Simulating molecular docking with haptics

Georgios Iakovou

A thesis submitted for the degree of
Doctor of Philosophy
at the University of East Anglia
December 2015

Simulating molecular docking with haptics

Georgios Iakovou

Abstract

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 the study of the binding process. In molecular docking, haptics enables the user to sense the interaction forces and intervene cognitively in the docking process. Haptics-assisted docking systems provide an immersive virtual docking environment where the user can interact with the molecules, feel the interaction forces using their sense of touch, identify visually the binding site, and guide the molecules to their binding pose. Despite a forty-year research effort however, the docking community has been slow to adopt this technology. Proprietary, unreleased software, expensive haptic hardware and limits on processing power are the main reasons for this. Another significant factor is the size of the molecules simulated, limited to small molecules.

The focus of the research described in this thesis is the development of an interactive haptics-assisted docking application that addresses the above issues, and enables the rigid docking of very large biomolecules and the study of the underlying interactions. Novel methods for computing the interaction forces of binding on the CPU and GPU, in real-time, have been developed. The force calculation methods proposed here overcome several computational limitations of previous approaches, such as precomputed force grids, and could potentially be used to model molecular flexibility at haptic refresh rates. Methods for force scaling, multipoint collision response, and haptic navigation are also reported that address newfound issues, particular to the interactive docking of large systems, e.g. force stability at molecular collision. The result is a haptics-assisted docking application, Haptimol_RD, that runs on relatively inexpensive consumer level hardware, (i.e. there is no need for specialized/proprietary hardware).

Dedicated to my father Apostolos Iakovou, the father I aspire to be to my son, to my pappou (grandfather) Eleftherios Iakovou, and to my giagiades (grandmothers) Despoina Iakovou and Barbara Ziara. They are always in my thoughts and held dear to my heart.

Acknowledgements

I would like to thank, and acknowledge the help and guidance of my primary supervisor Dr Stephen Laycock and my secondary supervisor Dr Steven Hayward over the duration of this doctorate thesis. I have taken invaluable lessons from them on how to conduct proper research and be a good scientist. I have greatly enjoyed our meetings (especially those involving coffee and cake) and everyday interaction, which helped me in many ways ride this roll-coaster of mixed emotions and immense pressure I call PhD, successfully and enjoyably.

Thank you to all members (scientists and statisticians) of the Graphics, Colour and Visualisation Laboratory for the interesting conversations we had and the good times we spent together. May the force be with you!!...(and risk analysis guide your way).

My love and many thanks to my mother Anthoula Iakovou for being the excellent mother she is. Her limitless love, support and constructive criticism have been one of the driving forces in my life. I continue by thanking my brothers Lefteris and Dimitris Iakovou, my sister-in-law Sofia Athanasiadou, my nephew Apostolos Iakovou, and my nieces Anthoula and Anatoli Iakovou. I thank my brothers for helping me improve my deductive reasoning skills and information-querying/processing response times, via a series of well orchestrated pranks. Their life's goal to keep stress testing my cognitive abilities finally paid off, i.e. I am about to get a doctoral degree! Likewise, I thank my sister-in-law for warning me (whenever possible) on upcoming pranks; without her I would have been stressed out a lot more. Finally, thanks to my nephew and nieces for all the free treats I won from them playing card games; you have sweetened my way through my PhD. I close this paragraph, by thanking my pappou Georgios Ziaras, the man I was named after, for being always there for me ready to help. Thank you all for your love, care and support; you are a big part of my life.

Last but not least, I want to thank my wife Alexia Tsigka and my son Apostolos Iakovou, for their unconditional love, understanding, patience and inspiration. They are my beacons in life, that guide my actions and make things worthwhile. Without them, my life would have been meaningless and this thesis would have not been achieved. Thanks therefore to my father-in-law Vasilis Tsigkas and mother-in-law Irene Lazani for raising this wonderful woman and bringing her to my life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

"Molecular biology is mankind's attempt to figure out how God engineered His greatest invention-life. As with all great inventions, details are top secret; however, even top secrets may become known. I find it a great privilege to live in a time where God allows us to gain some insight into His construction plans, only a short step away from giving us the power to control life processes genetically. I hope it will be to the benefit of mankind, and not to its destruction."[Neu97]

Intermolecular complex formation is a fact of life underlying many biological processes. Molecules bind with other molecules to form complex structures that control various regulatory and metabolic processes of the living cell. Proteins, the building blocks of life, are often the main participants in such intermolecular interactions, with drug molecules sometimes being their counterparts.

