Eurographics Workshop on 3D Object Retrieval (2016)
A. Ferreira, A. Giachetti, and D. Giorgi (Editors)

An Experimental Shape Matching Approach for Protein Docking

F. Fernandes[1,2] and A. Ferreira[1,3]

[1]Visualization and Intelligent Multimodal Interfaces Group, INESC-ID, Lisbon, Portugal
[2]Protein Modelling Laboratory, ITQB, Oeiras, Portugal
[3]Instituto Superior Técnico, Universidade Técnica de Lisboa, Lisbon, Portugal

Abstract

Proteins play a vital role in biological processes, with their function being largely determined by their structure. It is important to know what a protein binds, where it binds, how it binds, and what is its final conformation. Several methodologies have been applied to solve this complex protein-protein docking problem, but the number of degrees of freedom renders this a very slow and computationally heavy challenge. To handle this problem, we propose a multi-level space partition approach to describe the three-dimensional shape of the protein. By combining two proteins in the same data structure we are able to easily detect the shape-complementary regions. Moreover, by directly integrating bio-energetic information, we can drive the algorithm by both parameters and provide a fast and efficient way to overcome some of the limitations of previous approaches.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

Proteins are biological macromolecules represented by a sequence of amino acid residues, which naturally folds and adopts a characteristic three-dimensional conformation. Many biological mechanisms are based on the interaction of two or more protein individuals, which are experimentally difficult to characterize at the molecular and atomic detail, which is why it is very important to understand, characterize, represent and reproduce these events computationally.

Molecular Mechanics and Molecular Dynamics (MD) [Lea01] are fundamental tools in Structural Bioinformatics that aim at simulating in-silico as best as possible all the movements, interactions, and forces that govern the biomolecular behavior at the atomic level. However, the detail found in these methods makes them very computationally expensive, which renders ab-initio, fully-flexible protein docking a very complex problem.

As a consequence, rigid-body approximations have been developed [Rit08] only considering the protein shape represented by its solvent-accessible surface area. But this too poses some challenges due to the multiple possible rotations, translations and orientations that the two docking partners can exhibit.

Besides, shape information is not always sufficient to determine if two proteins bind with each other, since they can be compatible at a shape level, but the intermolecular forces at the contact interface make them repel or have no affinity with each other, thus invalidating their docking.

To overcome these problems, we centered our approach in a data structure based on a modification and extension of the octree spatial index [Mea82], a tree-shaped data structure used for 3D spatial representation, where a cube volume is recursively subdivided into eight smaller cubes, allowing for a straightforward 3D multi-resolution decomposition. But instead of this binary division over each axis, we consider overlapped and successively-decreasing cubes centered at each one of the voxels of the object.

Our most relevant addition to this is the fact that we label each tree node by both a rotation-invariant representation of shape and by an energetic/biochemical score. This allows to improve the docking process by efficiently guiding it by both criteria simultaneously.

2. Related work

The most successful tools for rigid protein docking include approaches based on a global search over the entire conformational space using 3D FFT correlations [KKSE*92], such as ZDOCK [CLW03] or PIPER [KBCV06], or geometrical hashing [WR97], such as PatchDock [SDINW05]. These involve long calculations over thousands of coefficients that need to be calculated over thousands of incremental orientations and the quality of the representations decays with the increasing size of the protein. Other approaches, including RosettaDock [GMW*03] and AutoDock [MHL*09], consider random initial structures over a limited region and perform multiple Monte Carlo minimization steps. These meth-

DOI: 10.2312/3dor.20161083

ods may take a long time to reach the most favorable conformation and it is not guaranteed that it will be found in the available time span. An alternative method, applied in HADDOCK [DBB03], uses a priori information about interface residues and guides the search towards conformations that satisfy those restraints. But its success requires the availability beforehand of experimental information about the actual proteins in contact or about similar targets, which is not always the case.

3. Methods

We consider that the shape of each protein is delimited and identified by its *solvent-accessible surface area* (SAS) [LR71], i.e. the area at the surface of the protein that is susceptible of becoming in contact with surrounding solvent molecules or some other molecular entities.

To represent the shape of the protein, we introduce the concept of a *Multi-Level Shape Tree* (MLSTree), consisting of a tree-like data structure that basically stores voxelized representations of the protein's shape at different levels of detail. This concept is an extension of the one found in octrees, but with some major differences.

