein Engineering*, 14(2):105–113, February 2001.

[31] Julian Mintseris, Kevin Wiehe, Brian Pierce, Robert Anderson, Rong Chen, J. Janin, and Zhiping Weng. Protein-protein docking benchmark 2.0: An update. *Proteins: Structure, Function, and Bioinformatics*, 60(2):214–216, 2005.

[32] Julie C. Mitchell. Personal Communication, University of Wisconsin - Madison.

[33] Julie C. Mitchell. Sampling rotation groups by successive orthogonal images. *SIAM Journal on Scientific Computing*, 30(1):525–547, 2008.

[34] Garrett M. Morris, David S. Goodsell, Ruth Huey, and Arthur J. Olson. Distributed automated docking of flexible ligands to proteins: Parallel applications of AutoDock 2.4. *Journal of Computer-Aided Molecular Design*, 10:293–304, 1996. 10.1007/BF00124499.

[35] C. W. Mortensen, R. Pagh, and M. Pătraçcu. On dynamic range reporting in one dimension. In *STOC '05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 104–111, 2005.

[36] P. Rasmus and R. Flemming. Cuckoo hashing. *Journal of Algorithms*, 51(2), 2004.

[37] K. Sharp. Incorporating solvent and ion screening into molecular dynamics using the finite-difference Poisson-Boltzmann method. *J. Comput. Chem.*, 12:454–468, 1991.

[38] T. Simonson and A. Bruenger. Solvation free energies estimated from macroscopic continuum theory: An accuracy assessment. *J. Phys. Chem.*, 98:4683 – 4694, 1994.

[39] W. Still, A. Tempczyk, R. Hawley, and T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *J. Am. Chem. Soc.*, 112:6127–6129, 1990.

[40] J. Wagoner and N.A. Baker. Assessing implicit models for nonpolar mean solvation forces: The importance of dispersion and volume terms. 103:8331–8336, 2006.

[41] Huajun Wang. Grid-search molecular accessible surface algorithm for solving the protein docking problem. *Journal of Computational Chemistry*, 12(6):746–750, 1991.

[42] Scott J. Weiner, Peter A. Kollman, David A. Case, U. Chandra Singh, Caterina Ghio, Guliano Alagona, Salvatore Profeta, and Paul Weiner. A new force field for molecular mechanical simulation of nucleic acids and proteins. *Journal of the American Chemical Society*, 106:765–784, 1984.

[43] B.X. Yan and Y.Q. Sun. Glycine residues provide flexibility for enzyme active sites. *Journal of Biological Chemistry*, 272(6):3190, 1997.