Proteins align and change their tertiary (3D) shape in a way that facilitates their binding with other biomolecules and chemical entities such as proteins, nucleic acids, lipids, sugars, nucleotides, ions, and water [AJL+08]. Through these bindings, proteins become involved in and regulate composite biological processes, such as cellular signalling (signal transduction), gene regulation, metabolic control, and immunity [PSVV07]. In signal transduction for example, proteins located on the cell surface (called Cell-Surface Receptors) interact with signalling molecules, and convert extracellular signals into intracellular ones [AJL$^+$08]. Failure of these processes might lead to serious diseases, such as cancer, Alzheimers, Huntingtons, or cystic fibrosis [And03, ADPH11]. Comprehensive and accurate understanding of the underlying mechanisms will enable us to enhance or inhibit such protein activities, and regulate (as needed) their respective processes.

The therapeutic effects and pharmacological properties of drugs are dictated by the bindings formed between the drug agents and the target molecules [KSL05]. Drugs are small molecules that bind on large biomolecules, such as proteins, and can stimulate, act upon, or inhibit the activity of these biomolecules. Given these target molecules, the *ex-ante* identification of effective drug agents will speed-up the drug-discovery process. New methods and techniques that are capable of matching new drug candidates quickly, accurately, and cost-effectively with given target molecules are always in demand from pharmaceutical companies. Traditional laboratory-based techniques are too costly and time-consuming, and are incapable of keeping up with the current, ever-increasing demand for new drugs (e.g. to treat HIV or cancer).

We need tools and methods capable of modelling and replicating accurately the mechanics of intermolecular bindings. Such capabilities will help us to understand the processes of life and enable us to design and produce synthetic proteins and therapeutic drugs that could cure serious diseases and improve our quality of life.

For the past 40 years, scientists have been studying intermolecular bindings. They have relied on experimental (*in vitro*) work and computational methods (*in silico*) to study, model, and replicate them. Advances in biology, biochemistry, biophysics, and bioinformatics have laid the necessary foundations to support these efforts.

Experimental techniques such as X-Ray Crystallography (XRC) and Nuclear Magnetic Resonance spectroscopy (NMR) have enabled scientists to determine the structure of biomolecules at atomic level, store these structures in databases (e.g. Protein Data Bank (PDB) [BWF$^+$00]), and make them available to the community for further study and research. Molecular visualization algorithms (see Lee and Richard [LR71], Connolly [Con83b, Con83a], and Smith and Gund [SG78]) have described the rendering of these structures in 3D space, and computer-graphics software/hardware have made such rendering possible on computer screen. Molecular physics models have given measurable and close approximations of the force fields and energy potentials present in intermolecular interactions. Computer processing power has facilitated the development of advanced, computationally intensive, algorithms that attempt to address/solve the substantial combinatorial complexities of these interactions [Rit08, YAR11]. Finally, advances in 3D-computer-graphics (both in algorithms and hardware) and human-computer-interface tools (e.g. mouse/haptic devices) have allowed the scientists to scale-up these interactions from atomic-level to human-level, and acquire the ability to sense and manipulate them at interactive rates [DR09, OY90].

Despite all these developments in the field however, scientist are far from solving this problem, and numerous research groups still researching it very actively. The main reason for this is that the process is by nature extremely complex to model computationally, and becomes even more complex as the interacting molecules become larger. Existing methods and tools fail in many cases to simulate the binding process accurately and often produce incorrect results. It is therefore imperative that we continue our search for methods and tools that will help us overcome these issues, and allow us as such to unravel and conquer the mysteries of molecular binding.

1.1 Motivations and Research Objectives

Molecular docking systems comprise the computational tools that enable scientists to explore and study the process of intermolecular complex formation. The ultimate goal of docking is to fit two molecules together in a viable configuration based on their topographic and physiochemical properties. This involves the exploration of enormous amounts of potential binding conformations prior to selecting the correct one. Interactive docking systems address these issues by allowing human perception, intuition, and knowledge to assist in the binding process. They achieve this through the use of interactive molecular visualization systems and haptic force feedback devices. Interactive molecular visualization systems enable the user to view and explore the molecular structures, and identify and select potential binding sites and conformations. Haptic force feedback devices, on the other hand, allow the user to interact with the molecules, feel the interaction forces with the sense of touch, and navigate and orient the molecules during a docking simulation. Together, they offer an immersive virtual learning environment for the study of the docking process, and a test bed for exploring new ideas and hypotheses pertinent to intermolecular binding [far14].