Octrees perform a disjoint partition of the space by following the distribution of a 2x2x2 cube and packing each set of 8 voxels at level l into a larger voxel at level $(l+1)$, resulting in the intrinsic property that no voxel at a certain level overlaps with another voxel of the same level. On the other hand, the MLSTree follows the distribution of a 3x3x3 cube and takes the 27 voxels centered at each voxel at level l to define the voxel at the following level $(l+1)$, thus introducing overlap into voxels at the same level. This overlap is highly desirable since a region of interest could be found cut by the splitting planes of a regular octree in every level, and this way we guarantee that, up from a certain level, that region is entirely contained in at least one voxel.

Furthermore, each voxel at level $(l+1)$ in the MLSTree is not represented by only its filled or empty state like in octrees, but instead by a rotation-invariant representation of its 3x3x3 shape at level l. In addition to this shape information, we also add to each tree node information about molecular interaction energy scores [RG07]. This gives us a measure of the binding strength between the two docked proteins, and takes into account several aspects such as molecular force fields, hydrophobic regions and the frequency of similar contact regions in previous experimental studies. This enables us to later access the biological binding affinity of the pairs of shape complementary regions of both proteins, i.e., regions at the surface of each protein whose volumes fit together perfectly or with only a residual amount of gaps and overlaps.

To perform the docking of two proteins, we build an unified MLSTree for the shape of one protein and the complementary/outside shape of the other. Matching is performed by doing a top-bottom search, from the broader levels to the narrower levels, and following tree nodes whose shape belongs to both proteins simultaneously while checking if the biological function scores are compatible.

3.1. MLSTree data structure

The SAS of the protein is snapped onto a grid and voxelized. We consider this to be the level 0 of the tree. For each voxel, we consider the 26 neighborhood voxels directly surrounding it and delimiting a 3x3x3 cube, and use this volume to define a larger voxel at the next level. In every voxel created this away, the center position is always filled. A voxel at level l, from now on called an l-voxel, at a generic position is therefore defined by:

$$l\text{-}voxel[x,y,z] = \bigcup_{i,j,k\in\{-1,0,+1\}} (l-1)\text{-}voxel[x+i,y+j,z+k] \quad (1)$$

This process is repeated starting at every 0-voxel and applied recursively, level by level, until we end up with a single filled voxel.

Intuitively, this is equivalent to representing the protein shape by a set of spheres of progressively increasing (by a factor of 3) radius, centered at every point of the surface, but resorting to cubes instead of actual spheres.

All of these l-voxels are then arranged in a tree-like structure, sorted by decreasing level. l-voxels at the same level and sharing the same shape are grouped in the same tree node. The tree starts with the root node at the top, whose shape is defined by a cube containing a single filled voxel at the center, and ends with the bottom leaf nodes, whose shapes are the 1-voxels.

Figure 1 exemplifies how the shapes of the first bottom levels are obtained in a simplified two-dimensional version of this MLSTree data structure.

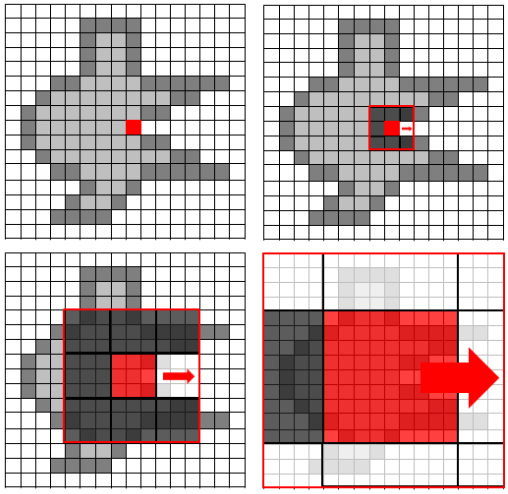

Figure 1: *An example of a simple 2D shape and the MLSTree levels 0 to 3 of one of its pixels. The full MLSTree would consider this for every pixel at the shape's boundary (pixels in dark grey).*

3.1.1. Rotational invariance

The tree nodes do not explicitly store the exact original voxel shapes, but instead a rotation-invariant version from the enumeration of all valid, normal-aligned, rotational-symmetries-free 3x3x3 shapes. Therefore, we do not consider all the possible 2^{27} combinations of 3x3x3 cubic shapes, but instead only a smaller subset after taking out all redundant rotations.

