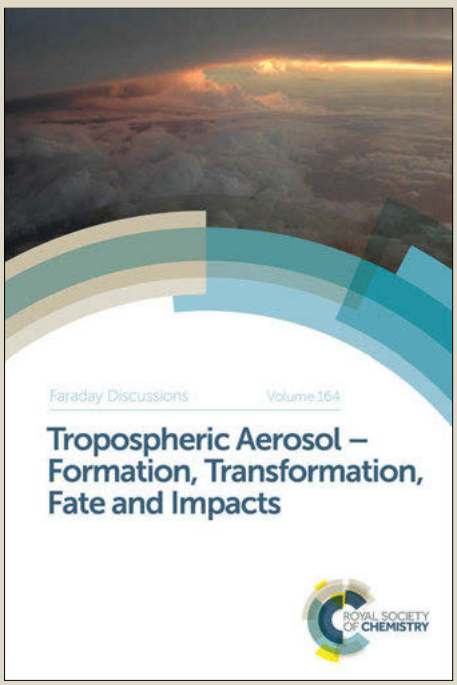
Faraday Discussions

Accepted Manuscript

This manuscript will be presented and discussed at a forthcoming Faraday Discussion meeting. All delegates can contribute to the discussion which will be included in the final volume.

Register now to attend! Full details of all upcoming meetings: http://rsc.li/fd-upcoming-meetings



This is an *Accepted Manuscript*, which has been through the Royal Society of Chemistry peer review process and has been accepted for publication.

Accepted Manuscripts are published online shortly after acceptance, before technical editing, formatting and proof reading. Using this free service, authors can make their results available to the community, in citable form, before we publish the edited article. We will replace this *Accepted Manuscript* with the edited and formatted *Advance Article* as soon as it is available.

You can find more information about *Accepted Manuscripts* in the Information for Authors.

Please note that technical editing may introduce minor changes to the text and/or graphics, which may alter content. The journal's standard Terms & Conditions and the Ethical guidelines still apply. In no event shall the Royal Society of Chemistry be held responsible for any errors or omissions in this *Accepted Manuscript* or any consequences arising from the use of any information it contains.

ROYAL SOCIETY OF CHEMISTRY

www.rsc.org/faraday_d

CREATED USING THE RSC LaTeX PCCP ARTICLE TEMPLATE – SEE www.rsc.org/electronicfiles FOR DETAILS
PAPER www. rsc.org/faraday_d | Faraday Discussions

A real-time proximity querying algorithm for haptic-based molecular docking

Georgios Iakovou,[a] Steven Hayward,[a] and Stephen Laycock*[a]

Received Xth XXXXXXXXXX 20XX, Accepted Xth XXXXXXXXX 20XX
First published on the web Xth XXXXXXXXXX 200X
DOI: 10.1039/c000000x

Intermolecular binding underlies various metabolic and regulatory processes of the cell, and the therapeutic and pharmacological properties of drugs. Molecular docking systems model and simulate these interactions *in silico* and allow us to study the binding process. Haptic-based docking provides an immersive virtual docking environment where the user can interact with and guide the molecules to their binding pose. Moreover, it allows human perception, intuition and knowledge to assist and accelerate the docking process, and reduces incorrect binding poses. Crucial for interactive docking is the real-time calculation of interaction forces. For smooth and accurate haptic exploration and manipulation, force-feedback cues have to be updated at a rate of 1 kHz. Hence, force calculations must be performed within 1ms. To achieve this, modern haptic-based docking approaches often utilize pre-computed force grids and linear interpolation. However, such grids are time-consuming to pre-compute (especially for large molecules), memory hungry, can induce rough force transitions at cell boundaries and cannot be applied to flexible docking. Here we propose an efficient proximity querying method for computing intermolecular forces in real time. Our motivation is the eventual development of a haptic-based docking solution that can model molecular flexibility. Uniquely in a haptics application we use octrees to decompose the 3D search space in order to identify the set of interacting atoms within a cut-off distance. Force calculations are then performed on this set in real time. The implementation constructs the trees dynamically, and computes the interaction forces of large molecular structures (i.e. consisting of thousands of atoms) within haptic refresh rates. We have implemented this method in an immersive, haptic-based, rigid-body, molecular docking application called Haptimol_RD. The user can use the haptic device to orientate the molecules in space, sense the interaction forces on the device, and guide the molecules to their binding pose. Haptimol_RD is designed to run on consumer level hardware, i.e. there is no need for specialized/proprietary hardware.

1 Introduction

Intermolecular complex formation underlies virtually all the metabolic and regulatory processes of the cell, as well as the therapeutic effects and pharmacologi-

[a] *School of Computing Sciences, University of East Anglia, Norwich, Norfolk, NR4 7TJ, UK. Fax: +44 (0) 1603 593345; Tel: +44(0) 1603 593795; E-mail: s.laycock@uea.ac.uk*

cal properties of drugs. For the past 40 years, scientists have been studying such bindings and relied on experimental (*in vitro*) work and computational methods (*in silico*) to study, model, and replicate them. *Molecular docking* refers to the computational methods devised and employed by researchers and field practitioners in order to simulate (as accurately as possible) this natural process. The ultimate goal of docking is to fit two molecules (often referred to as receptor and ligand molecules) together in a viable configuration based on their topographic and physiochemical properties.

There are two main approaches to docking, one automated the other interactive. Automated methods[1–3] utilize sophisticated, pose selection algorithms and rely only on computer power to carry them through. Conversely, interactive methods[4,5] allow human intervention, and their performance depends closely on human intuition, knowledge and expertise. Automated docking solutions comprise the majority of the applications available in the field. Nonetheless, they are time consuming (solutions to docking problems can take several hours)[4,6], often predict incorrect docking conformations[7,8], and by their very nature are not able to benefit from human knowledge and intuition. Haptic devices[9] and interactive molecular visualization systems[5,10–13] address these issues by transferring some of the complexity of the molecular binding process from the computer to the human. Haptic-based docking systems simulate the docking process in a 3D virtual environment, where the user interacts with the virtual molecules and performs a knowledge-guided search and selection of the final binding pose. They provide an immersive virtual docking environment where the user can sense (via a haptic device) the interaction forces and guide the molecules to their binding pose. They also establish a learning environment for the study of the docking process, and of the underlying interactions. It has been shown that such docking systems allow human perception, intuition and knowledge to assist and accelerate the docking process, and reduce incorrect binding poses[14].

Other applications of haptics in biomolecular science assist users to investigate the importance of haptic technology in e-learning and education[15,16], interact with molecules during molecular dynamics simulations[17,18], explore interactively the solvent accessible surface (ISAS) of a protein[19], deform an elastic network model of a biomolecule by applying forces to individual atoms[20], and interact with properties related to molecular quantum dynamics (wave-packet dynamics) and potential energy surfaces[21].

A fundamental part in a haptic-based docking solution is the calculation of the interaction forces. The forces and torques acting between molecules is a consequence of nonbonded (noncovalent) interactions (assuming no chemical reaction takes place between them) which are due to van der Waals (VDW) and electrostatic interactions, if we ignore solvent induced forces. If we are treating the two molecules as rigid, these are the only interactions required although when molecules are considered to be flexible bonded interactions must be included either explicitly or implicitly.

For smooth and accurate haptic exploration and manipulation of the interaction forces, haptic technology necessitates that all force-feedback cues must maintain a refresh rate of 1 kHz due to the sensitivity of the human haptic sense. Hence, all force calculations must be performed within 1ms. A brute force approach to calculate the VDW and electrostatic forces requires a time complexity

of O(NM), where N and M are the number of atoms in the receptor and ligand respectively. Such an approach is impractical on modern CPUs even for small molecules (i.e. molecules comprising of several hundred of atoms each) due to the lack of processing power. Modern, high-end GPUs offer an alternative execution platform for this approach, and, given their processing power, have the potential to perform force calculations for large molecules (i.e consisting of thousands of atoms) within the 1ms constraint. However, our preliminary results (discussed in Section 5) suggest that the performance gains achieved by a GPU-based implementation of this approach, although noticeable, do not suffice for molecular structures containing more than a thousand atoms each.