The field of haptics-assisted docking has been researched actively for the last forty years, and numerous interactive docking approaches have been produced as a result. However, the adoption rate of this technology by the community has been thus far slow. One of the reasons is intrinsic to haptics and relates to the calculation, and haptic-rendering of the interaction forces. A fundamental part in a haptics-assisted docking application is the calculation of the interaction forces in a continuous, smooth and stable fashion. Haptic technology imposes very demanding refresh rates on the application due to the sensitivity of the human haptic sense. Existing applications address this strict time constraint by employing various model approximations (e.g. treat both molecules as rigid structures) and computational-cost-cutting techniques (e.g. precomputed force grids) to accelerate force computations on modern computing hardware. However, none of these approaches was able to facilitate the interactive docking of large molecules (comprising several thousands atoms each), which limits their scope and usefulness for the molecular docking community. Other reasons relate to the fact that only a few of the existing applications are freely available to the community, since the majority of these applications are either proprietary/unreleased, and/or utilize expensive/proprietary haptic devices.

The main objective of the work presented in this thesis is to develop algorithms and techniques (for CPU (Central Processing Unit) and GPU (Graphics Processing Unit)) that can facilitate the interactive, haptics-assisted, rigid-docking of very large molecules (i.e. comprising of hundreds of thousands atoms each). The motivation is the development of a freeware that is easy to use, runs on relatively inexpensive consumer level hardware (i.e. does not require specialized/proprietary hardware), and accommodates the docking of such large structures. The goal is to provide a tool that will enable the community to study the docking interactions of large proteins, and not only of proteins/drugs. Here, both molecules are treated as rigid structures (a common model simplification used in the field) in order to reduce the complexity of the problem. However, we know that during docking the molecules, in many cases, are flexible, and deform their structures. As such, a secondary objective is to design the aforementioned algorithms/methods in a way that will, in principle, be able to accommodate the docking of flexible structures, with the hope that this will steer the research effort towards this direction, i.e. flexible docking.

1.2 Contributions

The main contributions to the field include the following three novel pieces of work. The first one is a real-time force calculation approach, that can compute (on the CPU) the interaction forces, at haptic refresh rates, for molecules comprising up to seven thousand atoms each [IHL14]. The approach addresses efficiently and successfully all issues related to existing CPU-based force calculation methods, and can be applied equally to large protein-protein, and protein-drug docking problems. The second piece of work extends the CPU-based approach to the GPU [IHL15]. The GPU-based method utilizes the many-core processing capabilities of modern GPUs in order to accelerate force computations. The method improves substantially upon the CPU-based approach, and as such can accommodate the haptics-assisted docking of very large molecules, i.e. comprising hundreds of thousands of atoms each. The final piece of work involves the development of a freeware that can be used for the interactive docking of large proteins. The application implements both force calculation approaches, and it can thus accommodate the docking of large molecules either on the CPU or the GPU. Given the size of the molecules supported by the application, several issues related to haptic stability and force rendering had to be addressed anew. The results of that work led to the following three additional contributions, a) a force scaling method that allows the user to study/experience a specific range of intermolecular forces during the simulation, b) a multipoint (distributed) collision response technique capable of providing stable forces at molecular collision and of prohibiting extensive atom overlapping, and c) a haptic navigation technique that can facilitate docking simulations of large proteins.

1.3 Publications

The following peer-reviewed journal publications were resulted from the work discussed in Chapters 3, 4, 5 and 6:

1. Georgios Iakovou, Steven J Hayward, and Stephen D Laycock. Fd169: A real-time proximity querying algorithm for haptic-based molecular docking. Faraday Discussions, 2014.

2. Georgios Iakovou, Steven Hayward, and Stephen D Laycock. Adaptive GPU-accelerated force calculation for interactive rigid molecular docking using haptics. Journal of Molecular Graphics and Modelling, 61:1-12, 2015.

3. Georgios Iakovou, Stephen Laycock and Steven Hayward. Determination of locked interfaces in biomolecular complexes using Haptimol_RD. Biophysics and Physicobiology, 2016.

1.4 Thesis Outline

Chapter 2 starts by defining the problem of molecular docking and its mechanics. It then outlines the functional characteristics of the two main categories of docking applications (i.e. automated and interactive), and moves on to focus on interactive haptics-assisted docking. After describing the main issues related to haptics-assisted docking, the chapter concludes by reviewing the advancements made in the field.

