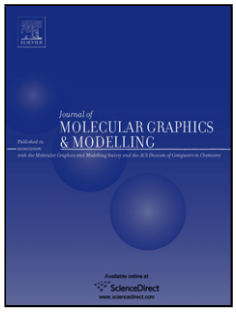


Title: Adaptive GPU-accelerated force calculation for interactive rigid molecular docking using haptics

Author: Georgios Iakovou Steven Hayward Stephen D. Laycock

PII: S1093-3263(15)30007-3
DOI: http://dx.doi.org/doi:10.1016/j.jmgm.2015.06.003
Reference: JMG 6557

To appear in: *Journal of Molecular Graphics and Modelling*

Received date: 16-4-2015
Revised date: 8-6-2015
Accepted date: 9-6-2015

Please cite this article as: Georgios Iakovou, Steven Hayward, Stephen D. Laycock, Adaptive GPU-accelerated force calculation for interactive rigid molecular docking using haptics, *Journal of Molecular Graphics and Modelling* (2015), http://dx.doi.org/10.1016/j.jmgm.2015.06.003

Adaptive GPU-accelerated force calculation for interactive rigid molecular docking using haptics

Georgios Iakovou, Steven Hayward*, Stephen D. Laycock*
School of Computing Sciences, University of East Anglia, Norwich, NR4 7TJ

Abstract

Molecular docking systems model and simulate *in silico* the interactions of intermolecular binding. Haptics-assisted docking enables the user to interact with the simulation via their sense of touch but a stringent time constraint on the computation of forces is imposed due to the sensitivity of the human haptic system. To simulate high fidelity smooth and stable feedback the haptic feedback loop should run at rates of 500Hz to 1kHz. We present an adaptive force calculation approach that can be executed in parallel on a wide range of Graphics Processing Units (GPUs) for interactive haptics-assisted docking with wider applicability to molecular simulations. Prior to the interactive session either the regular grid or an octree is selected according to the available GPU memory to determine the set of interatomic interactions within a cutoff distance. The total force is then calculated from this set. The approach can achieve force updates in less than 2ms for molecular structures comprising hundreds of thousands of atoms each, with performance improvements of up to 90 times the speed of current CPU-based force calculation approaches used in interactive docking. Furthermore, it overcomes

*Corresponding author
Email addresses: sjh@cmp.uea.ac.uk (Steven Hayward), s.laycock@uea.ac.uk (Stephen D. Laycock)

several computational limitations of previous approaches such as pre-computed force grids, and could potentially be used to model receptor flexibility at haptic refresh rates.

Keywords: Molecular Docking, Protein-Protein Interactions, Structure-based Drug Design, Force Feedback, Proximity Querying

1. Introduction

Molecular docking refers to those computational methods that try to fit two molecules (often referred to as *receptor* and *ligand*) together in their binding pose based on their topographic and physiochemical properties. It is a challenging computational problem due to the high dimensionality of the underlying search space of binding conformations (especially when the molecules are treated as flexible), with application areas in the fields of protein-protein interactions and drug design.

Docking approaches are often categorised as *automated* or *interactive*. Automated docking approaches search the space for possible binding conformations utilizing sophisticated pose selection and scoring algorithms[1, 2, 3, 4]. They can result in a large number of scored poses where the correct binding conformation is expected to have a high score.

Interactive systems provide 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. Interactive docking is not able to search a large number of docking pairs as in automated docking. However, it allows the user to focus the search based on their knowledge and expertise[5, 6, 7]. It could also provide insight into the process of docking itself [8]. Many interactive systems utilize *haptic* feedback devices to enhance human-computer interaction with the sense

of touch. Haptics-assisted docking systems enable the use of the sense of touch to feel the interaction forces and guide the molecules to their binding configuration. They offer an immersive virtual learning environment for the study of the docking process, and a test bed for exploring new ideas and hypotheses [8] (e.g. whether electrostatic steering is involved in the process). Moreover in virtual screening, they can assist experts to improve upon or reject the leading docking conformations identified by their automated counterparts[9, 8]. It has been shown that such docking systems can reduce incorrect binding poses[10], and improve the users' (experts or students of structural biology) understanding of the process of molecular binding[11, 12].

A fundamental part of haptics-assisted interactive docking is the calculation of the interaction forces acting between the molecules. Forces and torques are a consequence of nonbonded (noncovalent) interactions and their calculation can be time consuming, particularly when considering molecules comprising large numbers of atoms. For continuous, smooth and stable kinesthetic and tactile responses, modern haptic technology requires haptic feedback cues to be updated at a refresh rate of 500Hz to 1 kHz due to the sensitivity of the human haptic system [13][14][15]. When this rate is not met device vibrations and force discontinuities can occur limiting practical use. Due to this haptic docking applications are mostly constrained to rigid molecules with interaction forces calculated as sums of pairwise distance-dependent interactions.

Existing haptics-assisted docking systems which are executed on the CPU address this time constraint in various ways. The simplest approach is the brute force method[16, 17, 18] (i.e. compute all interatomic interactions between the two molecules) studied initially by Nagata et. al.[16]. On modern CPUs this approach can accommodate molecules comprising several hundred atoms each[19]. Molecule

sizes larger than these make this approach impractical. An alternative is to use pre-computed 3D force grids[20] as first proposed by Brooks et. al.[21]. Force grid systems usually treat both molecules or just the receptor as rigid structures, and pre-compute a 3D force grid of the van der Waals (vdW) and/or electrostatic forces around the receptor[21, 22, 23, 24, 25, 26] or the receptor's active site[27]. Such grids, however, have high memory requirements and induce rough force transitions at cell boundaries[27]. Moreover, by design they cannot accommodate receptor flexibility since the grids must be computed at haptic refresh rates after each structural deformation. Though this approach remains the most popular one in the field it has been applied thus far only to rigid protein small-molecule docking problems, it would be impractical for very large rigid protein-protein docking problems[9], and it cannot be extended to deal with molecular flexibility. Other CPU-based approaches include the works of Daunay et. al.[28], and Zonta et. al.[29] Daunay et. al. developed a system that models molecular flexibility, and uses a molecular dynamics engine to compute the forces. The system could not achieve haptic force-refresh rates, and therefore it circumvented the 2ms constraint entirely by using wave transformations to bridge the rate disparities between haptic rendering and force calculations. Zonta et. al.[29] addressed ligand flexibility and used a third-party library to accelerate force computations. Both approaches however, are applied only to the study of protein small-ligand docking problems.

Recently GPU-accelerated approaches for haptic-assisted docking have been reported. Anthopoulos et.al. applied a GPU-based force calculation approach[30] to their haptic-driven molecular modelling simulator[31] in order to evaluate the induced fit effect during protein-drug docking. Their approach addresses flexibility to some degree, but not at haptic refresh rates since it updates the forces at 33Hz (30ms response time). Furthermore, it can be applied only to the study of

protein-small ligand docking. Current haptics-assisted interactive docking systems cannot manage (within the 2ms constraint) docking problems (rigid or flexible) of very large molecules and they have thus been limited to a) rigid protein-ligand docking problems of average size molecules (i.e comprising a couple of thousand of atoms), and b) rigid receptor-flexible ligand docking problems of very small ligand molecules.

Our contribution to the field is an adaptive GPU-accelerated force calculation approach which uses either a regular grid or an octree spatial partitioning structure. The choice of partitioning structure is made automatically based on the GPU resources and the molecules loaded. This leads to a force calculation approach capable of computing the intermolecular forces (vdW and electrostatic) within 2ms for very large molecular structures, comprising hundreds of thousands of atoms each, with no pre-computation requirements on the receptor. It can be applied equally to the interactive haptics-assisted study of protein-protein and protein-drug docking problems. Moreover, it can in principle support receptor flexibility (providing conformational change can be computed sufficiently fast), as in contrast to pre-computed force grids there is no additional overhead in the force calculation when atoms change position.

2. Methods

2.1. Calculating the Force

As for most haptics-assisted, interactive docking approaches, we consider only the vdW and electrostatics interactions. The vdW interaction is modelled by the Lennard-Jones potential and the electrostatic interaction by Coulomb's law.

Equation 1 gives the total force acting on the ligand of M atoms while interacting

with a receptor of N atoms,

$$\vec{F}_{Tot} = \sum_{i}^{N}\sum_{j}^{M}\left(\left[12\frac{A_{ij}}{r_{ij}^{13}} - 6\frac{B_{ij}}{r_{ij}^{7}} + \frac{q_i q_j}{4\pi\epsilon\epsilon_0 r_{ij}^2}\right]\vec{r}_{ij}\right) \quad (1)$$

where A_{ij} and B_{ij} are constants that depend on the type of interacting atoms, q_i and q_j are the atomic charges of the two atoms, ϵ_0 is the permittivity of free space, ϵ is the relative permittivity dependent on the dielectric properties of the solvent, r_{ij} is the distance between these atoms, and $\vec{r}_{ij}$ is the unit vector in the direction from atom i to j. By reversing the force direction we get the total force acting on the receptor due to its interactions with the ligand. In this study we do not model the torques acting on those molecules as most low-cost haptic devices are unable to render them. Furthermore, we apply Equation 1 only for those inter-atomic interactions within the cut-off distance.

We used the Gromos54a7[32] force field (as specified and implemented in Gromacs version 4.6.2[33]) to get values for the parameters A_{ij}, B_{ij}, q_i and q_j; namely, we compute 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 Lennard-Jones parameters of atoms i and j defined by the force field. We set the Coulomb constant $\frac{1}{4\pi\epsilon_0}$ equal to 138.935485 $kJ\ mol^{-1}\ nm\ e^{-2}$ and we set ϵ equal to 1.0, 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 $6.02329 \times 10^{11}\ kJ\ mol^{-1}\ nm^{-1}$, and then scale it by 10^9 to ensure 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[34], CHARMM[35] and OPLS-aa[36].