Existing haptic-based docking approaches compute these forces on the CPU and as such, they often utilize pre-computed 3D force grids[4,22] and linear interpolation to accelerate the respective computations. These force grids address the 1kHz refresh rate requirement efficiently, however, they are time-consuming to pre-compute (especially for large molecules), memory hungry, can induce rough force transitions at cell boundaries[22] and cannot be applied to flexible docking problems (since the force grids must be recomputed in real time as the molecule deforms).

In this work, we propose an efficient octree-based, proximity querying method that enables us to compute (on the CPU) in real time and at haptic refresh rates the intermolecular forces of docking. Our approach addresses efficiently and successfully all issues related to the pre-calculated force grids and can facilitate a haptics-driven docking of large molecular structures (i.e consisting of thousands of atoms). As such, it can be applied equally in the study of protein-protein and protein-drug docking problems. We have implemented this method in a haptic-based, immersive, rigid body docking system called Haptimol_RD, which is designed to run on consumer level hardware, (i.e. there is no need for specialized/proprietary hardware). Our motivation is the eventual development of a haptic-based docking solution that can model molecular flexibility.

2 Previous work

The potential benefits of integrating haptic technology in molecular docking solutions have been under investigation since the late 60's[23]. Nonetheless, the progress made in this field has been slow. The main reasons for that were the lack of product commercialization (early haptic technology was proprietary), and the lack of the necessary computing power which rendered the use of haptic technology in docking solutions either prohibitively expensive and/or computationally infeasible. The emergence of powerful desktop computers, affordable haptic devices and open-source rendering APIs (Application Programming Interface), at the beginning of the 21st Century, alleviated these obstacles, and enabled molecular-docking researchers to incorporate haptic technologies in their studies. Though the number of related studies still remains small, it is anticipated that this number will increase as haptic technology becomes easier and cheaper to use and integrate.

Brooks et. al.[23], from the University of North Carolina, pioneered the field of haptic-based docking with the GROPE III project. They addressed and solved the force calculation problem by adopting Pattabiraman et. al.'s pre-computed,

energy-potential, grid method[24]. Similarly to Pattabiraman et. al., Brooks et. al. pre-computed and stored the inter-atomic force at predefined 3D-grid cells. Each grid cell stored the summation of the VDW and electrostatic contributions to the force. They then computed the total force of the intermolecular interactions with a tri-linear interpolation on the appropriate force vectors. Bayazit et. al.[25] applied a similar force calculation approach in their study. They developed a hybrid docking solution by integrating haptic technology into their automated motion planning method (OBPRM). In their solution, the haptic device allowed the user to sample the conformational space, sense the interaction forces, identify sites with low energy potentials, and connect these findings into a roadmap. This roadmap was then given as an input to the road planner that calculated the final docking path. Their energy and force calculations modelled only VDW interactions, and were accelerated with the use of a force grid.

Lee and Lyons[26] improved upon the force grid method by calculating separate force grids for each component of the non-bonded interactions (i.e. VDW and electrostatic). Moreover, the forces stored in the grid cells accounted for all possible pairwise atom-type combinations between the receptor and ligand molecules. Their approach provided a better approximation of the interaction forces, and enabled the user to scale, and turn on/off in real time the forces attributed to the VDW and electrostatic interactions. Their method has been adopted by many current, haptic-based docking solutions and studies related to the teaching of structural biology to students[27], to computer-aided drug design[28], and to overcoming the *trapping* problem in Molecular Dynamics simulations[14]. Like Lee and Lyons, Wollacott and Merz Jr.[22] with HAMStER utilized two different 3D-grids for force calculations (one for VDW and one for electrostatic), but they pre-computed the grids only around the active site of the receptor.

Nagata et. al.[29] attempted to compute the force of all non-bonded interactions (VDW, electrostatic, and hydrogen bonds) in real time, without utilizing pre-computed grids. They concluded that (even for small molecules) their brute-force approach could not compute the binding interactions at haptic refresh rates, without a 100-fold increase in processor power available to them at the time. Within project CoRSAIRe, Ferey et. al.[30] calculated the docking forces using a force grid for the electrostatic contribution and a brute force approach for the VDW contribution. Their brute force calculations, however, did not account for all interatomic VDW interactions, but only for those involving the surface atoms of the molecules. Hou and Sourina[31] and Sourina et. al.[32] used a brute-force approach to calculate the VDW forces in their haptic-based, helix-helix docking system called HMolDock. They were able to dock with their system a aIIb helix (with 154 atoms) and a anti-aIIb helix (with 266 atoms).

Other approaches to the computation of the intermolecular interaction forces include the works of Daunay et. al., and Zonta et. al. Daunay et. al.[7] developed a haptic-based flexible docking application and utilized a simulation engine for the respective energy computations. The forces and torques, rendered on the haptic device, were not calculated; they were converted from the total energy potential by a novel force-field reconstruction method. Their system could not support "as is" 1kHz, haptic rendering rates, and thus they relied on wave theory (and on the appropriate wave transformations) to bridge the rate disparities between haptic rendering (1kHz) and force calculations. Finally, Zonta et. al.[33] integrated the

OpenBabel library into their docking system ZODIAC and used it for all force calculations. Using ZODIAC, they were able to calculate the forces at haptic refresh rates while docking a ligand of 16 atoms to a receptor of 5.5K atoms.

3 Methods

3.1 Force Equations

The primary interest of this work is the real-time calculation of the VDW and electrostatic forces exerted between the receptor and ligand molecules during docking. The VDW interaction is modelled by the Lennard-Jones (LJ) potential and the electrostatic potential by Coulomb's law.

The VDW force acting on atom j as exerted by atom i and modelled by the LJ potential is given by,

$$\vec{F}_{ij}^{VDW} = \left[12\frac{A_{ij}}{r_{ij}^{13}} - 6\frac{B_{ij}}{r_{ij}^{7}} \right] \vec{\hat{r}}_{ij} \tag{1}$$

where A_{ij} and B_{ij} are constants that depend on the type of interacting atoms, r_{ij} is the distance between these atoms (measured from their centre), and $\vec{\hat{r}}_{ij}$ is the unit vector in the direction from atom i to j. In Equation 1, the first term inside the brackets defines the repulsive part of the force, whereas the second term defines the attractive part - the dispersion force.

The force acting on atom j due to the electrostatic interaction with atom i as modelled by Coulomb's law is given by:

$$\vec{F}_{ij}^{ES} = \frac{q_i q_j}{4\pi\varepsilon\varepsilon_0 r_{ij}^2} \vec{\hat{r}}_{ij} \tag{2}$$

where q_i and q_j are the atomic charges of the two atoms, ε_0 is the permittivity of free space, and ε is the relative permittivity dependent on the dielectric properties of the solvent. Given Equations 1 and 2 we compute the total force of the intermolecular interactions by summing up all pairwise, inter-atomic interactions involved. Namely, if N atoms from the receptor interact with M atoms from the ligand then the total force is given by,

$$\vec{F}_{Tot} = \Sigma_i^N \Sigma_j^M \left(\left[12\frac{A_{ij}}{r_{ij}^{13}} - 6\frac{B_{ij}}{r_{ij}^{7}} + \frac{q_i q_j}{4\pi\varepsilon\varepsilon_0 r_{ij}^2} \right] \vec{\hat{r}}_{ij} \right) \tag{3}$$

Equation 3 is the total force acting on the ligand due to its interactions with the receptor. The total force acting on the receptor due to its interactions with the ligand is of the same magnitude but in the opposite direction. There are related expressions for the torque acting on the molecules but these are not modelled as most haptic devices are unable to exert torques on the user. In our work, Equation 3 is applied only for those inter-atomic interactions within a cut-off distance.