Chapter 3 describes the development of a haptics-assisted docking application called Haptimol_RD. The chapter discusses the design and development of the application's molecular visualization and haptic navigation routines, and identifies the force field and molecular-structure-defining format supported by Haptimol_RD. A novel method for the haptic navigation of large data sets is also proposed here.

Chapter 4 discuses novel methods and implementation details for the real-time computation, on the CPU, of the electrostatic and VDW force contributions in molecular docking. The performance of these methods was tested using molecules of different size, and the chapter reports these results.

Chapter 5 presents a scalable, GPU-parallelizable version of the force calculation approach discussed in Chapter 4. The chapter provides a quick overview of the GPU computing environment, and details the implementation of this real-time GPU-based force calculation approach. Performance results pertinent to this approach are also provided here.

Chapter 6 describes the issues of force scaling and stability, and proposes novel force scaling and multi-point collision response methods that address these issues. An implementation overview for Haptimol_RD is also given here.

Chapter 7 outlines the main conclusions drawn by this thesis, and states several ideas for future research in the field.

Chapter 2

Molecular docking and Haptics

2.1 Molecular Docking

Molecular docking refers to the computational methods devised and employed by researchers and field practitioners in order to simulate (as accurately as possible) the natural process of intermolecular complex formation. In its general form the problem can be stated as:

> *Given two molecules and their tertiary structures employ computational methods and computer power to search and find the appropriate structural conformations, orientations and alignments (in 3D space) that will enable the binding of these molecules into a larger more complex one.*

In simpler terms, molecular docking tries to fit two molecules together based on their topographic and physiochemical properties. One of these molecules is called the *receptor* and the other one is called the *ligand* (see Figure 2.1).

The receptor is usually a large biomolecule (e.g. a protein), whereas the ligand is a smaller molecule (a molecule that consists of a couple of tens of atoms, e.g. a drug molecule), but it can equally be another large biomolecule such as protein, nucleic acid, lipid or nucleotide. During docking, the receptor and the ligand molecules transform (i.e. translate, rotate, and deform) their tertiary structure, and acquire conformations that favour binding. In these conformations, both molecules exhibit strong geometric and chemical complementarity and the free energy (read Section 2.1.1) of the system is at its global minimum. The fast, accurate, and *in silico* emulation/prediction of this process is the ultimate goal of molecular docking [MEL+08].

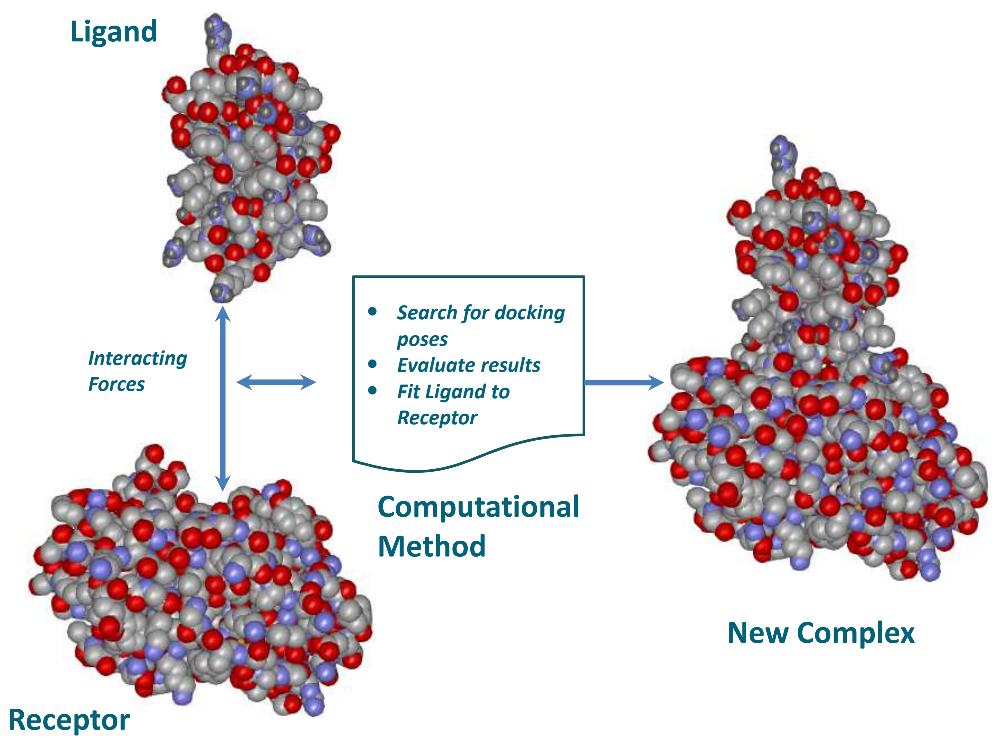


Figure 2.1: The process of molecular docking. The ligand and receptor molecules bind together into a larger molecular complex. At the docking site the two structures are complementary to each other.