2.2. GPU Computing

We have implemented our methods using the Open Computing Language[37] (OpenCL) parallel programming framework and executed them on an NVIDIA GPU. We used OpenCL in order to maximize the portability of our application to different GPU architectures. A scalable and efficient GPU-based algorithm should incorporate in its design effective thread deployment, instruction execution and memory access patterns[38]. Namely, the algorithm should: (a) maintain a high level of occupancy (i.e. number of resident threads against the theoretical number of GPU threads) at all times to maximize execution performance and hide global memory latency; (b) attain fine-grained data parallelism to minimize execution divergence (i.e. threads that execute different kernel instructions); (c) utilize shared memory whenever possible to reduce global memory accesses; (d) avoid scattered global memory reads and writes (i.e. uncoalesced memory accesses). Our approach takes into account all of the aforementioned design principles in order to optimize its performance.

2.3. Constructing Spatial Partitioning Structures

To efficiently compute the force spatial partitioning structures and a cutoff distance are used in order to reduce the number of interacting atom pairs considered in Equation 1. To acheive this we developed two GPU-accelerated proximity querying methods based on octrees and regular grids. Our querying methods are designed to support GPUs of different memory sizes, facilitate docking simulations of various complexities and memory footprints, and have minimal precomputation requirements. These points were addressed by utilizing regular grids and octrees interchangeably based on the size of the simulation problem, and by constructing them only for the molecule with the least number of atoms. Octrees have a smaller

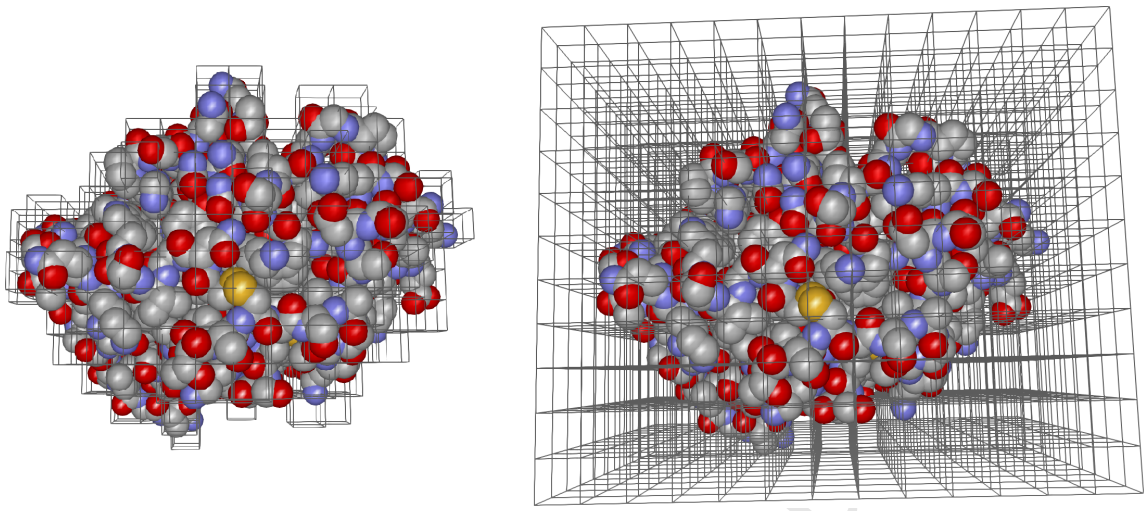


Figure 1: The molecule Trypsin subdivided with the same level of detail by the two spatial partitioning structures. Left image: The octree structure with its leaf octants. Right image: The regular grid structure with its cells. The total number of octants is far less than the total number of cells, resulting in a smaller memory footprint for the octree.

memory footprint than grids, but accessing a grid cell is a constant time operation, whereas, accessing an octree node is a logarithmic operation on the height of the octree (Figure 1). With that in mind, our hybrid approach constructs a grid if the underlying memory requirement, given by m_G, does not exceed the available GPU memory, or an octree otherwise. To compute m_G we use Equation 2,

$$m_G = \left\lfloor \frac{\ell_x}{c_g} \right\rfloor \left\lfloor \frac{\ell_y}{c_g} \right\rfloor \left\lfloor \frac{\ell_z}{c_g} \right\rfloor c_b \tag{2}$$

where m_G is the total memory required for the regular grid, ℓ_x, ℓ_y, and ℓ_z are the side-lengths of the molecule's tightest rectangular bounding box in the x, y and z axes respectively, c_g is the desired size of a grid cell side (i.e. each cell is bounded by a cube), and c_b the memory requirement in bytes of each cell. When m_G is less than or equal to the GPU's available memory, the method chooses a regular grid; otherwise it chooses an octree. All sizes are measured in Ångstrom, c_b is 12 bytes

and $c_g = nr_C$ (r_C is 1.7Å, the radius of a carbon atom), where n was determined empirically. The actual cell size used might change slightly in order to divide the bounding box into an integer number of subdivisions. Both partitioning structures are built on the CPU, and then transferred to the GPU as a 1D array of cell's or octants S. Each cell or octant defines a record which holds, among other entries, the total number of atoms assigned to it, and an index to a 1D array of atoms A. A is constructed concurrently with S and contains the ligand atoms in a sequential order that maps the order the cells/octants are indexed within S. For example, if the initial grid consists of the two cells C_a and C_b, each of which contains atoms *a*,*d*,*e*,*f* and *k*,*m*,*b*,*h* respectively, and these cells are transferred to S as C_a^{GPU} and C_b^{GPU} (i.e. S=$\{C_a^{GPU}, C_b^{GPU}\}$), then the array of atoms is formed as A=$\{a,d,e,f,k,m,b,h\}$, and the cell records as C_a^{GPU}=(1, 4) and C_b^{GPU}=(5, 4) (see Figure 2).

To construct the regular grid on the CPU we use a similar approach to Fang and Piegl[39]. We then obtain the 1D cell array S for the GPU, by looping through the grid in an x first, y second, z last order, mapping the 3D grid cell index into a 1D index, and then using it to assign the cell in the 1D array. A cell, as stated earlier, is 12 bytes and contains an index in A referencing the first atom in the set of atoms assigned to the cell (4-byte integer), the cardinality of this set (4-byte integer), a flag stating whether the cell is empty or not (1 byte), and memory-alignment padding (3 bytes) to facilitate memory-access coalescing on the GPU.

To construct an octree on the CPU, we use the algorithm described in Iakovou et.al.[40]. We then obtain the 1D octant array S for the GPU, by executing a breadth-first traversal of the tree and assigning the respective octants in the array in that order. An octant is 32 bytes, and contains an index in A referencing the first atom in the set of atoms assigned to the octant (4-byte integer), the cardinality of this set (4-byte integer), a flag stating whether the octant is a leaf or not (1 byte),

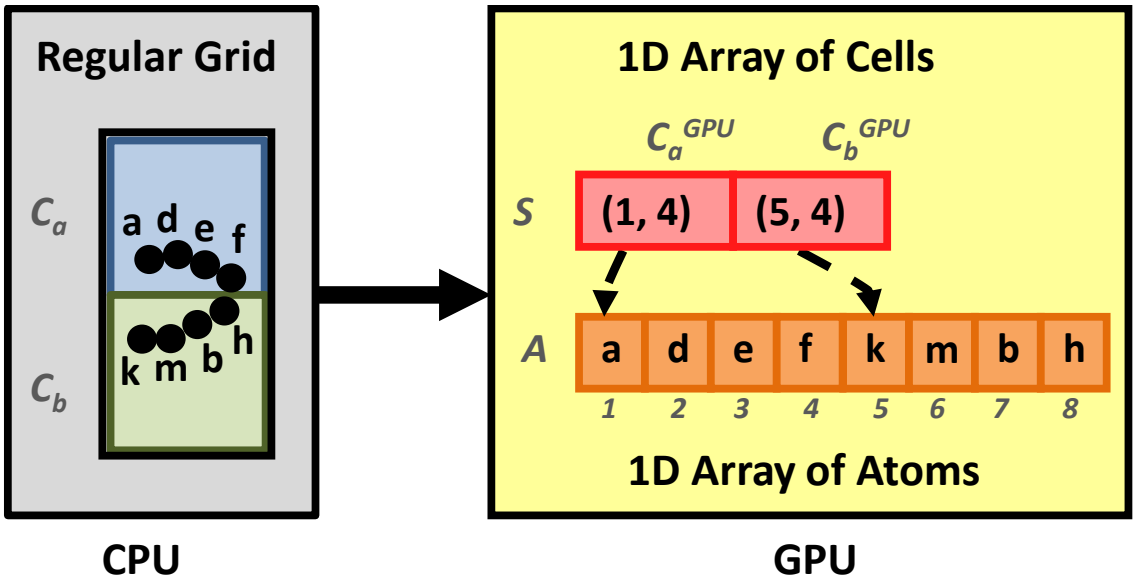


Figure 2: A 2D depiction of a regular grid built on the CPU and transferred to the GPU as a 1D array of cell records S and 1D array of atoms A. The initial grid consisted of the cells C_a and C_b containing the atoms a,d,e,f and k,m,b,h, respectively. Both cells are represented in the S array as cell records C_a^{GPU} and C_b^{GPU}. Each cell record holds the total number of atoms assigned to it (4 in both cases), and an index to the array of atoms A pointing to the first atom assigned to this cell (indices 1 and 5 in this case).

the octant's homogeneous centre coordinates (4×4-byte floats), the length of the octant's bounding-sphere radius (4-byte float), and memory-alignment padding (3 bytes). In our construction, the number of levels in the octree, L, depends on the size of the molecule, and is decided dynamically using Equation 3,

$$L = min\left(\left\lfloor log_2\left(\frac{max(\ell_x, \ell_y, \ell_z)}{c_o}\right)\right\rfloor, L_{max}\right) \tag{3}$$

where L is the octree subdivision target, c_o is the side-length of the leaf octant we are aiming for (i.e. the length of one of the bounding cube sides), and L_{max} is the maximum subdivision level our GPU-based query algorithm can support, i.e. 7 due to memory constraints. We take as numerator the maximum side because our query requires the subdivision to be uniform along all three dimensions, i.e. the octant bounding volume is a cube. L is set equal to L_{max} only when the derived level is greater than L_{max}. The side-length of the leaf-octant c_o is given by $c_o = nr_C$, where n is determined empirically. The values of the targeted leaf-octant side-lengths c_o and the actual leaf-octant side-lengths obtained after construction would differ when the value $max(\ell_x, \ell_y, \ell_z)/c_o$ is not a power of 2.