To produce this "normalized" version, first the normal vector $\vec{n}$

of the overall shape inside the 3x3x3 cube is calculated at its center voxel, considering the directional contributions of all the surrounding voxels. This way we get the shape's average outward direction. Then, we rotate the shape inside the cube so that the normal vector points upwards, aligned with the z-axis. Finally, for all the possible 8 rotations of this shape around the vertical z-axis, we calculate the sum of the 3D rectilinear distances of every filled voxel to the (1,1,1) corner of the cube, and then select the rotation with the larger distance. This corresponds to sorting the rotations and choosing the one that packs the highest density of filled voxels closely to the (-1,-1,-1) corner of the cube. These last steps are applied in order to eliminate the additional redundancy associated with the z-axis.

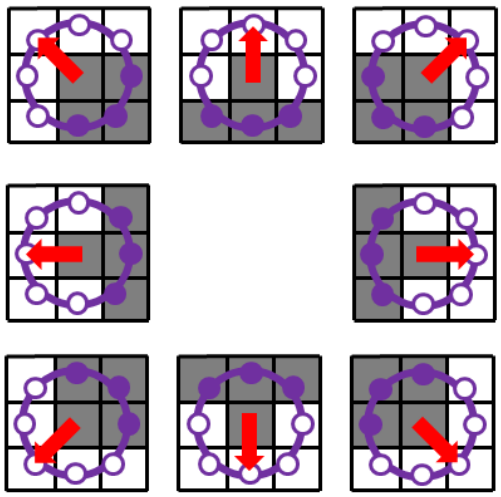

Figure 2: *All the 8 possible rotations of an example shape formed by 4 pixels around the central pixel in a 3x3 square. After vertically aligning all the shapes by their respective normal vectors, in red, all the shapes are reduced to a single one (top center).*

Figure 2 shows this rotational redundancy in the two-dimensional case. In a 3x3 square, there exist at most 8 possible rotations of a generic shape around the central pixel. In this lower dimension case, the alignment by the normal vector is sufficient and an additional last rotation step is not needed.

If at the same tree level we have l-voxels from different starting positions but with the same "normalized" shape, they are combined into the same tree node (but retaining their individual connections to lower levels), which is crucial in order to achieve rotational-invariance in the shape matching algorithm that will be explained later.

3.1.2. Intermolecular energy score

In addition to shape information, we also incorporate into the MLSTree an appropriate scoring for protein-protein interactions [KDFB04]. This includes a weighted combination of values about electrostatic, van der Waals and desolvation energies, hydrogen bonding and hydrophobicity. For further information about these biological scores, the interested reader is referred to [MTBFR13].

On higher levels, this energetic information is determined by calculating, for each component, the average of the scores present in the volumes of the preceding lower level. By adding this additional biochemical information to each tree node, we can guide the docking search by both shape and energetic criteria simultaneously.

3.1.3. Implementation and complexity

Each tree node stores three major elements:

1. A label consisting of a 3x3x3 cubic shape from the non-redundant rotations-free set of valid shapes, represented by an array of bits
2. A biochemical score including several components of the intermolecular energies inside the 3x3x3 volume, represented by an array of floating point values
3. And an array of (at most 27) links/pointers to the nodes of the tree level below. There can be several of these arrays when distinct l-voxels get collapsed into the same node.

The maximum number of levels L of the structure is given by $L = \lceil \log_3(max\{N_x, N_y, N_z\}) \rceil$, where N_ω is the dimension of axis ω in terms of 0-voxels, i.e. the size of the ω axis over the initial voxelized representation of the protein shape. The space requirement for the MLSTree is therefore $O(L \cdot n)$, where n is the total number of 0-voxels.

3.2. Matching algorithm

In general, for two proteins to bind in the docking process, they must contain at least one region where the shape of one is highly complementary to the shape of the other. Therefore, if we consider the "outside" shape of one of them, it will be highly similar to the regular shape of the other in those regions where the binding occurs. With this in mind, we exploited the MLSTree data structure to devise an algorithm that is able to find these similar shared regions.

We first build an MLSTree that includes the shape P_1 of the larger protein and the complementary shape $\overline{P_2}$ of the smaller protein. This complementarity step can be done by taking the initial voxelization of the protein and swapping empty and filled voxels, thus changing convex shapes into concave shapes and vice-versa.