Many research groups have been studying this problem, producing a plethora of docking algorithms and software applications ([MEL+08] provides an extensive list). In the majority of these studies, the receptor is a protein and the ligand is either another protein or a small chemical substance. Researchers usually term these two subcases of the docking problem as *protein-protein* docking [EKK04, SS02] and *protein-ligand* docking [BK03, YAR11] respectively. Lastly, there are few studies addressing the problem of *protein-DNA* docking (i.e. ligand is a nucleic acid) as well [KAR$^+$94, SGJ98].

Protein-protein docking methods are used mainly for modelling protein-complex formations. Their task is the prediction of the tertiary structure of the new protein complex given the two constituent proteins [Rit08]. Many research fields (e.g. structural/molecular biology, proteomics) have been using them to gain structural information about proteins. The field of computational proteomics [MR01], for example, employs such methods to generate and analyse protein structures at large scales. Such knowledge enables scientists to predict the chemical and biological properties [Neu97] of the given protein-complex, and define the complex's functionality and behaviour. Experimental techniques such as XRC, and NMR are capable of providing such structural information, but are costly, time-consuming and not universally applicable. Namely, there are known cases of protein-complexes that are short-lived and cannot be studied by such techniques [EKK04]. In those cases, protein-protein docking methods represent the only alternative. According to some authors, accurate prediction of protein structure will eventually revolutionize human medicine in understanding, diagnosing, preventing and treating human diseases [MR01].

Protein-ligand docking methods are often used for modelling drug behaviour. The therapeutic action and side effects of a pharmaceutical agent depend closely on where and how the agent binds to a given receptor. The pharmaceutical industry has been using extensively such docking methods for computer-aided drug design (CADD) and virtual screening (VS) [AT01, BH05, KDFB04, VEM09]. In CADD, drug engineers design and test the binding propensity of synthetic drug molecules against specific target receptors. In VS, specialized algorithms traverse large libraries of compounds in search of candidate drug substances that bind to given target molecules (i.e. reduce the search space). Although the resulting candidates will eventually be subjected to experimental testing, it is extremely beneficial for the industry to identify the right candidates at the early stages of the drug design process (commonly referred to as *lead generation*) without the need to resort to costly and time-consuming experimental techniques such as synthesis, co-crystallisation and assay. Thus, docking methods enable the pharmaceutical companies to expedite and optimize the drug discovery process in a quick, efficient, cost-effective, affordable and guided manner.

Overall, molecular-docking is a problem that has been researched extensively by the scientific community and active research in the field spans across various disciplines such as molecular biology, molecular physics, structural biology, biochemistry, bioinformatics, computer graphics/simulation, medicinal chemistry and pharmacology [AGO08, MR01, Rit08].

2.1.1 Docking Mechanics

During docking, the ligand attaches itself on the *binding/active site* of the receptor. This is a region in the receptor's tertiary structure that favours binding with other molecules, and it could differ from one ligand to another. At the active site, the two molecules exhibit optimal structural and chemical complementarity. In order to achieve that, both perform a series of rotations, transformations, and (in many cases) deformations on their conformation (a process referred to as *pose selection* [ADPH11]). There are three prevailing theories/models that underpin the structural behaviour of the two molecules during pose selection. These theories/models are the following [CPN10] (see Figure 2.2):