We use the non-bonded parameters of the Gromos54a7[34] force field (as specified and implemented in Gromacs version 4.6.2[35]) to provide values for the parameters A_{ij}, B_{ij}, q_i and q_j. Specifically, we calculate the values of A_{ij} and B_{ij} as $A_{ij} = \sqrt{A_i \times A_j}$ and $B_{ij} = \sqrt{B_i \times B_j}$ respectively, where A_i, B_i and A_j, B_j are

the LJ parameters of atoms i and j as given by the Gromos54a7 force field. The Coulomb constant $\frac{1}{4\pi\varepsilon_0}$ is set equal to 138.935485 $kJ\ mol^{-1}\ nm\ e^{-2}$ and we set ε equal to 1.0 for the purposes of this benchmarking study, i.e. we assume interactions take place in vacuo. The total force is measured in $kJ\ mol^{-1}\ nm^{-1}$. To render it on the haptic device we convert it first to *Newtons* by dividing by 6.02329×10^{11} since 1N is equivalent to $6.02329 \times 10^{11}\ kJ\ mol^{-1}\ nm^{-1}$. We then scale it by a factor of the order of 10^{11} to ensure that a good range of forces can be felt by the user through the haptic device. In addition to Gromos54a7, our method can utilize other force fields such as AMBER[36], CHARMM[37] and OPLS-aa[38].

3.2 Octree-based proximity query for force calculations

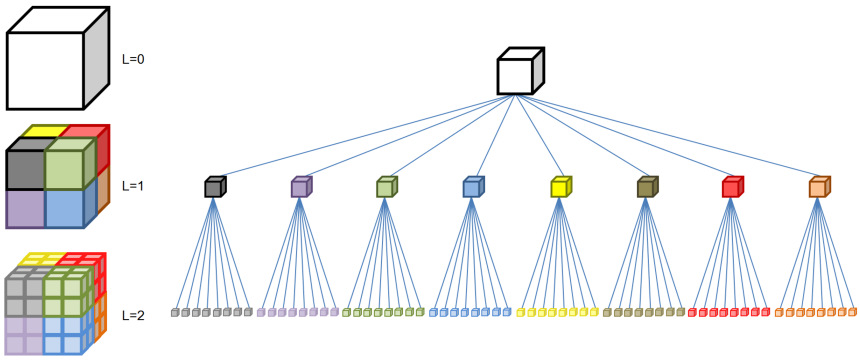


Fig. 1 An octree structure of depth 2 with its octant subdivisions. At depth 0 (L=0) lies the root octant, at depth 1 are the eight child octants of the root, and at depth 2 are the eight leaf octants of those child octants. A complete octree of depth L contains 8^L leaf octants.

For a real-time calculation of the docking forces we devised a method that uses a cut-off distance to reduce the set of interatomic interactions considered in Equation 3, and octrees[39] to obtain this set at haptic refresh rates. Octrees are spatial partitioning hierarchies that recursively divide the geometry of an object into smaller subunits (called octants), until they reach a certain subdivision depth. Initially the object's bounding box (often modelled as a cube) is subdivided into eight child octants of the same size, then each child is subdivided into eight smaller child octants, and this process continues recursively L times, where L is the subdivision depth. This recursive subdivision results in a tree structure of degree 8 and depth L that represents the object's geometry, and in which each leaf octant contains a part of that geometry (Figure 1). Such treelike structures simplify the implementation, and accelerate the execution of costly operations[40] such as object intersection discovery, neighbour finding, proximity querying etc. With octrees, these operations are most often implemented as simple, recursive tree traversals of the underlying structures. Our method constructs and stores within different octrees a spatial decomposition of the tertiary (3D) structure for both receptor and ligand molecules, and then uses these tree structures to efficiently query the 3D space and identify all atom pairs i and j whose r_{ij} distance

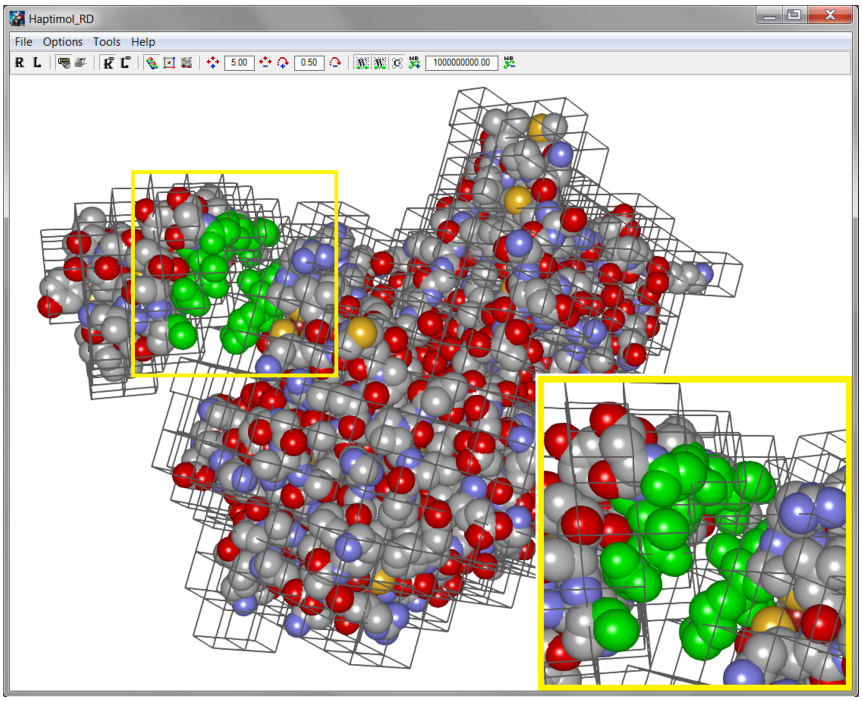


Fig. 2 Main image: Two proteins interact within Haptimol_RD. Green colours denote the atoms that interact within a given cut-off distance. The respective octree leaf octants are also depicted. Inset: Zoomed in view of the interacting atoms.

is within this cut-off (Figure 2). Force calculations are then performed on this set in real time. All tree traversals during octree construction and querying are performed recursively in a depth-first order starting from the root. Octrees have been applied in many different problems related to solid modelling, computer graphics, computer-aided design, computer vision and image processing, and robotics[40,41], but there are no reports of their application in molecular-docking problems. Octrees perform very well in problems addressing deformable geometry[42], and we want to exploit this property while studying and modelling molecular flexibility. In the following three subsections we describe our octree construction and querying algorithms, and outline how we compute the total force, respectively.

3.2.1 Octree Construction

Our method constructs an octree, and populates it with atoms simultaneously. It requires as input a target subdivision level L, and the dimensions of the box that bounds the geometry to be partitioned. The method derives initially the dimensions of the molecule's minimum bounding cube, and sets this cube as the tree's root octant. It then utilizes Algorithm 1, ***AddAtomAndConstructChildOctants***, to subdivide the tree into equally-sized child/leaf octants, and to populate them with atoms. Given an atom's coordinates, Algorithm 1 traverses the octree, and assigns the atom to the leaf octant that either intersects or contains this centre. If the path to this leaf octant (or the leaf itself) does not exist, the method creates the required octants (forming this path) dynamically, and subdivides the tree struc-

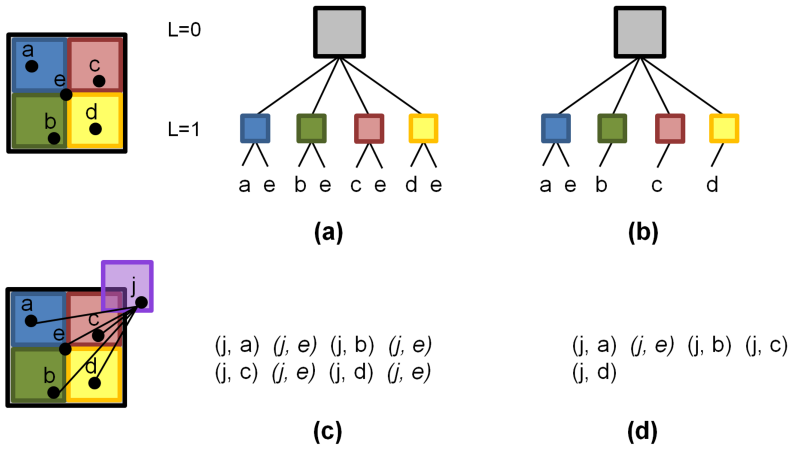


Fig. 3 Top row: A conceptual 2D visualization of an octree cube containing atoms a, b, c, d, e, and the corresponding tree structures (of depth 1) constructed. Atom e intersects all four leaf octants. **(a)** A typical construction method assigns e to all leaf octants. **(b)** Our construction method assigns e to the first applicable leaf octant traversed. Bottom row: The octree is intersected by an octree containing atom j. Atom j interacts with atoms a, b, c, d, and e. **(c)** Under the typical construction, the set of interacting atom pairs produced contains duplicates of (j,e). **(d)** Under our construction, the set contains only one instance of (j,e).