Overall, our construction strategy allows us to a) construct the grid/octree structure at the appropriate subdivision level adaptively at run time, b) reduce the memory footprint of both structures, and c) attain coalesced memory accesses during querying (since ligand atoms within the cell/octant are listed sequentially). It also helps our query kernel achieve optimum execution convergence, since nearby receptor atoms are more likely to query the same cells/octants in 3D space, access the same ligand atoms, and have their threads execute the respective kernel instructions synchronously. Given that there are no pre-processing requirements (i.e. construction of a space partitioning structure) for the receptor, our approach

can facilitate, in principle, docking problems that model receptor flexibility.

2.4. Querying Partitioning Structures and Calculating Forces on the GPU

To compute the total interaction force, the method queries the grid/octree (built for the ligand) in parallel for each receptor atom a_i individually. Each query identifies all ligand atoms within d_{cutoff} from a_i, and computes in real time the contribution of a_i to the total interaction force. The method derives the total force by accumulating these partial contributions (Figure 3). We reduce further the total computational cost of querying, using a combined viewing transformation matrix $\boldsymbol{T_{New}}$ as suggested in Iakovou et.al.[40].

The following list outlines the key execution steps of our approach.

1. Spawn a work-item (i.e. OpenCL term for thread) for every atom a_i within the largest molecule and group them into workgroups (i.e. OpenCL term for thread blocks).
2. Transform the coordinates of a_i into the local coordinates of the ligand using $\boldsymbol{T_{New}}$.
3. Execute the partitioning-structure-specific querying algorithm.
   (a) Find the set of ligand atoms within the cut-off distance to a_i.
   (b) Compute the force for all pairs in the set.
4. For all work-items in a workgroup sum their contributions to the total force $\boldsymbol{F_i^W}$, and store the result in an array $\boldsymbol{F^W}$ of length equal to the number of workgroups.
5. Sum the partial forces in $\boldsymbol{F^W}$to obtain the total force $\boldsymbol{F_{Tot}^W}$.

Steps 1), 2), 4) and 5) are steps common to both partitioning structures. The execution flow differs in Step 3) because our method queries the regular grid and

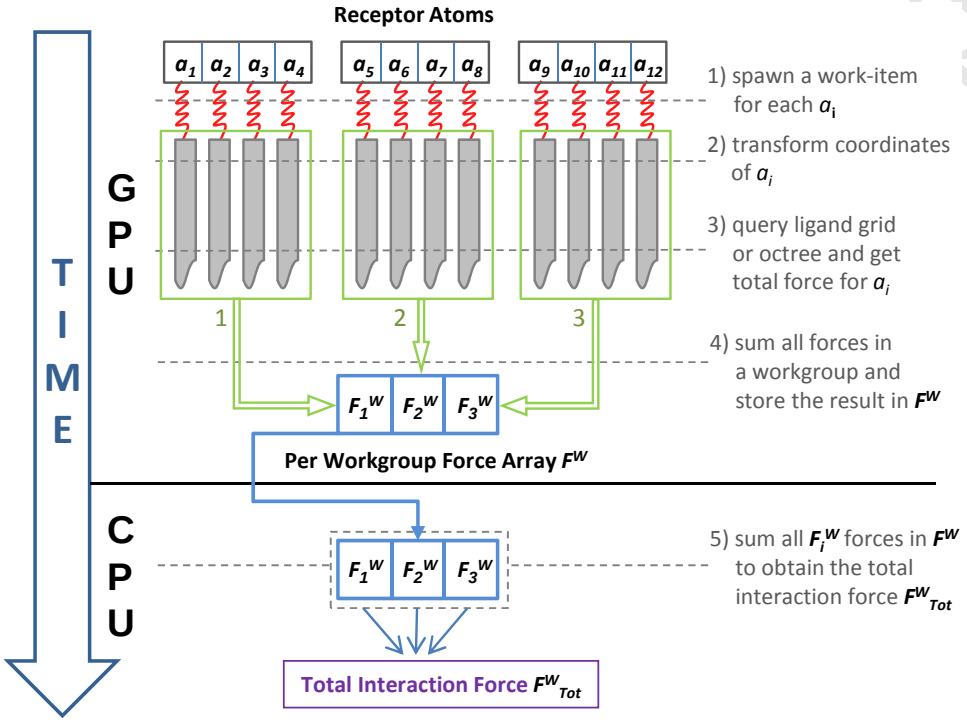


Figure 3: A visualization of our GPU-accelerated force calculation approach, illustrating the main execution steps, and the processing unit (i.e. GPU or CPU) that executes them. The method starts by deploying on the GPU one work-item (red springs) for each receptor atom a_i (12 receptor atoms in this case), and grouping these work-items in workgroups (the 3 green boxes with 4 work-items each). Each work-item executes our proximity querying/force calculation kernel (grey semi-rectangular shape) in parallel, within its workgroup, and computes the force contribution of a_i to the total force (execution steps 1-3). The first work-item in each workgroup accumulates these force contributions from all work-items in the group, and stores the result F_i^W in a global number-of-workgroups-long force array F^W (execution step 4). Array F^W is transferred back to the CPU, where its entries are accumulated to produce the total interaction force F_{Tot}^W (execution step 5).

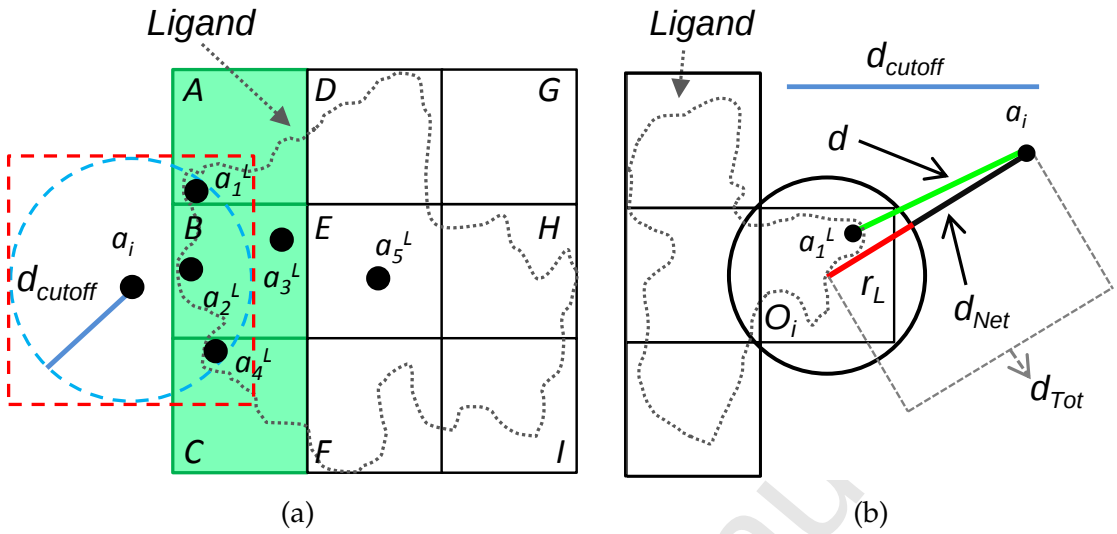


Figure 4: A conceptual 2D visualization of our proximity querying strategies. (a) Querying the regular grid. The method uses the cut-off distance d_{cutoff} to form a bounding cube (red dashed square) centred on receptor atom a_i. Using the cube's min/max coordinates, the query identifies all grid cells (green cells A, B and C) intersecting the cube and produces a search range. The method calculates an interatomic distance d between a_i and each of the ligand atoms contained within these cells (i.e. ligand atoms a_1^L, a_2^L, a_3^L and a_4^L), but computes the total force only for those atom pairs with $d \leq d_{cutoff}$ (in this case pairs $a_i a_1^L$, $a_i a_2^L$, $a_i a_4^L$, since atom a_3^L is not within the cut-off radius). (b) Querying the octree. The coordinates of the receptor atom a_i are tested against octant O_i. The method calculates d_{Tot} (i.e. distance between the octant centre and a_i) and subtracts it from r_L (i.e. radius of the octant's bounding sphere) to obtain d_{Net} (i.e. net distance). If $d_{Net} \leq d_{cutoff}$ and O_i is not a leaf octant then the method traverses the children of O_i in the same manner. When O_i is a leaf octant (as in the case shown), the method calculates an interatomic distance d between a_i and each of the atoms indexed by O_i (a_1^L in this case), but again computes the force only for those atom pairs with $d \leq d_{cutoff}$ (i.e. pair $a_i a_1^L$).

the octree differently. To query the grid the method obtains first a search range and then indexes the cells within this range; whereas, to query the octree it performs a combination of depth-first and breadth first traversals on the octants starting from the root (Figure 4).

In the next two paragraphs we describe our GPU-accelerated, regular grid and octree-based force calculation algorithms.

2.4.1. Querying and Calculating Forces Using a Regular Grid

We utilize the random access property of regular grids to determine in parallel the subset of grid cells containing those ligand atoms within the cut-off, and then compute the total force on this set. We begin by executing one work-item for each receptor atom, arranged in workgroups of 256 items each. Using its global ID, each work-item accesses the underlying receptor atom and updates the atom's coordinates with $\boldsymbol{T_{New}}$. Based on the new coordinates we then identify our search region of grid cells using Algorithm 1, ***GetSearchRange*** (provided as supplementary information).