While building this unified MLSTree, we keep track of which tree nodes belong to which protein. If both proteins share a common voxel shape at a certain level, this information gets collapsed into the same tree node and we mark it as belonging to both proteins. The fact that the tree nodes are labelled by a rotation-invariant version of the voxel shapes, allows the procedure to find the matching regions between both proteins independently of their initial orientations.

The search procedure is executed over the MLSTree in a top-down manner, following the nodes shared by both proteins and recursively collecting scores, for both the shape similarity and the biochemical docking affinity, from the shared nodes at the levels bellow. The pseudo-code for this procedure is presented in Algorithm 1.

Before starting the search we define both a threshold for the minimum shape similarity and for the minimum binding affinity. We

Algorithm 1 Detect shape complementarity regions between two proteins, P_1 and P_2

```
1: tree ← MLSTree(P1) ∪ MLSTree(P̄2)
2: procedure SHAPEMATCHING(tree)
3:     for level ← maxLevel, ..., minLevel do
4:         for all nodes in level do
5:             if (Shared(node) and NotReported(node)) then
6:                 if NotVisited(node) then
7:                     CollectScores(node)
8:                     MarkNodesBellowAsVisited(node)
9:                 end if
10:                if scores ≥ thresholds then
11:                    Report(node)
12:                    MarkNodesBellowAsReported(node)
13:                end if
14:            end if
15:        end for
16:    end for
17: end procedure
```

also set a limit for the minimum level of the tree to stop the search to prevent the algorithm from reporting shared regions with a very small area.

The "distance" between two nodes of the same level is defined as the number of voxels that are different between their shapes, which is equivalent to computing their Hamming distance, but weighted by the level that they are at, since higher levels correspond to a larger cubic volume. Therefore, for the same number of shape differences, distances at higher levels are larger than at lower ones

The biochemical energy scores are stored in the shared nodes separately for each protein and are compared and combined in order to address the level of biochemical binding compatibility between two already shape-complementary regions.

If the currently active node does not display a sufficient level of complementary in either one of the predefined thresholds for shape or for energy, the search is early interrupted and proceeds to the following node. The procedure returns a list of all matching cubic regions that satisfy both shape and energetic thresholds. The original orientations and the coordinates in both proteins for these accepted regions can be obtained by storing this information in the shared tree nodes while building the combined tree.

3.3. Implementation aspects

The voxels at the various levels of the tree can be represented using bit arrays that fit in even less space than that of a 32-bit computer word. This means that shape comparisons and differences while searching can be computed extremely fast using bit-level parallelism hardware instructions (AND, XOR, POPCNT, etc) present in all of today's CPUs. Besides, both the construction of the data structure and the search procedure can be easily adapted to take advantage of multi-threading.

The space requirements of this data structure, although being loglinear, can have a large constant factor due to the high degree of branching. But since close points at lower levels will most likely share the same representation on higher levels, we can take advantage of some basic compression techniques to reduce its space usage.

When building the tree, we can also use some level of sampling, e.g. take only every other 0-voxel, to reduce even further the memory usage. The tree construction can also be done in time $O(n \log_3 n)$ with the added advantage of the algorithm being easily parallelizable.

As for the matching algorithm, in the worst-case scenario, the search procedure will visit every node of the tree. Since the work performed for each node is constant and the total number of nodes is of order approximately $O(n \log_3 n)$, so is the time complexity of the algorithm. This time estimation can be somewhat improved because we only need to visit the smaller subset of nodes that are shared simultaneously by both proteins.

4. Conclusions and future work

We intended to efficiently tackle the problem of protein-protein docking by developing new algorithms and data structures. We took the concept of octrees and proposed an extended data structure which includes not only shape information but also biochemical scoring meta-data in a single combined structure.

The on-going practical implementation of these ideas will culminate in a useful software that will solve some of the disadvantages and shortcomings of previous approaches. This new support structure will allow fast searches and easy shape matching and indexing of large structures. By guiding the search through both geometric and energetic criteria simultaneously, we will be able to early discard improbable conformations that would otherwise only be detected in the later phases of the analysis by expensive MD calculations.

We are currently working on obtaining experimental results, as well as additional time and space benchmarks. The current version is single-threaded and running on the CPU, but later on we will also exploit the speed benefits of GPGPU programming and release a GPU-based version which will, among other optimizations, be able to process distinct branches of the tree in parallel.