ture accordingly. This subdivision terminates when the target level L is reached, the respective leaf octant is created, and the atom is assigned successfully to it. When the atom is intersected by more than one octant our construction algorithm assigns the atom to the first leaf octant traversed, and not to all of them, as is customary for octree-based collision detection algorithms (Figures 3a and 3b). The query algorithm (see Section 3.2.2) ensures that the given atom will be considered for the final query set, irrespective of the child nodes it is assigned to. Furthermore, this one-to-one relationship between atoms and octants accelerates the performance of the query algorithm because each inter-atomic pair is considered only once, and thus the cost of handling duplicate pairs is avoided (Figures 3c and 3d). Since an octant is created only when an atom intersects it, the resultant octree contains no empty octants (octants with no atoms), and thus it is compact and memory efficient, and helps the query algorithm avoid unnecessary octant traversals. Lastly, our experiments indicated (see Section 4.1) that when the octree depth was set to four, our approach attained a performance balance between its octree construction and querying times.

3.2.2 Octree Querying

As the two molecules move in space, their geometry changes position and orientation. These changes are stored in the two viewing transformation matrices T_R and T_L, for the receptor and ligand molecules respectively. To query correctly the two octree structures, these matrices must be applied to both octant and atom coordinates stored within these trees. To save a substantial amount of matrix-vector multiplications, we combine T_R and T_L into one matrix using the relation $T_{New} = T_R^{-1} T_L$, and then apply T_{New} only to the ligand-related coordinates (we

Algorithm 1 AddAtomAndConstructChildOctants

Require: maxDepth, the octree's maximum allowable depth
Require: currentTreeLevel, the octant's current level in the octree
Require: minBoxCoord, the octant's minimum box coordinate
Require: maxBoxCoord, the octant's maximum box coordinate
Require: atomCoord, the coordinates of the atom's centre
Require: atom, the atom object to add
Ensure: filledOctree

```
1:  if isOctantNew = true then
2:      octantMinBoxCoord ← minBoxCoord
3:      octantMaxBoxCoord ← maxBoxCoord
4:      octantCentroid ← (minBoxCoord+maxBoxCoord)*0.5
5:      octantRadius ← (minBoxCoord-maxBoxCoord)*0.5
6:      isOctantNew ← false
7:  end if
8:  childrenTreeLevel ← currentTreeLevel+1
9:  // reached a tree leaf node, thus assign to this octant the given atom
10: if childrenTreeLevel > maxDepth then
11:     octantsAtomList.AddEnd(atom)
12:     isOctantNew ← true
13:     return atom added
14: else
15:     for RP = 1 to 8 do
16:         childMinBoxCoord ← get the min box coordinates for child octant
            childOctant<RP>
17:         childMaxBoxCoord ← get the max box coordinates for child octant
            childOctant<RP>
18:         if atomCoord intersect/being enclosed by childMinBoxCoord and child-
            MaxBoxCoord then
19:             if childOctant<RP> has not been created then
20:                 childOctant<RP> ← create new Octant Node
21:                 isOctantALeaf ← false
22:                 // add this child at the end of the octant's children list
23:                 octantsChildrenList.AddEnd(childOctant<RP>)
24:             end if
25:             // forward/add the atom to childOctant<RP> octants recursively
26:             return childOctant<RP>.AddAtomAndConstructChildOctants(maxDepth,
                childrenTreeLevel, childMinBoxCoord, childMaxBoxCoord, atom-
                Coord, atom)
27:         end if
28:     end for
29: end if
30: end
```

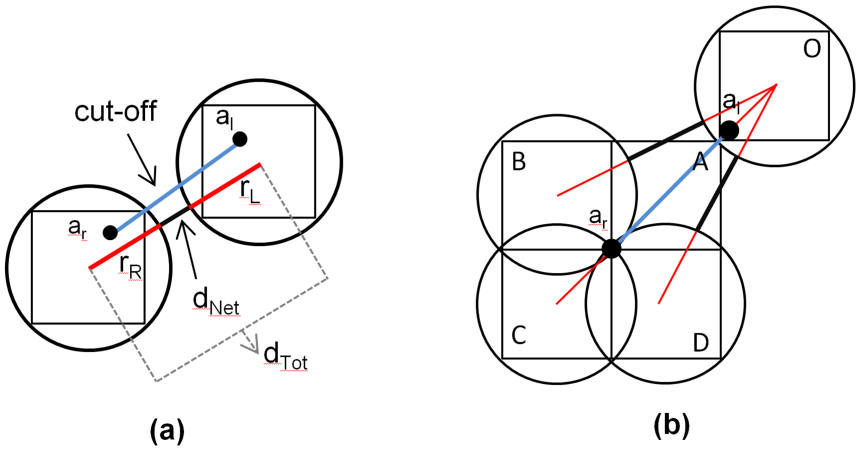


Fig. 4 **(a)** A 2D visualization of Equation 4. The two leaf octants belong to two different octrees and contain atoms a_r and a_l, respectively. These atoms are within the cut-off distance and thus interact. In such cases, d_{Net} will always be less than or equal to the cut-off, since it defines the closest distance between the two spheres (of radii r_R and r_L) bounding these atoms. **(b)** A diagram-based proof that our algorithm will handle the atom pair (a_r, a_l) correctly, irrespective of the leaf octant (A, B, C or D) atom a_r was initially assigned to. Even in the worst case scenario (a_r was assigned to C), the d_{Net} of octants C and O will be equal to the cut-off (blue line), whereas in the other three cases it will be less than the cut-off. In this example octant A is the best insertion case for a_r since it is in zero d_{Net} distance from octant O.

use $T_{New} = T_L^{-1} T_R$ and apply T_{New} to the receptor-related coordinates if the ligand is larger than the receptor). This approach maintains the relative orientation of both structures intact, and instead of transforming both molecules in space, it keeps the receptor geometry fixed and transforms only the ligand geometry with respect to it.

Given T_{New} and the two octree structures our method applies Algorithm 2, ***DeriveInteractingAtomPairsSet***, to query pairwise and recursively the underlying octants, identify all interacting atom pairs within the cut-off distance, and return the set, S_{Pairs}, of these pairs. Using the atoms stored in S_{Pairs} (and their non-bonded parameters), the force calculation procedure derives the total force. The query algorithm starts from both octree roots and performs a pairwise traversal of their respective child octants. For each non-leaf octant pair examined, it updates the necessary octant coordinates using T_{New}, and then computes the net distance d_{Net} between the octant centres using:

$$d_{Net} = d_{Tot} - (r_R + r_L) \tag{4}$$

where d_{Tot} is the total distance between the octant centres, and r_R and r_L are the radii of their bounding spheres (Figure 4a). If d_{Net} is less than or equal to the cut-off distance plus an error margin e_s (to safeguard the query from floating point arithmetic errors), the algorithm continues and examines recursively the children of this octant pair, and stops when it reaches the relevant leaf octants. At the leaf level, the algorithm computes all pairwise inter-atomic distances between

the centres of the atoms stored in the respective leaf-octants, and saves in S_{Pairs} those atom pairs with a distance less than or equal to the cut-off.