Initially, Algorithm 1 computes the tightest bounding cube of a sphere with centre equal to the coordinates of receptor atom a_i, and radius equal to d_{cutoff}. It then uses the cube's minimum and maximum coordinates to derive a minimum/-maximum search range for the grid along the three dimensions x, y, and z (Figure 4a). Using this range, it loops through the grid cells and for all ligand atoms a_i^L within each cell it checks whether or not the interatomic distance between the receptor and ligand atoms is within the cut-off. It then computes the forces, for all atom pairs that pass this test, and accumulates these forces in force vector $\boldsymbol{f_i}$. As such, vector $\boldsymbol{f_i}$ holds (upon loop termination) the force contribution of the given receptor atom a_i to the total force. Each work-item saves $\boldsymbol{f_i}$ within a local array of force values, and waits on a group-synchronization primitive. When all group work-items are synchronized, the first work-item in the workgroup sums up the values within the local array, and stores the result $\boldsymbol{F_i^W}$ in a group-specific global array of force values $\boldsymbol{F^W}$. We accumulate the entries in $\boldsymbol{F^W}$ to compute the total force. In almost all practical cases the size of this array is very small (e.g. even for one million atoms the size is 1000000/256=3907). As such we perform this

accumulation on the CPU since it can perform this summation faster than the 0.2ms overhead (time from submission to start) required by NVIDIA's OpenCL drivers to deploy a kernel on the GPU. The size of the local array equals the workgroup size (i.e. 256), whereas the size of the global array equals the number of workgroups, i.e. $\left\lceil \frac{receptoratoms}{256} \right\rceil$. A work-item indexes these local and global arrays using its local (block-specific) and workgroup IDs, respectively. Overall, the use of the local and global arrays allows us to perform the majority of force calculations on the GPU in a memory-coalesced fashion, and hence optimize the performance of our method. Algorithm 2, ***GPUQueryRegularGrid*** (provided as supplementary information), outlines the aforementioned key execution steps.

2.4.2. Querying and Calculating Forces Using an Octree

Similar to our grid-based algorithm, our octree querying algorithm begins by executing a work-item per receptor atom, in workgroup sizes of 256, and updates the coordinates of the receptor atoms with $\boldsymbol{T_{New}}$. It then begins the tree traversal loop by assigning the root as the current octant, and looping through all of its children. Normally, octree traversal is done recursively starting from the root octant, but OpenCL does not support recursive control flow. Even if it did support recursion[41], such a query would be prone to high execution divergence (with substantial performance penalties) since the recursive branching to the child octants would need to be made independently by each work-item. To address this we developed a stack-based, octree querying method that emulates programmatically recursive behaviour, while minimizing execution divergence. The method traverses the tree iteratively utilizing a stack to mimic recursive calls. The stack is defined as an array of octant indices, and is allocated in private memory by each work-item (since OpenCL does not support dynamic memory allocation). We set the size

of the stack equal to fifty six four-byte integers (7 octree levels and 8 octants for each level), which can accommodate octree traversals of height seven (which is our maximum subdivision level and a good balance point between subdivision and total stack memory requirements). Using this stack, the tree traversal loop begins by checking (in a breadth-first manner) whether the net distance d_{Net} between the receptor atom and the child octants is within cut-off or not. To compute d_{Net} we apply Equation 4,

$$d_{Net} = d_{Tot} - r_L \tag{4}$$

where d_{Tot} is the total distance between the octant centre and the atom, and r_L is the radius of the octant's bounding sphere. If $d_{Net} \leq d_{cutoff}$ and the child octants are leafs, it loops through all atoms indexed by these octants, calculates their interatomic distance d with the receptor atom, and accumulates the force (in a similar way to our regular grid method) only for those receptor/ligand atom pairs with $d \leq d_{cutoff}$ (Figure 4b). Otherwise it sets the first one of these octants (in a depth-first manner) as current, and pushes the remaining ones onto the stack in reverse order. When the downward tree traversal comes to an end (i.e. the index of the current octant is -1), the algorithm pops an octant off the stack and repeats the loop. When the stack becomes empty the traversal loop terminates, and the algorithm calculates the total force on the CPU the same way as described in our grid-based force calculation method. Each work-item, regardless of its traversal path, executes the same loop repetitively until it has no more octants to traverse. Hence, for a number of iterations the work-items (especially those indexing receptor atoms nearby in 3D space) will be executing the same kernel instructions, which allows our algorithm to achieve substantial execution convergence during octree traversals. Algorithm 3, ***GPUQueryOctree***

(provided as supplementary information), describes the main steps of our octree-based approach.

3. Results

We have implemented our approach using Visual C++ and OpenCL 1.1, and integrated it within our haptics-assisted interactive rigid-docking application (Figure 5). We conducted a series of experiments in order to benchmark the performance of our approach (against demanding simulation loads), compared it to a current CPU-based implementation, and measured its efficiency during interactive rigid-docking simulations on known complexes. We executed all tests on a 2.93GHz Intel Core i7 PC running under a 64bit version of Windows 7 with an NVIDIA GTX580 GPU. The PC was equipped with 8GB RAM, and the GPU with 1.5GB RAM. We used the 3DOF Geomagic Touch haptic device, formerly known as the Phantom Omni from SensAble Technologies. For the purpose of benchmarking and GPU-CPU performance comparisons, we used arbitrary force parameters, as we were only interested in timing their force-computation. For the haptics-assisted rigid-docking simulations, however, we used Gromacs' pdb2gmx tool in order to obtain the actual Gromos54a7 force field topology/nonbonded parameters file, and add the necessary hydrogens. Specifically, for each one of these molecules we executed the following command,

```
pdb2gmx -f xxxx.pdb -o gmx_xxxx.pdb -p gmx_xxxx.top
        -ff gromos54a7 -ignh -water none -merge all
```

where xxxx is the molecule's pdb code (information about this command can be found in the Gromacs manual[33]).

Figure 5: Conducting an interactive rigid docking simulation with proteins GroEL (larger molecule) and GroES (smaller molecule) and the 3DOF Geomagic Touch haptic device. Both molecules are defined in the PDB file with accesion code 1GRU where they are in a bound conformation. The user controls GroES and feels the interaction forces using the haptic device.

3.1. Benchmarking Experiments

We conducted benchmarking experiments to measure the scalability of our approach, and identify its limitations. To achieve that, we devised and subjected our approach to various artificial docking simulations of demanding computational workloads and different molecular complexities (comprising up to two hundred thousand atoms each). At this stage emphasis was given in finding those molecules that can stress test the two proximity querying algorithms effectively. Since proximity queries are sensitive to the atom granularity of the underlying cells/octants, we selected molecules with different sizes and shapes (e.g. compact, extended). Although unrealistic, we also allowed the molecules to overlap in order to increase the number of interacting atom pairs, and attain sufficient, upper-bound, performance indicators. In these simulations both molecules were modelled as rigid structures.

We used the molecules *Alcohol Dehydrogenase* dimer (PDB code: 1ADG), *Aspartate Carbamoyltransferase*, (1AT1), *GroEL-E434K Mutant* (2YEY) and *Clathrin* (1XI4), as defined in their respective Protein Data Bank[42] (PDB) files (Figure 6). Using these proteins we generated the artificial protein-protein docking test cases 1ADG-1ADG (i.e. 1ADG with 1ADG), 1AT1-1AT1, 2YEY-2YEY and 1XI4-1XI4. For each test case we ran seven rigid docking simulations using regular grids of different cell sizes (we used c_g values equal to nr_C, where n=1,2,..7), and another seven simulations using octrees of subdivision levels L, where L=1,2,..7. For each of these test cases we also created a 4×4 matrix that specified the position and orientation of the ligand such that the ligand overlapped with the receptor, and generated a substantial set of interatomic interactions to benchmark sufficiently our approach (these matrices are provided as supplementary information). For each simulation we recorded 10000 different response times (i.e. a simulation time

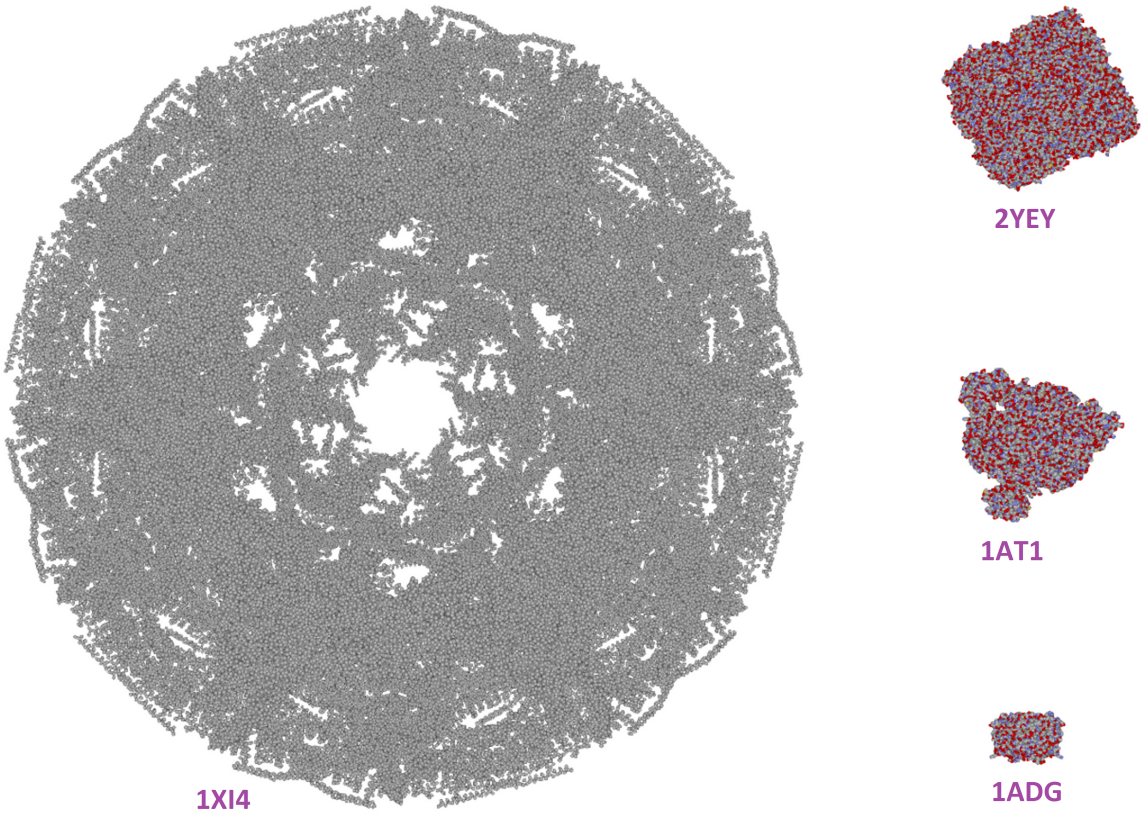


Figure 6: The four molecules used in our benchmarking experiments, showing their relative sizes. 1XI4 is the largest one with 184k atoms, and a bounding box with largest axis of 747.22Å in z (see Table 1).