The search phase of the algorithm can be adapted to allow suboptimal fitting and account for side chain flexibility. This can be accomplished by checking not only nodes with shared identical shapes, but also pairs of nodes whose shapes have an Hamming distance lower than a given threshold. Backbone flexibility should also be possible to achieve to some degree by detecting and combining contiguous or close parts of rigidly docked segments, and bending the final protein conformation accordingly.

By taking advantage of the developed data structure and integrating the data of multiple (more than two) proteins into a single search tree, we could later open the door to other practical applications involving large data sets of proteins, such as approximate query matching against a database, multiple structural alignment, detection of conserved regions among all the elements of a set, finding elements that bind to something inside a "positive set" but that do not bind to anything inside a "negative set", and several other open possibilities to explore.

5. Acknowledgments

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 and through the project A-MOP, with reference UTAP-EXPL/QEQ-COM/0019/2014. F.F. was additionally supported by FCT's individual post-doctoral grant with reference SFRH/BPD/111836/2015.

References

[CLW03] CHEN R., LI L., WENG Z.: Zdock: an initial-stage protein-docking algorithm. *Proteins: Structure, Function, and Bioinformatics 52*, 1 (2003), 80–87. 1

[DBB03] DOMINGUEZ C., BOELENS R., BONVIN A. M.: Haddock: a protein-protein docking approach based on biochemical or biophysical information. *Journal of the American Chemical Society 125*, 7 (2003), 1731–1737. 2

[GMW*03] GRAY J. J., MOUGHON S., WANG C., SCHUELER-FURMAN O., KUHLMAN B., ROHL C. A., BAKER D.: Protein–protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations. *Journal of molecular biology 331*, 1 (2003), 281–299. 1

[KBCV06] KOZAKOV D., BRENKE R., COMEAU S. R., VAJDA S.: Piper: an fft-based protein docking program with pairwise potentials. *Proteins: Structure, Function, and Bioinformatics 65*, 2 (2006), 392–406. 1

[KDFB04] KITCHEN D. B., DECORNEZ H., FURR J. R., BAJORATH J.: Docking and scoring in virtual screening for drug discovery: methods and applications. *Nature reviews Drug discovery 3*, 11 (2004), 935–949. 3

[KKSE*92] KATCHALSKI-KATZIR E., SHARIV I., EISENSTEIN M., FRIESEM A. A., AFLALO C., VAKSER I. A.: Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences 89*, 6 (1992), 2195–2199. 1

[Lea01] LEACH A. R.: *Molecular modelling: principles and applications*. Pearson education, 2001. 1

[LR71] LEE B., RICHARDS F. M.: The interpretation of protein structures: estimation of static accessibility. *Journal of molecular biology 55*, 3 (1971), 379–IN4. 2

[Mea82] MEAGHER D.: Geometric modeling using octree encoding. *Computer graphics and image processing 19*, 2 (1982), 129–147. 1

[MHL*09] MORRIS G. M., HUEY R., LINDSTROM W., SANNER M. F., BELEW R. K., GOODSELL D. S., OLSON A. J.: Autodock4 and autodocktools4: Automated docking with selective receptor flexibility. *Journal of computational chemistry 30*, 16 (2009), 2785–2791. 1

[MTBFR13] MOAL I. H., TORCHALA M., BATES P. A., FERNÁNDEZ-RECIO J.: The scoring of poses in protein-protein docking: current capabilities and future directions. *BMC bioinformatics 14*, 1 (2013), 286. 3

[RG07] RAJAMANI R., GOOD A. C.: Ranking poses in structure-based lead discovery and optimization: current trends in scoring function development. *Current opinion in drug discovery & development 10*, 3 (2007), 308–315. 2

[Rit08] RITCHIE D. W.: Recent progress and future directions in protein-protein docking. *Current protein and peptide science 9*, 1 (2008), 1–15. 1

[SDINW05] SCHNEIDMAN-DUHOVNY D., INBAR Y., NUSSINOV R., WOLFSON H. J.: Geometry-based flexible and symmetric protein docking. *Proteins: Structure, Function, and Bioinformatics 60*, 2 (2005), 224–231. 1

[WR97] WOLFSON H. J., RIGOUTSOS I.: Geometric hashing: An overview. *Computing in Science & Engineering*, 4 (1997), 10–21. 1