By utilizing Equation 4, the cutoff distance, and the octree hierarchy, our query strategy performs quick rejection tests on the underlying molecular geometry, and converges rapidly to those leaf-octants containing the interacting atom pairs. Our octant rejection test (i.e. $d_{Net} > (cutoff + e_s)$) is a simple numerical test with no substantial computational cost. Moreover, it is invariant to octant orientation in space, since d_{Net} is computed based on octant bounding sphere radii and not on octant box dimensions (i.e. bounding cube dimensions). Since atoms are bounded by octant boxes and octant boxes are bounded by octant bounding spheres, two atoms will interact, if and only if the d_{Net} distance of their bounding octants is less or equal to the cut-off, regardless of octant orientation. Hence, if the d_{Net} distance between two octants is not within the cut-off it is safe to discard that part of molecular geometry from the solution set, and query it no further. The same reasoning applies to the special tree construction cases stated in Section 3.2.1 (i.e. when atoms are intersected by more than one octant). As previously mentioned, the construction algorithm assigns such atoms to the first leaf octant traversed. Let a_r be one such atom. If a_r is intersected by multiple octants, then its centre must lie on a common point shared by the bounding boxes/spheres of these octants. Let, now, a_l be an atom interacting with a_r. Clearly the d_{Net} distance between the octants containing a_l and a_r must be less than or equal to the cut-off. But since a_r lies on a shared point then the d_{Net} distance between the octant containing a_l and the remaining octants must also be less than or equal to the cut-off (Figure 4b). As such, a_r will be queried and inserted in S_{Pairs} as expected, regardless of its placement within the octree during construction. Hence, our rejection test will prune correctly the octrees (and their underlying geometry), under all construction cases, special or trivial. Nonetheless, floating point precision errors might induce false fails in our test, and inadvertently lead to atom omissions. The use of e_s safeguards the rejection test and query algorithm from such erroneous outcomes. In our queries we let e_s to be the VDW radius of the smallest atom in the given molecule.

3.2.3 Calculating the Force

Our force calculation procedure traverses sequentially the S_{Pairs} set, and for each atom pair found in SPairs, it computes the VDW and electrostatic force contributions and adds them to the total force. Since all force calculations are performed in real time, our method can facilitate independent handling of the electrostatic and VDW forces in a manner similar to the one reported in Lee and Lyons[26]. Namely, it enables the user to scale and switch on/off dynamically the electrostatic and VDW forces, as well as, the repulsive and attractive parts of the VDW force. Such force calculation flexibility allows the user to experiment with different types of interactions easily.

4 Results

To test our approach we have implemented the methods discussed above in a haptic-based, rigid-docking application called Haptimol_RD. Haptimol_RD was developed using the OpenHaptics HD and OpenGL libraries, and the Visual C++

Algorithm 2 DeriveInteractingAtomPairsSet

Require: T_{New}, the combined viewing transformation matrix
Require: octree1Octant, an octant from the first octree structure
Require: octree2Octant, an octant from the second octree structure
Require: cutoff, the cut-off distance
Ensure: S_{Pairs}

retValue ← false
if both octree1Octant **AND** octree2Octant are leaf-octants **then**
  for all atoms a_r in octree1Octant and all atoms a_l in octree2Octant **do**
    d_{atoms} ← compute inter-atomic distance between a_r and a_l
    if $d_{atoms} \leq$ cutoff **then**
      save pair (a_r, a_l) in S_{Pairs}
      retValue ← true
    end if
  end for
else
  if octree1Octant **OR** octree2Octant is a leaf-octant **then**
    // set non-leaf octant to tmpNLOctant and leaf octant to tmpOctant
    if octree1Octant is a leaf-octant **then**
      tmpNLOctant ← octree2Octant
      tmpOctant ← octree1Octant
    else
      tmpNLOctant ← octree1Octant
      tmpOctant ← octree2Octant
    end if
    for all child octants oct_c in tmpNLOctant **do**
      d_{Net} ← compute net distance between oct_c and tmpOctant
      if $d_{Net} \leq$ (cutoff+e_s) **then**
        if DeriveInteractingAtomPairsSet(T_{New}, oct_c, tmpOctant, cutoff) **then**
          retValue ← true
        end if
      end if
    end for
  else
    for all child octants oct_r in octree1Octant and all child octants oct_l in octree2Octant **do**
      update necessary octant coords. (i.e. the shortest octree) with T_{New}
      d_{Net} ← compute net distance between oct_r and oct_l
      if $d_{Net} \leq$ (cutoff+e_s) **then**
        if DeriveInteractingAtomPairsSet(T_{New}, oct_r, oct_l, cutoff) **then**
          retValue ← true
        end if
      end if
    end for
  end if
end if
return retValue
end

programming language. Using Haptimol_RD we conducted the following set of experiments:

1. benchmarking construction and querying performances

2. measuring querying performance during real-time rigid-docking simulations

To account for possible speed inconsistencies caused by background processes, we executed each benchmarking experiment ten times and reported the average. All of our tests were conducted on a 2.93GHz Intel Core i7 PC, running a 64bit version of Windows with 8GB RAM. The haptic device utilized in our simulations was the Geomagic Touch (formerly known as SensAble Technologies Phantom Omni).

4.1 Benchmarking Performance

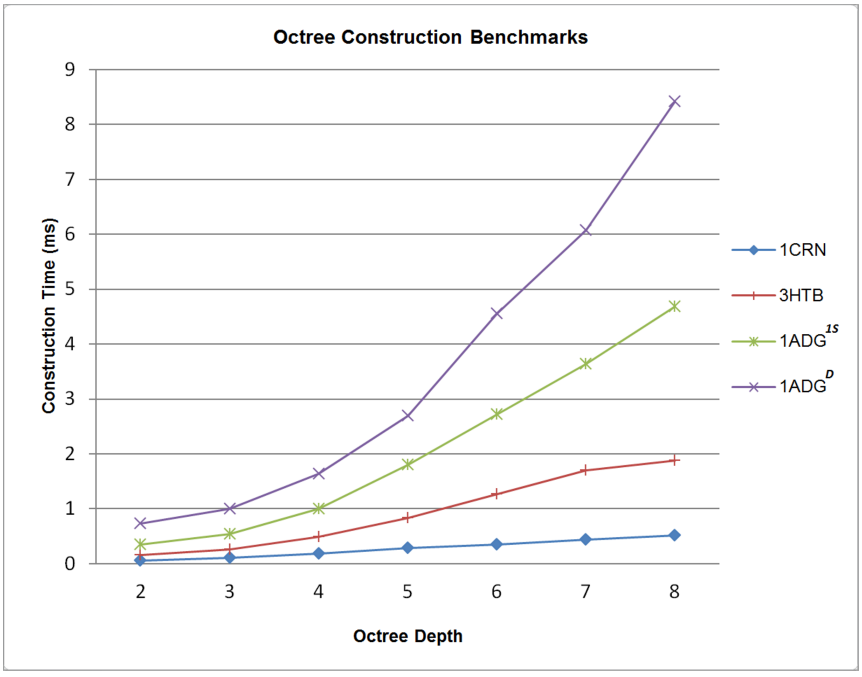


Fig. 5 Octree construction times per molecule in milliseconds, for octree depth levels 2-8.

We report the octree construction and querying times on various molecular structures. The construction times must be considered if receptor flexibility is modelled, given that the trees would have to be constructed repeatedly and in real time as the molecule deforms. For rigid-body docking, the querying times are the only values of practical importance, since the trees need only be calculated once prior to the interactive session. We benchmarked our octree construction

method using four proteins comprising different numbers of atoms. The protein structures were obtained from the Protein Data Bank (PDB)[43]. The proteins used were *Crambin* (PDB code: 1CRN), *Lysozyme* (3HTB), *Alcohol Dehydrogenase* [containing only one of the two subunits, 1ADG^{1S}], *Alcohol Dehydrogenase* [containing both subunits (dimer), 1ADG^{D}]. For each test protein we constructed seven octrees of different depths (from 2 to 8), and recorded the number of atoms each protein comprises, the total number of octants created, and the total number of leaf octants (Table 1). For depths lower than four, our method constructed the respective octree structures within one millisecond, in all test cases. At depth four it supported millisecond/sub-millisecond construction times for proteins containing up to 3500 atoms. For depths larger than four, sub-millisecond construction times were attained only for small proteins (comprised of several hundreds of atoms). Figure 5 depicts these construction times per protein, for all seven depth values. We did not test our construction method for depth values larger than eight due to physical memory limitations.