of about 10ms), and computed the percentage of those responses found below 1ms, within 1-2ms(inclusive), within 2-4ms and above 4ms (Figures 7c and 7d), since there are reports suggesting that acceptable haptic refresh rates in some cases can go as low as 250-300Hz[43, 44]. We report test-case/construction specific data (Table 1) and simulation specific data (Figures 7a-7d and supplementary Tables 1 and 2). Moreover, we set d_{cutoff}=8Å in all tests.

The results show that the interaction forces were updated within the 2ms time constraint consistently, in the majority of the simulations, regardless of the querying

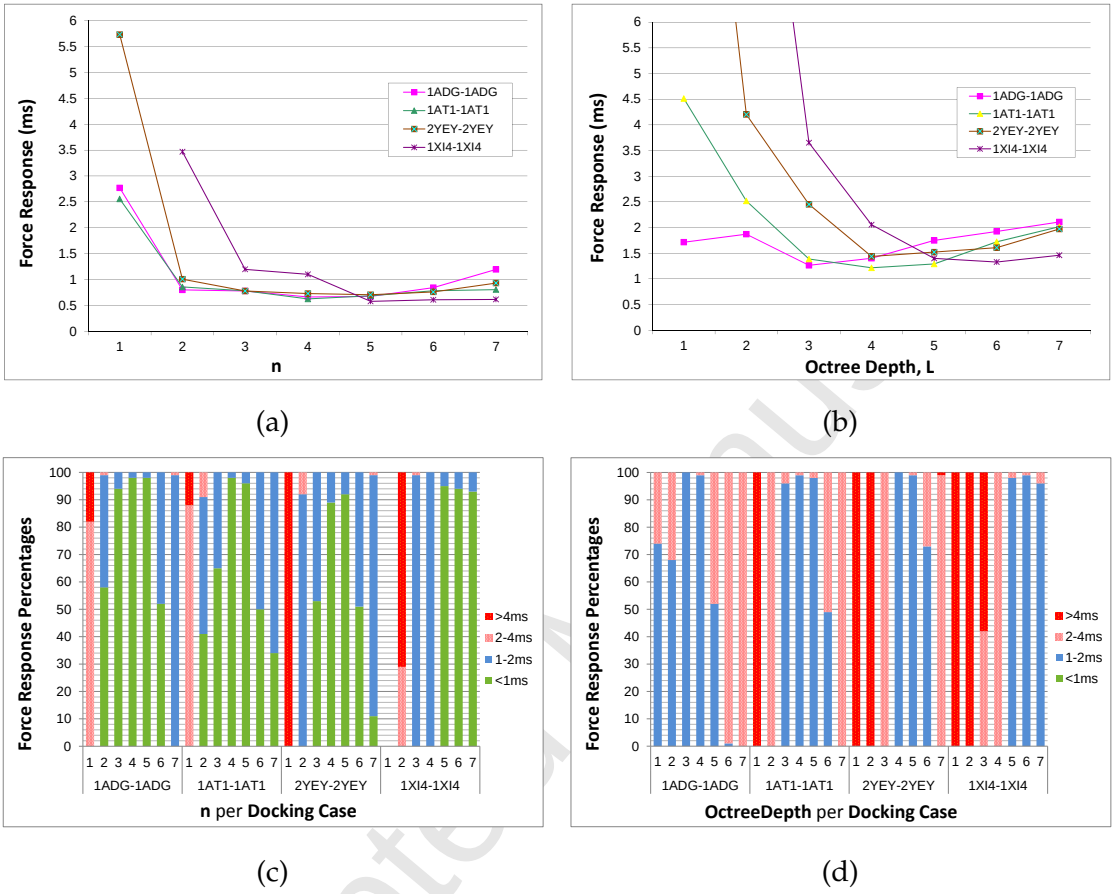


Figure 7: Benchmarking the two GPU-accelerated force calculation methods using the four artificial protein-protein docking cases 1ADG-1ADG (where 1ADG is the PDB code), 1AT1-1AT1, 2YEY-2YEY and 1XI4-1XI4. Each test was repeated 10000 times, and all response times were calculated based on more than 20K interacting atom pairs. (a) The best force response times obtained using regular grids constructed with c_g values equal to nr_C, where n=1,2,..7 and r_C is the radius of a carbon atom. (b) The best force response times obtained using octrees at depth levels 1-7. The force response times for test cases 2YEY-2YEY and 1XI4-1XI4, at depths 1 and 1-2 respectively, are not shown here (to improve graph readability). The times for these test cases were 14.92ms for 2YEY-2YEY, and 74.11 ms (level 1) and 12.96ms (level 2) for 1XI4-1XI4. (c) The percentage of those 10000 response times found below 1ms, within 1-2ms(inclusive), within 2-4ms and above 4ms (for each test case), obtained using the same regular grids as in (a). (d) The percentage of those 10000 response times found below 1ms, within 1-2ms(inclusive), within 2-4ms and above 4ms (for each test case), obtained using the same octrees as in (b).

Table 1: Molecule specific information used for the construction of both partitioning structures. The table lists the molecule's PDB code, the number of atoms comprising each molecule, and the molecule's largest bounding box dimension.

Molecule	# of heavy atoms	Bounding Box Largest Side (Å)
1ADG	7046	112.10
1AT1	21318	150.51
2YEY	53984	184.50
1XI4	183600	747.22

method used. Moreover, there was at least one simulation in each test case, under which the grid-based method delivered sub-millisecond force response times more than 90% of the time (e.g. at $5r_C$). Similarly, at octree levels 3, 4, 4 and 6 under test cases 1ADG-1ADG, 1AT1-1AT1, 2YEY-2YEY and 1XI4-1XI4 respectively, almost 90% of the force responses were computed by the octree-based method in less than 2ms. In all four cases, the grid-based method attained force responses in the range of 0.58-0.71ms, whereas the octree-based method attained force responses in the range of 1.22-1.44ms. In theory, the approach could maintain such force updates throughout a simulation, if given exclusive use of the CPU/GPU resources. In practice however, we observed fluctuations between the best and worst response times (in all simulations), reaching in some instances a difference of up to 1.5ms. We attribute these performance fluctuations to intervening CPU/GPU workloads (e.g. background processes, display rendering). We set the grid-based method as the first choice, since it performed consistently faster than the octree-based method, and use the latter for cases in which the available GPU memory cannot accommodate the construction of a regular grid. The dimensions of the grid-cell/leaf-octant also influenced the performance of the querying method. In the case of grid-based querying, a cell size c_g=$5r_C$ appears to construct those grids that can facilitate efficient query responses (Figure 7a). Slightly better responses were

attained in 1ADG-1ADG and 1AT1-1AT1 at c_g=4r_C, however these performance differences are insignificant. As such, a c_g=5r_C could be used in our approach as a universal subdivision criterion for regular grids. We also use this c_g criterion in Equation 2 in order to decide which partitioning structure to use in our queries. In the case of octree-based querying, we identified, initially, the subdivision levels L with the fastest response times (Figure 7b), used the formula $max(\ell_x, \ell_y, \ell_z)/2^L$ to obtain the actual side-length of the leaf octants, and related this to r_C (i.e. found those multiples of r_C that would cause our method to construct a tree of level L). Using Equation 3 we determined L for each value of n by setting c_o equal to nr_C. Though in many cases this relation was not one-to-one (e.g. in 1ADG-1ADG n=5,6, or 7 all resulted in L=3), it did identify the correct subdivision level (bold entries in supplementary Table 2) when c_o=5r_C. In all cases, the octrees occupied less memory, at the respective leaf/cell sizes than the regular grids (supplementary Tables 1 and 2), and their byte difference increased proportionally to the simulation workload (e.g. more than a six fold difference in 1XI4-1XI4). In almost all simulations, with an exception of 1XI4-1XI4 at cell size equal to 1.7Å, our approach was able to construct both partitioning structures on the GPU. We were unable to find a PDB file containing a molecule large enough for an octree to be chosen over a regular grid for the GPU used. To overcome this, we created a test molecule from four 1XI4 molecules aligned along the main diagonal of the bounding box so as to maximise its volume (referred to as 4_1XI4). This artificial structure comprised approximately 735K atoms, and was bounded by a cube with side length of 2873.69Å. Using 4_1XI4, we generated and benchmarked the docking case 4_1XI4-4_1XI4, at a targeted octant size c_o=5r_C. For this test case our approach utilized an octree (of 1.4MB), since the GPU could not allocate the 463.5MB of memory needed for the regular grid. Again, we overlapped both molecules along their longest surface, and generated a

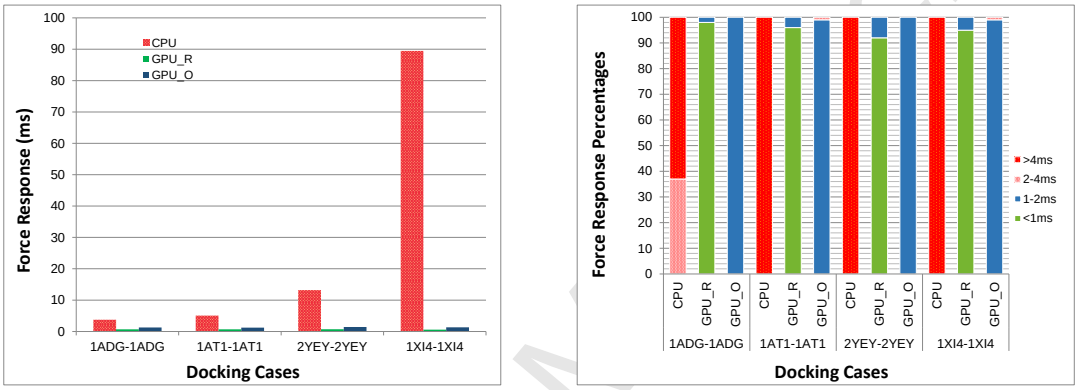
set S_{Pairs} of 27151 interacting atom pairs. The octree-based method averaged force response times at 4.6ms, indicating that simulation cases of such size pose an upper limit to our approach.

Finally, the results also show that both querying methods scale very well to the size of the interacting molecules. In all four test cases, both methods obtained similar response times regardless of the underlying molecule sizes. The one-to-one work-item-per-atom strategy adapts very well to the Single Instruction Multiple Data execution model of the GPU, and thus utilizes efficiently the GPU's computational resources.

3.2. GPU-CPU Comparisons

We compared our two GPU-accelerated methods to our reported CPU octree-based force-calculation method[40], which can facilitate the interactive, rigid docking of large scale systems and is optimized for the CPU. The purpose of these tests is to identify/measure the performance gains attained by the GPU methods over the CPU method. We did not compare our approach to other current CPU-based approaches (e.g. brute force, pre-computed force-grid based etc.) since reportedly they cannot manage molecular systems of more than 3000 atoms each [9].

Using the same four benchmarking test cases (1ADG-1ADG, 1AT1-1AT1, 2YEY-2YEY and 1XI4-1XI4), cut-off distance and number of iterations (i.e. 10000), we tested the CPU-based approach to obtain comparable results. In these tests, the sub-division levels of all octrees were set equal to 4, as stated by Iakovou et. al [40]. We then compared and reported for each test case and for each querying method (CPU, GPU-Regular grid, and GPU-Octree) the best response times obtained (Figure 8a), and the best response-time intervals for these 10000 iterations (i.e. $<$1ms, 1-2ms, 2-4ms, $>$4ms) as percentages (Figure 8b, see Figure legend for further details).



(a) (b)

Figure 8: GPU-CPU force response comparisons between our two GPU-accelerated force-calculation methods (i.e. regular grid/GPU-R and octree/GPU-O) and the CPU-based force-calculation method reported in Iakovou et. al.[40] All three methods were tested on the four artificial protein-protein docking cases 1ADG-1ADG (where 1ADG is the PDB code), 1AT1-1AT1, 2YEY-2YEY and 1XI4-1XI4. Each test was repeated 10000 times, and all response times involved more than 20K interacting atom pairs. (a) The best response times obtained by each force calculation method for each docking case. (b) The best response-time intervals (as percentages) for the 10000 iterations (i.e. <1ms, 1-2ms, 2-4ms, >4ms) obtained by each force calculation method for each docking case. The best response-times were calculated using GPU-based grids of cell size $c_g = 5r_c$, GPU-based octrees of Level, L, given by Equation 3 with $c_o = 5r_c$ and CPU-based octrees of Level, L=4.

The results show that there were significant performance gains when utilizing the GPU-based methods over the CPU-based method, especially as the sizes of the molecules increased, due to the high levels of GPU occupancy/parallelism attained. Specifically, for the 1ADG-1ADG case (comprising 7K atoms each) both GPU methods outperformed the CPU method by 5×, and by 90× for the very large test case 1XI4-1XI4 (180k of atoms each). In all cases and for more than 90% of the trials, the regular-grid based method (GPU-R) was able to provide force updates in less than 1ms. The octree-based method (GPU-O) although slower still updated consistently the forces in less than 2ms. On the other hand, the CPU-based method failed to satisfy the 2ms time constraint in every case. Overall, both GPU-based methods improve substiantially upon the CPU-based method and, as such, can be applied to haptics-assisted, interactive docking simulations of very large systems, which would have been impossible otherwise.

3.3. Haptics-assisted Interactive Rigid-Docking Simulations

In addition to benchmarking, we tested the performance under actual rigid-docking scenarios. The purpose of these simulations was twofold: a) to measure force-response times under real docking examples during which atom-overlapping cannot occur, and b) to sense the rendering quality (e.g. stability, smoothness) of the resulting interactions on the haptic device. We used four well known complexes, and conducted six rigid-docking simulations related to protein-protein and protein-drug docking. We used the complexes of *Epidermal Growth Factor* (EGF) with *EGF receptor* (EGF*r*), *Bovine Pancreatic Trypsin Inhibitor* (BPTI) with *Trypsin*, anticancer drug *BAY43-9006* (sorafenib, Nexavar) with cancer target *B-raf*, and *GroES* with *GroEL* as defined in the PDB files 1NQL, 3OTJ, 1UWH, and 1GRU, respectively. The first three complexes are examples of protein-protein

docking, whereas the fourth complex is an example of protein-drug docking. Each file contained the structures of the receptor and the ligand in their bound conformation. For each, we created two new separate PDB files, one for the receptor and one for the ligand. Using Gromacs 4.6.2 and the 8 PDB files, we obtained their respective Gromos54a7 non-bonded force parameters for all molecules except the drug sorafenib, for which, we used the PRODRG[45] server from the University of Dundee (http://davapc1.bioch.dundee.ac.uk/programs/prodrg/). We ran the simulations using the ligand as the haptic interface to the virtual world. To capture the relation between force response times and number of interacting atom pairs we conducted one simulation per complex. Using the haptic device, we moved the ligand around the receptor, sensed the interaction forces on the device, and guided the molecules back to their binding conformation (as defined in the original PDB file). Each simulation ran for approximately one minute during which we recorded at 10 millisecond intervals the force response times and the number of interacting atom pairs. To identify how the rendering quality relates to the number of interacting atom pairs we repeated the simulations for complexes B-raf-sorafenib and GroEL-GroES (for the smallest and largest ligand) for approximately 6ms (i.e. just moving the ligand around the receptor), and recorded at each haptic frame the total force, and the number of interacting atom pairs. Table 2 gives structural information on the molecules used. All force queries were executed using a regular grid with c_g=5r_C (8.5Å), and a value of 8Å as the cut-off distance. Figures 9 and 10 illustrate graphically the results obtained from these simulations.

The results show that all interaction forces were calculated in less than one millisecond throughout the simulation period and for varying numbers of atom pairs. In general, the interaction forces displayed and felt on the haptic device were fairly smooth, without any device-induced instabilities and vibrations. How-

Table 2: Structural information for the eight molecules used in our real-time docking simulations. The table lists the molecule's PDB code, the number of atoms comprising each molecule, and the molecule's largest bounding box dimension.

Molecule	# of heavy atoms	Bounding Box Largest Side (Å)
sorafenib	48	17.20
EGF	483	41.50
BPTI	604	45.60
TRYPSIN	2094	58.70
B-raf	5376	83.10
EGF*r*	5836	113.09
GroES	6321	102.70
GroEL	66451	374.17

ever, the rapid change in the magnitude of the force during the simulation of the GroEL-GroES complex (especially when the molecules were in contact) caused device jittering which could be perceived by the user as unstable force rendering. Force smoothing methods such as the one proposed by Bolopion et. al.[46] could address this. Response times did not drop below 0.2ms, even when there were no interactions, because NVIDIA's OpenCL drivers induce a 0.2ms kernel deployment overhead. Furthermore, in many instances the response times for the same set of atom pairs were found to fluctuate by up to 0.45ms. Like the benchmarking experiments, these fluctuations reflect delays introduced by interfering system processes.