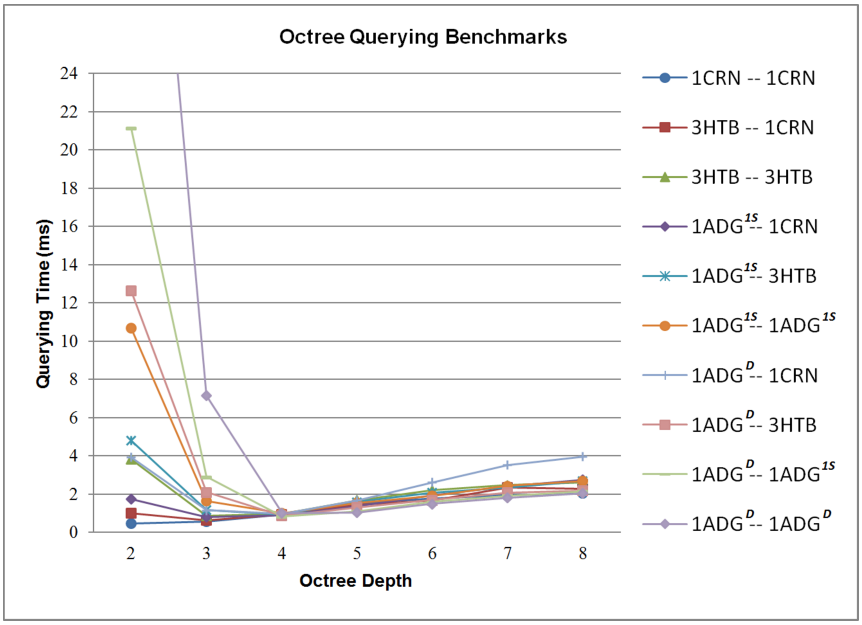


Fig. 6 Octree querying times of our ten interacting molecular pairs in milliseconds for the same seven octree depths (i.e. 2-8). The querying time for test case 1ADG(dimer)–1ADG(dimer) at depth 2 is not shown here (to avoid graph scaling and to improve graph readability). The time for this test case was 51.275 ms.

To benchmark our querying method we used the same four proteins and we devised ten test cases as shown in Figure 6. Each test case consisted of two proteins (out of this set), the larger of which was assigned as the receptor. Based on their centres, the proteins were placed at the origin of the Cartesian space and the ligand molecule was then translated along the positive x-axis by a displacement distance δ_T. The δ_T distance varied per test case, and was chosen empirically. Namely, we displayed both proteins on the screen, moved the lig-

and along the positive x-axis while allowing it to intersect the receptor, and then visually selected the distance that generated a substantial amount of overlapping octants (and atoms), that allowed us to test adequately the performance of our querying method in relation to tree depth and protein size. Since such extensive atom overlapping would never occur during an actual docking simulation (because of the VDW repulsive forces), the querying response times recorded can also act as sufficient, upper-bound, performance indicators for our proximity query method. We used the value of 8Å for the cut-off distance, and for each test case we recorded the δ_T distance used, the querying response time, the cardinality of the S_{Pairs} set, the total number of child and leaf octants traversed and the total number of inter-atomic distance calculations (Table 2). The querying response time is the time to determine the set, S_{Pairs}, the time to perform the force calculation being negligible in comparison. Figure 6 depicts these querying times per test case, for all seven depth values. According to these results, our method achieved sub-millisecond querying response times for all test cases when the respective octree depths were set to four. Faster response times were attained for smaller octree depths in several cases, however our measurements indicated that at depths equal to four our approach maintained a performance balance between the construction and querying times. Evidently, the octree construction and querying costs are depth and geometry-size dependent, and impose a trade-off between construction speed and querying performance. This relationship is depicted in Tables 1 and 2 which list the construction and querying measurements pertinent to each test case, for depths 3, 4, and 5. Clearly as the size of the interacting proteins increases the construction cost also increases monotonically by almost twofold at every step, whereas the querying cost remains substantially lower at depths higher than four, than it is at depths three and lower. Measurements for depths 2, 6, 7 and 8 were not included for table clarity.

Table 1 Octree construction times per molecule in milliseconds, for depth levels 3, 4, and 5. The table lists the number of atoms comprising each molecule, the total number of octants created, and the total number of octants at the leaf level. *D* stands for Dimer and 1*S* for one subunit.

Molecule	# of heavy atoms	L	Tot. # of octants	Tot. # of leaf octants	Construction time (ms)
1CRN	327	3	132	97	0.1023
		4	372	240	0.1837
		5	699	327	0.2837
3HTB	1388	3	165	124	0.2617
		4	683	518	0.4879
		5	1810	1127	0.8329
$1ADG^{1S}$	3445	3	190	143	0.539
		4	888	698	1.0025
		5	3122	2234	1.8006
$1ADG^{D}$	7046	3	137	108	1.0035
		4	644	507	1.6403
		5	3135	2491	2.6987

Table 2 Octree querying times for ten interacting molecular pairs in milliseconds, for depth levels 3, 4, and 5. The table lists the displacement distance used in our experiments, the number of interacting atom pairs (S_{Pairs}) returned, the total number of child/leaf octants traversed, and the total number of atom pairs examined in order to generate the set S_{Pairs}. *D* stands for Dimer and 1*S* for one subunit.

Interacting Molecules	δ_T (nm)	S_{Pairs}	L	Tot. # of octants travers.	Tot. # of leaf octants travers.	Tot. # of atom pairs exam.	Querying time (ms)
			3	2841	2450	30427	0.5666
1CRN–1CRN	1.55	4146	4	10405	7564	13792	0.9428
			5	18542	8137	8137	1.5023
			3	2076	1729	70700	0.6236
3HTB–1CRN	2	4040	4	8117	6041	22046	0.9337
			5	16690	8573	10234	1.4212
			3	1514	1114	123272	0.8613
3HTB–3HTB	2.9	3224	4	6707	5193	34134	0.9497
			5	14623	7916	11761	1.6833
			3	1474	1183	109636	0.8126
1ADG^{1S}–1CRN	2.8	4792	4	6885	5411	33116	0.9073
			5	16536	9651	14524	1.3562
			3	1140	848	211686	1.1711
1ADG^{1S}–3HTB	3.88	4338	4	5128	3988	48060	0.9148
			5	14077	8949	16695	1.587
			3	1008	670	315964	1.6513
1ADG^{1S}–1ADG^{1S}	4.29	3168	4	4254	3246	62847	1.0028
			5	11787	7533	17576	1.5472
			3	1040	838	218443	1.1536
1ADG^{D}–1CRN	3.27	8853	4	4824	3784	72293	0.9758
			5	16112	11288	31079	1.6743
			3	918	686	437063	2.096
1ADG^{D}–3HTB	4.2	2952	4	3670	2752	83642	0.8688
			5	10375	6705	22332	1.3097
			3	787	562	715797	2.902
1ADG^{D}–1ADG^{1S}	5.21	2569	4	2642	1855	99087	0.837
			5	8108	5466	23060	1.0944
			3	730	565	1719242	7.1563
1ADG^{D}–1ADG^{D}	5.58	2452	4	1830	1100	122310	0.9926
			5	5703	3873	25108	1.0454

4.2 Real-time Rigid-Docking Simulation

In addition to our benchmarking experiments we measured the performance of our proximity querying method in real-time, rigid-docking simulations, using three well known complexes, related to protein-protein and protein-drug docking. Although we did not model receptor flexibility in these simulations, we report the respective octree-construction times for the reader who wants to take into account these construction overheads (necessary if molecular flexibility is addressed). In our docking simulations, we used the complexes of *Epidermal Growth Factor* (EGF) with EGF receptor (EGFr), *Bovine Pancreatic Trypsin Inhibitor* (BPTI) with *Trypsin*, and anticancer drug *BAY43-9006* (sorafenib, Nexavar) with cancer target *B-raf* as defined in the 1NQL, 3OTJ, and 1UWH PDB files respectively. Out of these files, we extracted the 3D geometry of the six underlying molecules, in their binding conformations. Some molecules had gaps in their geometry such as incomplete residues with missing atoms. These geometry gaps, however, were not significant enough to influence (positively or negatively) our performance measurements. We utilized Gromacs 4.6.2 (with the *-missing* flag when required) to get the non-bonded force parameters for all molecules except the drug sorafenib. For sorafenib we obtained the parameters through PRODRG server (http://davapc1.bioch.dundee.ac.uk/programs/prodrg/)[44]. We conducted three docking simulations using Haptimol_RD, and the respective geometry and force parameter files. During the simulations, the user performed a haptic exploration of the receptor with the ligand, guided the ligand to its docking position and orientation (as defined in the original PDB file), and sensed the underlying intermolecular interactions on the haptic device. The simulation lasted slightly more than a minute, and Haptimol_RD recorded at 10 millisecond intervals the querying response times, and the number of atom pairs generated. Figures 7, 8, and 9 depict these docking simulation results, whereas Table 3 lists the corresponding construction results. In our simulations we used octrees of depth 4, and a cut-off distance of 8Å.