4. Discussion and Conclusion

We have described methods and implementation details for haptic-assisted interactive docking. The approach utilizes effectively the many-core processing capabilities of modern GPUs, the space partitioning properties of regular grids and octrees, and two efficient proximity querying algorithms based on these partition-

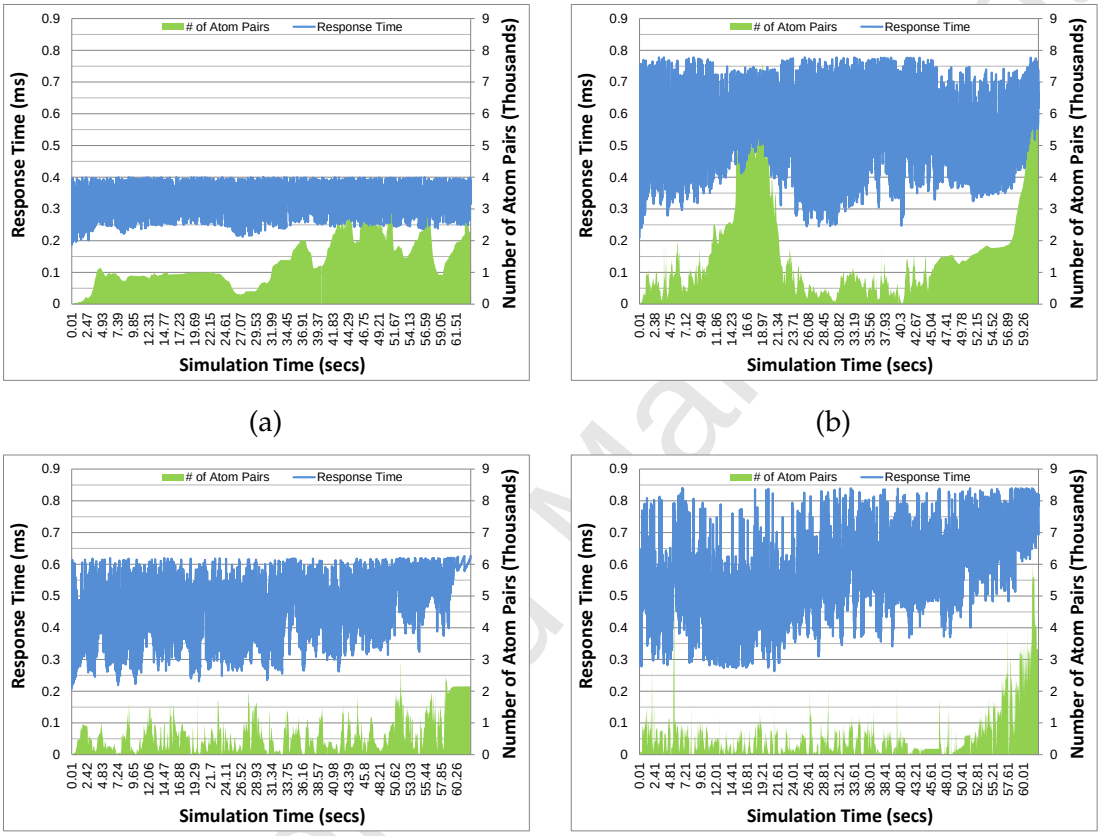


Figure 9: A haptics-assisted rigid-docking simulation between: (a) the drug molecule *sorafenib* and the receptor protein *B-raf*; (b) protein *BPTI* and the receptor protein *Trypsin*; (c) protein *EGF* and the receptor protein *EGFr*; (d) protein *GroES* and the receptor protein *GroEL*. The graph depicts the force response times attained, at 10ms intervals, and the respective sets of interatomic interactions accounted for by the approach during the simulation.

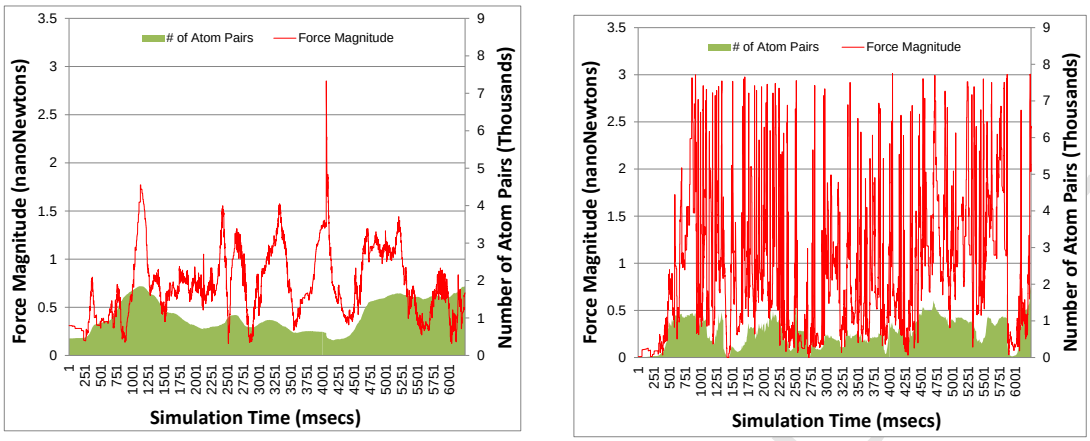


Figure 10: A haptics-assisted rigid-docking simulation between: (a) the drug molecule *sorafenib* and the receptor protein *B-raf*; (b) protein *GroES* and the receptor protein *GroEL*. The graph depicts the force magnitudes (scaled to nanoNewtons) attained at each haptic frame, and the respective sets of interatomic interactions accounted for by the approach during the simulation.

ing structures. A major issue in haptics-assisted docking is the 2ms force-update constraint, required for smooth and stable force-feedback. Current interactive approaches can achieve such refresh rates only for molecules comprising up to a couple of thousand of atoms each. We presented a GPU-accelerated force calculation approach that can effectively address the 2ms constraint for interactive docking simulations of molecules comprising hundreds of thousands of atoms each, thus enabling the haptics-assisted study of protein-protein interactions. When compared to other CPU-based force calculation approaches, our approach was up to 90 times faster. In addition, the method pre-computes a space partitioning structure for the smaller molecule (ligand) only, meaning there would be no additional overhead in the force calculation when receptor atoms move due to conformational change. Providing the new positions of the receptor atoms are calculated sufficiently quickly, receptor flexibility could be modelled. One approach already taken

in haptics for modelling flexibility is to use an elastic network model[47].

As it stands, GPU memory does not seem to be an issue for either method. Furthermore, it appears that modern GPUs can accommodate the memory requirements of the grids used in almost all practical, haptics-assisted docking cases. However, for cases where (a) the GPU has limited memory specifications (e.g. less than 256MB memory), (b) the GPU performs at the same time other memory hungry tasks (e.g. ray tracing, texture mapping), and/or (c) the simulation involves systems that cannot be accommodated by a regular grid, our octree-based methods represents an effective alternative.

We have presented a scalable, GPU-parallelizable force calculation approach that overcomes the computational limitations of previous approaches (e.g. precomputed force grids), and can compute the intermolecular forces of docking between very large molecules, within haptic refresh rates. It computes the total force in real time for all interatomic interactions within a cut-off distance. In MD simulations variation of the cut-off distance can have significant effects. As we use an 8 Å cut-off distance it is expected that any inaccuracy will arise from the longer range electrostatic interactions rather than from the van der Waals. We have tried a docking experiment with BPTI on the receptor trypsin where instead of an 8 Å cut-off distance all atom pairs were included in the force calculation. We could not find, within our present system, any perceptible difference.

Currently only forces are perceived through the haptic device but torques obviously play a crucial role in the docking process. Torque will rotate a ligand relative to the receptor helping to orient it correctly for docking. Affordable haptic devices do not allow the user to feel torques although they allow the user to rotate objects. A partial solution is to give a graphical depiction of the torque.

We have implemented and tested these methods with docking simulations of

different molecular shapes and sizes. They achieved the targeted force update rates (less than 2ms) in all test cases, which ranged from standard protein-drug to very large protein-protein interaction problems. Our work also demonstrates that the inherent execution parallelism of GPUs can benefit haptics-assisted docking systems, and help overcome early computational barriers (e.g. pre-computed grids, rigid molecules) that limited their applicability.

[1] G. M. Morris, R. Huey, W. Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, A. J. Olson, Autodock4 and autodocktools4: Automated docking with selective receptor flexibility, J. Comput. Chem. 30 (16) (2009) 2785–2791.

[2] N. Moitessier, P. Englebienne, D. Lee, J. Lawandi, Corbeil, CR, Towards the development of universal, fast and highly accurate docking/scoring methods: a long way to go, Br. J. Pharmacol. 153 (S1) (2008) S7–S26.

[3] M. Eisenstein, E. Katchalski-Katzir, On proteins, grids, correlations, and docking, C. R. Biol. 327 (5) (2004) 409–420. doi:10.1016/j.crvi.2004.03.006.
URL http://www.ncbi.nlm.nih.gov/pubmed/15255472

[4] E. Yuriev, M. Agostino, P. A. Ramsland, Challenges and advances in computational docking: 2009 in review, J. Mol. Recognit. 24 (2) (2011) 149–164. doi:10.1002/jmr.1077.
URL http://dx.doi.org/10.1002/jmr.1077

[5] M. Ouhyoung, Force display in molecular docking, Ph.D. thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, tR90-004 (2 1990).

[6] W. Humphrey, A. Dalke, K. Schulten, Vmd: visual molecular dynamics, J. Mol. Graphics 14 (1) (1996) 33–38.

[7] M. Stocks, S. Hayward, S. Laycock, Interacting with the biomolecular solvent accessible surface via a haptic feedback device, BMC Struct. Biol. 9 (1) (2009) 69–75.

[8] Computing power revolution and new algorithms: Gp-gpus, clouds and more: general discussion, Faraday Discuss. 169 (2014) 379–401. doi:10.1039/C4FD90021A.
URL http://dx.doi.org/10.1039/C4FD90021A

[9] A. Ricci, A. Anthopoulos, A. Massarotti, I. Grimstead, A. Brancale, Haptic-driven applications to molecular modeling: state-of-the-art and perspectives, Future Medicinal Chemistry 4 (10) (2012) 1219–1228.

[10] E. Subasi, C. Basdogan, A new haptic interaction and visualization approach for rigid molecular docking in virtual environments, Presence: Teleoperators and Virtual Environments 17 (1) (2008) 73–90.

[11] P. Bivall, S. Ainsworth, L. A. Tibell, Do haptic representations help complex molecular learning?, Science Education 95 (4) (2011) 700–719.

[12] P. B. Persson, M. D. Cooper, L. A. Tibell, S. Ainsworth, A. Ynnerman, B.-H. Jonsson, Designing and evaluating a haptic system for biomolecular education, in: Virtual Reality Conference, 2007. VR'07. IEEE, IEEE, 2007, pp. 171–178.

[13] M. Minsky, O.-y. Ming, O. Steele, F. P. Brooks Jr, M. Behensky, Feeling and seeing: issues in force display, ACM SIGGRAPH Computer Graphics 24 (2) (1990) 235–241.