Table 3 Octree construction times for the six molecules used in our real-time docking simulations.

Molecule	# of heavy atoms	Tot. # of octants	Tot. # of leaf octants	Construction time (ms)
sorafenib	48	100	45	0.0384
EGF	483	391	268	0.1929
BPTI	604	399	285	0.237
TRYPSIN	2094	843	656	0.6758
B-raf	5376	1012	795	1.3382
EGF*r*	5836	615	468	1.2572

Our results show that our method attained sub-millisecond response times for the majority of the simulation period. Our querying times exceeded slightly the 1ms barrier only when BPTI assumed its final docking position. At that position our method performed a significant number of octant comparisons, induced by a substantial octree overlapping, because BPTI was docked deep into Trypsin's binding pocket. The response times in that case fluctuated between 1.015 and

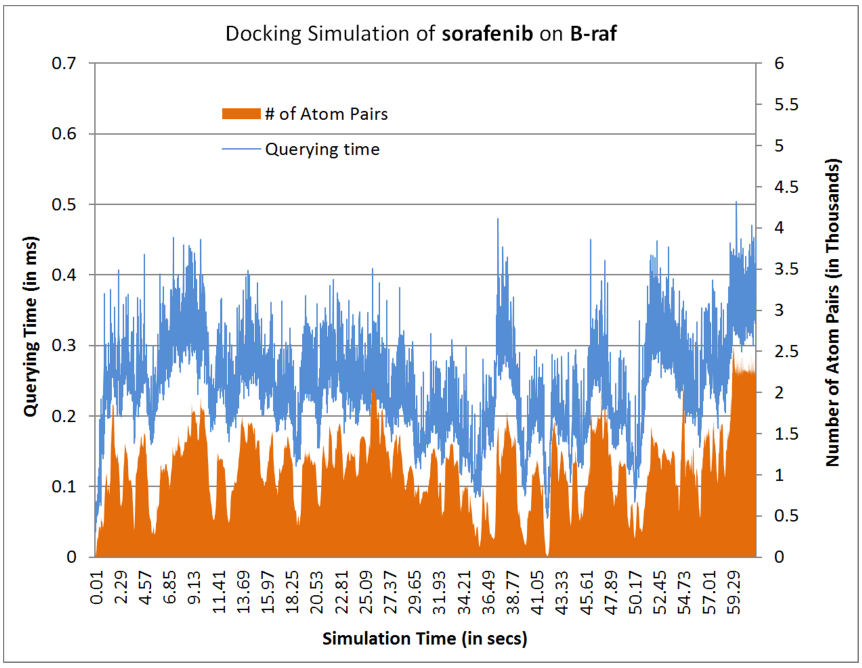


Fig. 7 A graph of the querying times and the number of atom pairs generated during a real-time, haptic-based docking simulation between the drug molecule sorafenib and the receptor protein B-raf.

1.092 ms.

5 Discussion

We have developed an octree-assisted, force calculation method applicable to haptic-based, rigid/flexible docking. When applied to rigid docking, our approach can facilitate larger molecular structures than the ones reported in previous studies, and it is therefore not constrained to protein-drug docking but can also be applied to protein-protein docking. Since tree construction times are irrelevant in rigid docking, our approach can maintain 1ms force refresh rates for molecular pairs comprising of 7K atoms each (e.g. $1ADG^D$-$1ADG^D$). On the other hand, flexible docking necessitates the real-time construction of the tree, and as such, 1ms refresh rates would be achieved only for molecular pairs of up to 1.4k atoms each (e.g. 3HTB-3HTB). These sizes can be increased by more than a twofold if we relax the 1kHz refresh rate requirement down to 500Hz (as noted in Otaduy and Lin[45]).

We also anticipate substantial performance gains if we adapt our method for a high-end GPU. Our belief stems from the observation that our GPU-based implementation of the brute force approach outperformed the CPU-based implementation by almost eightfold. Our GPU implementation (on an NVidia GeForce GTX 580) enabled us to compute the interaction force between eight hundred thousand atom pairs (simulating a receptor with one thousand atoms and a ligand

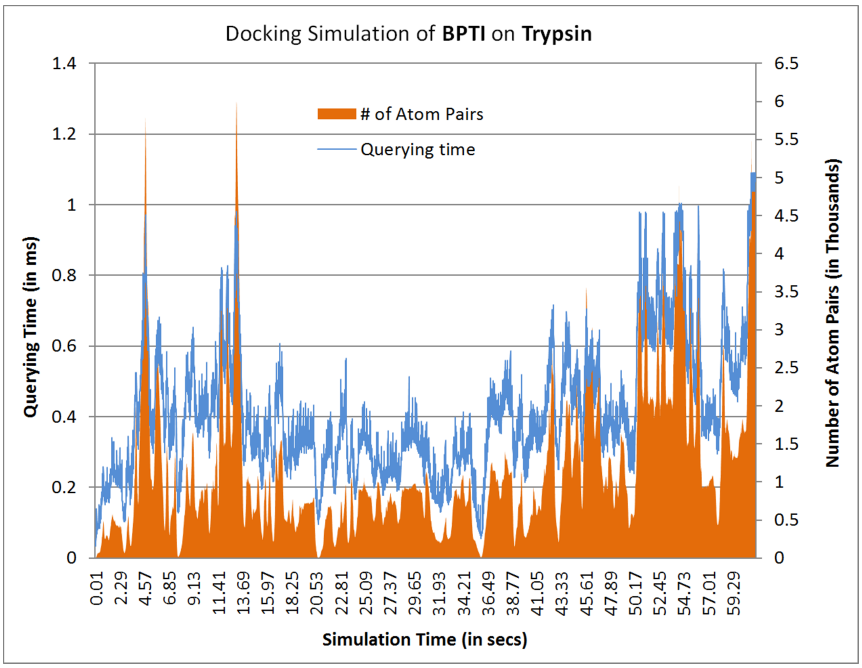


Fig. 8 A graph of the querying times and the number of atom pairs generated during a real-time, haptic-based docking simulation between the inhibitor BPTI and the receptor protein Trypsin.

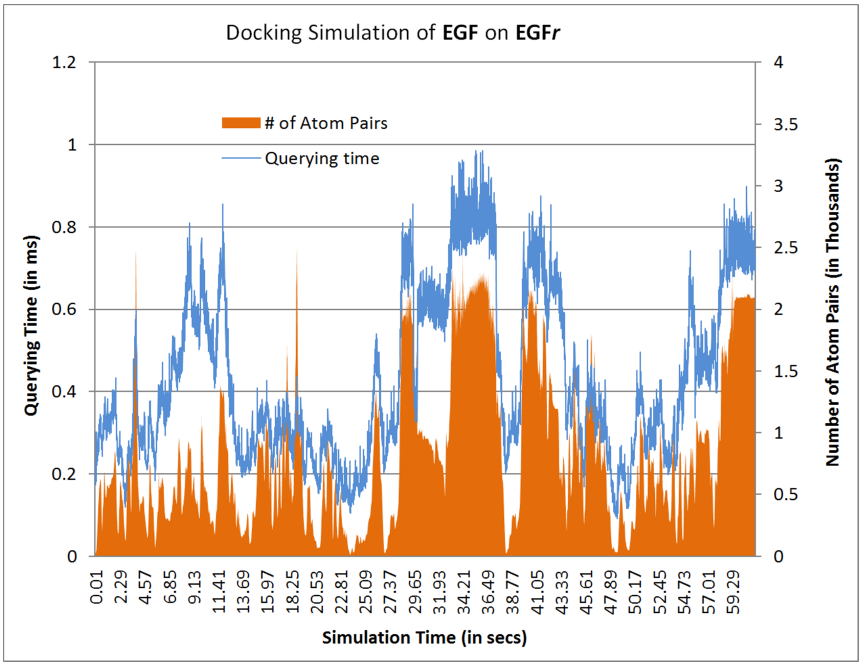


Fig. 9 A graph of the querying times and the number of atom pairs generated during a real-time, haptic-based docking simulation between EGF and the receptor protein EGF*r*.

with eight hundred atoms), whereas the CPU implementation (on a 2.93GHz Intel Core i7) constrained this set only to twenty thousand atom pairs (simulating a receptor with two hundred atoms, and a ligand with one hundred atoms). We expect to obtain analogous performance gains (if not better) when we transfer our octree proximity querying approach to the GPU. Therefore, a GPU-based implementation of our approach is the next logical step.

6 Conclusion

We have developed a method based on octrees for the real-time computation of the electrostatic and VDW force contributions in molecular docking. Our implementation calculates the total interaction force within haptic refresh rates using a cut-off distance, and overcomes the computational limitations of precomputed force grids.

We have presented our octree construction and querying algorithms, and implemented and tested the performance of our method using different molecular structures. Our results show that we can achieve sub-millisecond force calculation responses for examples of interest from protein-protein and protein-drug docking.

Haptic-based docking enables intuitive and interactive exploration of binding poses which may give an advantage over automated approaches. It is expected that future research will attempt to improve the docking accuracy of such systems by incorporating more realistic force calculations (e.g. that model solvent effects implicitly), and addressing receptor-ligand flexibility.

References

1 N. Moitessier, P. Englebienne, D. Lee, J. Lawandi, Corbeil and CR, *Br. J. Pharmacol.*, 2008, **153**, S7–S26.
2 M. Eisenstein and E. Katchalski-Katzir, *C. R. Biol.*, 2004, **327**, 409–420.
3 E. Yuriev, M. Agostino and P. A. Ramsland, *J. Mol. Recognit.*, 2011, **24**, 149–164.
4 M. Ouhyoung, *PhD thesis*, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1990.
5 W. Humphrey, A. Dalke and K. Schulten, *J. Mol. Graphics*, 1996, **14**, 33–38.
6 D. Schneidman-Duhovny, Y. Inbar, R. Nussinov and H. J. Wolfson, *Nucleic Acids Res.*, 2005, **33**, W363–W367.
7 B. Daunay, A. Micaelli and S. Régnier, Actes de ICRA'07 IEEE International Conference on Robotics and Automation, Rome, Italie, 2007, pp. 840–845.
8 A. R. Leach, B. K. Shoichet and C. E. Peishoff, *J. Med. Chem.*, 2006, **49**, 5851–5855.
9 K. Salisbury, F. Conti and F. Barbagli, *IEEE Computer Graphics and Applications*, 2004, **24**, 24–32.
10 R. Sayle and J. Milner-White, *Trends Biochem. Sci.*, 1995, **20**, 374–376.
11 E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng and T. E. Ferrin, *J. Comput. Chem.*, 2004, **25**, 1605–1612.
12 W. L. Delano, *The PyMOL Molecular Graphics System*, 2002, `http://www.pymol.org`.
13 T. E. Ferrin, C. C. Huang, L. E. Jarvis and R. Langridge, *J. Mol. Graphics*, 1988, **6**, 13–27.
14 E. Subasi and C. Basdogan, Proceedings of EuroHaptics Conference, 2006, pp. 141–145.
15 A. Krenek, Proceedings of Eurohaptics, 2001, pp. 142–145.
16 C. M. Sauer, W. A. Hastings and A. M. Okamura, Proceedings of Eurohaptics, 2004, pp. 5–7.
17 N. Férey, O. Delalande, G. Grasseau and M. Baaden, Proceedings of the 2008 ACM symposium on Virtual reality software and technology, 2008, pp. 91–94.

18 J. E. Stone, J. Gullingsrud and K. Schulten, Proceedings of the 2001 symposium on Interactive 3D graphics, 2001, pp. 191–194.

19 M. Stocks, S. Hayward and S. Laycock, *BMC Struct. Biol.*, 2009, **9**, 69–75.

20 M. B. Stocks, S. D. Laycock and S. Hayward, *J. Comput.-Aided Mol. Des.*, 2011, **25**, 203–211.

21 R. A. Davies, N. W. John, J. N. MacDonald and K. H. Hughes, Proceedings of the tenth international conference on 3D Web technology, 2005, pp. 143–150.

22 A. M. Wollacott and K. M. Merz Jr, *J. Mol. Graphics Modell.*, 2007, **25**, 801–805.

23 F. P. Brooks Jr, M. Ouh-Young, J. J. Batter and P. Jerome Kilpatrick, ACM SIGGraph computer graphics, 1990, pp. 177–185.

24 N. Pattabiraman, M. Levitt, T. Ferrin and R. Langridge, *J. Comput. Chem.*, 1985, **6**, 432–436.

25 O. B. Bayazit, G. Song and N. M. Amato, Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, 2001, pp. 954–959.

26 Y.-G. Lee and K. W. Lyons, *Computer-Aided Design*, 2004, **36**, 75–90.

27 P. Bivall, *PhD thesis*, Linköping University, Linköping, Sweden, 2010.

28 S. K. Lai-Yuen and Y.-S. Lee, Computer Aided Design and Computer Graphics, 2005. Ninth International Conference on, 2005, pp. 199–204.

29 H. Nagata, H. Mizushima and H. Tanaka, *Bioinformatics*, 2002, **18**, 140–146.

30 N. Férey, J. Nelson, C. Martin, L. Picinali, G. Bouyer, A. Tek, P. Bourdot, J.-M. Burkhardt, B. F. Katz, M. Ammi *et al.*, *Virtual Reality*, 2009, **13**, 273–293.

31 X. Hou and O. Sourina, Cyberworlds (CW), 2010 International Conference on, 2010, pp. 25–31.

32 O. Sourina, J. Torres and J. Wang, *Transactions on Edutainment II*, Springer, 2009, pp. 105–118.

33 N. Zonta, I. J. Grimstead, N. J. Avis and A. Brancale, *J. Mol. Model.*, 2009, **15**, 193–196.

34 N. Schmid, A. P. Eichenberger, A. Choutko, S. Riniker, M. Winger, A. E. Mark and W. F. van Gunsteren, *Eur. Biophys. J.*, 2011, **40**, 843–856.

35 D. van der Spoel, E. Lindahl, B. Hess and the GROMACS development team, *GROMACS User Manual Version 4.6.2.*, University of Groningen, Royal Institute of Technology and Uppsala University, www.gromacs.org, 2013.

36 S. J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, S. Profeta and P. Weiner, *J. Am. Chem. Soc*, 1984, **106**, 765–784.

37 B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, S. Swaminathan, M. Karplus *et al.*, *J. Comput. Chem.*, 1983, **4**, 187–217.

38 R. C. Rizzo and W. L. Jorgensen, *J. Am. Chem. Soc*, 1999, **121**, 4827–4836.

39 C. L. Jackins and S. L. Tanimoto, *Computer Graphics and Image Processing*, 1980, **14**, 249–270.

40 H. H. Chen and T. S. Huang, *Computer Vision, Graphics, and Image Processing*, 1988, **43**, 409–431.

41 M. C. Lin and D. Manocha, 2003.

42 M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser *et al.*, Computer Graphics Forum, 2005, pp. 61–81.

43 H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. N. Shindyalov and P. E. Bourne, *Nucleic Acids Res.*, 2000, **28**, 235–242.

44 A. W. Schuttelkopf and D. M. Van Aalten, *Acta Crystallogr., Sect. D: Biol. Crystallogr.*, 2004, **60**, 1355–1363.

45 M. A. Otaduy and M. C. Lin, ACM SIGGRAPH 2005 Courses, 2005, p. 3.