[14] K. B. Shimoga, A survey of perceptual feedback issues in dexterous telema-

nipulation. ii. finger touch feedback, in: Virtual Reality Annual International Symposium, 1993., 1993 IEEE, IEEE, 1993, pp. 271–279.

[15] R. Ellis, O. Ismaeil, M. Lipsett, Design and evaluation of a high-performance haptic interface, Robotica (1996) 321–327.

[16] H. Nagata, H. Mizushima, H. Tanaka, Concept and prototype of protein–ligand docking simulator with force feedback technology, Bioinformatics 18 (1) (2002) 140–146.

[17] N. Férey, J. Nelson, C. Martin, L. Picinali, G. Bouyer, A. Tek, P. Bourdot, J.-M. Burkhardt, B. F. Katz, M. Ammi, et al., Multisensory vr interaction for protein-docking in the corsaire project, Virtual Reality 13 (4) (2009) 273–293.

[18] X. Hou, O. Sourina, Haptic rendering algorithm for biomolecular docking with torque force, in: Cyberworlds (CW), 2010 International Conference on, IEEE, 2010, pp. 25–31.

[19] O. Sourina, J. Torres, J. Wang, Visual haptic-based biomolecular docking and its applications in e-learning, in: Transactions on Edutainment II, Springer, 2009, pp. 105–118.

[20] N. Pattabiraman, M. Levitt, T. Ferrin, R. Langridge, Computer graphics in real-time docking with energy calculation and minimization, J. Comput. Chem. 6 (5) (1985) 432–436.

[21] F. P. Brooks Jr, M. Ouh-Young, J. J. Batter, P. Jerome Kilpatrick, Project grope - haptic displays for scientific visualization, in: ACM SIGGraph computer graphics, Vol. 24, ACM, 1990, pp. 177–185.

[22] O. B. Bayazit, G. Song, N. M. Amato, Ligand binding with obprm and user input, in: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, Vol. 1, IEEE, 2001, pp. 954–959.

[23] Y.-G. Lee, K. W. Lyons, Smoothing haptic interaction using molecular force calculations, Computer-Aided Design 36 (1) (2004) 75–90.

[24] P. Bivall, Touching the essence of life: Haptic virtual proteins for learning, Ph.D. thesis, Linköping University, Linköping, Sweden (2010).

[25] S. K. Lai-Yuen, Y.-S. Lee, Computer-aided molecular design (camd) with force-torque feedback, in: Computer Aided Design and Computer Graphics, 2005. Ninth International Conference on, IEEE, 2005, pp. 199–204.

[26] E. Subasi, C. Basdogan, A new approach to molecular docking in virtual environments with haptic feedback, in: Proceedings of EuroHaptics Conference, 2006, pp. 141–145.

[27] A. M. Wollacott, K. M. Merz Jr, Haptic applications for molecular structure manipulation, J. Mol. Graphics Modell. 25 (6) (2007) 801–805.

[28] B. Daunay, A. Micaelli, S. Régnier, 6 dof haptic feedback for molecular docking using wave variables, in: Actes de ICRA'07 IEEE International Conference on Robotics and Automation, Rome, Italie, 2007, pp. 840–845.

[29] N. Zonta, I. J. Grimstead, N. J. Avis, A. Brancale, Accessible haptic technology for drug design applications, J. Mol. Model. 15 (2) (2009) 193–196.

[30] A. Anthopoulos, I. Grimstead, A. Brancale, Gpu-accelerated molecular mechanics computations, J. Comput. Chem. 34 (26) (2013) 2249–2260.

[31] A. Anthopoulos, G. Pasqualetto, I. Grimstead, A. Brancale, Haptic-driven, interactive drug design: implementing a gpu-based approach to evaluate the induced fit effect., Faraday Discuss. 169 (2014) 323–342.

[32] N. Schmid, A. P. Eichenberger, A. Choutko, S. Riniker, M. Winger, A. E. Mark, W. F. van Gunsteren, Definition and testing of the gromos force-field versions 54a7 and 54b7, Eur. Biophys. J. 40 (7) (2011) 843–856.

[33] D. van der Spoel, E. Lindahl, B. Hess, 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).

[34] S. J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, S. Profeta, P. Weiner, A new force field for molecular mechanical simulation of nucleic acids and proteins, J. Am. Chem. Soc 106 (3) (1984) 765–784.

[35] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, S. Swaminathan, M. Karplus, Charmm: A program for macromolecular energy, minimization, and dynamics calculations, J. Comput. Chem. 4 (2) (1983) 187–217.

[36] R. C. Rizzo, W. L. Jorgensen, Opls all-atom model for amines: resolution of the amine hydration problem, J. Am. Chem. Soc 121 (20) (1999) 4827–4836.

[37] K. Opencl, A. Munshi, The opencl specification version: 1.2 document revision: 19 (2012).
URL `https://www.khronos.org/registry/cl`

[38] NVIDIA, NVIDIA OpenCL Best Practices Guide, NVIDIA, https://developer.nvidia.com/cuda-toolkit-32-downloads (2010).

[39] T.-P. Fang, L. A. Piegl, Delaunay triangulation using a uniform grid, Computer Graphics and Applications, IEEE 13 (3) (1993) 36–47.

[40] G. Iakovou, S. Hayward, S. Laycock, A real-time proximity querying algorithm for haptic-based molecular docking, Faraday Discuss. 169 (2014) 359–377.

[41] NVIDIA, Cuda toolkit 3.1.
URL https://developer.nvidia.com/cuda-toolkit-31-downloads

[42] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. N. Shindyalov, P. E. Bourne, The protein data bank, Nucleic Acids Res. 28 (1) (2000) 235–242.

[43] A.-E. Molza, N. Férey, M. Czjzek, E. Le Rumeur, J.-F. Hubert, A. Tek, B. Laurent, M. Baaden, O. Delalande, Innovative interactive flexible docking method for multi-scale reconstruction elucidates dystrophin molecular assembly, Faraday Discuss. 169 (2014) 45–62.

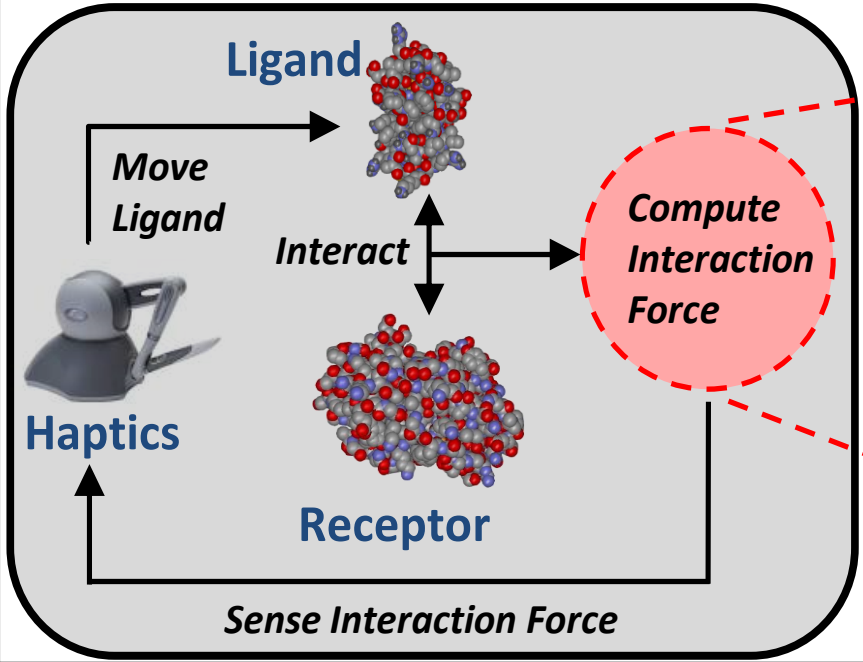
[44] C. Duriez, F. Dubois, A. Kheddar, C. Andriot, Realistic haptic rendering of interacting deformable objects in virtual environments, Visualization and Computer Graphics, IEEE Transactions on 12 (1) (2006) 36–47.

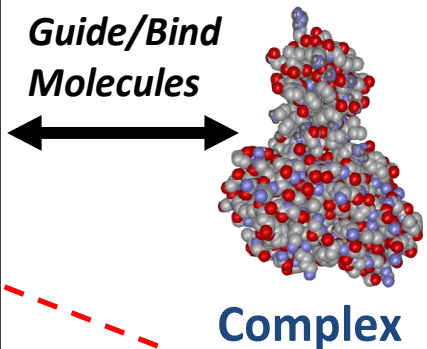
[45] A. W. Schuttelkopf, D. M. Van Aalten, Prodrg: a tool for high-throughput crystallography of protein-ligand complexes, Acta Crystallogr., Sect. D: Biol. Crystallogr. 60 (8) (2004) 1355–1363.

[46] A. Bolopion, B. Cagneau, S. Redon, S. Régnier, Variable gain haptic coupling for molecular simulation, in: World Haptics Conference (WHC), 2011 IEEE, IEEE, 2011, pp. 469–474.

[47] M. B. Stocks, S. D. Laycock, S. Hayward, Applying forces to elastic network models of large biomolecules using a haptic feedback device, J. Comput.-Aided Mol. Des. 25 (3) (2011) 203–211.

GPU-accelerated Interactive Haptics-assisted Rigid Docking





GPU

do** in **parallel

- *update ligand atom coordinates*
- *find all ligand/receptor atoms within a cut-off distance*
- *compute van der Waals and electrostatic interactions for these atom pairs*

Highlights

- A GPU data-parallel adaptive force calculation approach for haptic-based docking.
- Force updates in less than 2ms for molecules comprising more than 100,000 atoms.
- Speed improvements of up to 90 times CPU-based force calculation approaches.
- No precomputation requirements on the receptor.
- A formula-based strategy for selecting at run-time the partitioning structure.