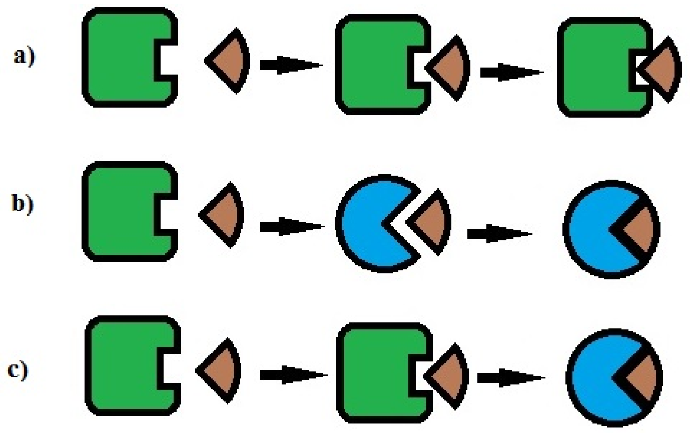


Figure 2.2: The binding kinetics of the receptor (green or blue) and the ligand (brown) molecules, under the three structural behaviour theories (note: in this example only the receptor can deform structurally). a) Under *rigid-body* the receptor does not deform and the docking fit is not the best possible; b) Under *selected fit* the receptor deforms its structure in the second step (before the binding occurs), and assumes the most favourable (for the ligand) docking conformation. The docking fit is perfect; c) Under *induced fit* the receptor deforms its structure in the third step (while the binding occurs), in order to maximize the fit between the two surfaces. Again, the docking fit is perfect (adapted from Weikl and von Deuster [WvD09])

- *Rigid-body (lock-key fit)* suggests that the ligand (key) binds to the receptor (lock) like a key fits to its lock. It models receptor and ligand molecules as rigid bodies.

- *Conformational Selection (selected fit)* postulates that out of a given set of receptor conformations the ligand selects the most favourable one, and shifts the receptor toward this conformation, prior to binding. It usually models the ligand as a rigid-body and the receptor as a deformable one.

- *Induced fit (hand-in-glove fit)* suggests that the receptor and the ligand molecules deform their tertiary structure during binding to obtain the best docking fit (like a hand fits in a glove). Both molecules are modelled as fully flexible.

The first model allows the two molecules to have only translational and rotational degrees of freedom (DOF) during pose selection. The other two augment the translational and rotational DOF with internal DOF, to account for ligand and/or receptor flexibility. In the first case, molecular docking becomes a problem of 6DOF, whereas in the latter two, a problem of thousands of DOF [TPJK01]. Although most of the studies have utilized the rigid body model (due to its simplicity), it is widely recognized that the other two simulate the pose selection process more accurately.

The structural behaviour of the two molecules is a crucial aspect of the docking process, since it underlies the strength of the binding forces, and the type of interactions involved. Under any model, the strength of the binding is controlled by a group of non-covalent (non-bonded) interactions formed between the two molecules such as steric, electrostatic (i.e. Coulomb forces), hydrogen bonds, Van der Waals (VDW) and hydrophobic ones [MWS96]. The characteristics of these interactions are outlined as follows:

- *Steric interaction* describes the space-filling effect present when two complementary shapes attach to one another. Shape complementarity of some degree is a necessary condition for this type of interaction.

- *Electrostatic interaction* explains the force developed between electronegative and electropositive charged atoms. It is an attractive force when the two atoms are oppositely charged and a repulsive force when both atoms have the same charge.

- *Hydrogen bond* defines the electromagnetic attractive force between an electropositive hydrogen atom partially shared by other electronegative atoms such as oxygen, nitrogen, or fluorine. It is a special form of polar interaction in which an electropositive hydrogen atom is partially attached to two electronegative atoms. Researchers usually model such bonds as electrostatic interactions, but unlike the electrostatic ones, these bonds are highly directional.

- *Van der Waals interaction* explains the short-range repulsion and long-range attraction forces present in oppositely polarized flickering dipoles. Namely, the electron cloud fluctuations of nonpolar atoms produce flickering dipoles, which induce momentarily oppositely polarized flickering dipoles in the nearby atoms, and thus generate an attractive force. When these atoms come in close proximity their clouds repel themselves in order to prevent electron overlaps, according to the Pauli Exclusion Principle.

- *Hydrophobic interaction* is present in aqueous solutions when nonpolar molecules push away the surrounding water molecules, and thus drive their nonpolar surfaces in close proximity. The hydrophobic forces are important for the proper folding of protein molecules.

Individually, these non-bonded interactions are quite weak, but when grouped together they can produce an attractive force between the two molecules that is strong enough to keep them attached into a stable complex.

When molecular flexibility is taken into account, bonded interactions also contribute to the bindings total energy potential and force field. When molecules deform their tertiary structure to accommodate binding, they eventually change the position of their bonded atoms, which leads to bond stretching, bending, and twisting [Sub06].

Molecular dynamics (MD) refer to these bonded-interactions as bond stretching, bond angle, and torsion angle potentials, respectively [OY90].

The summation of the bonded and non-bonded force potentials characterizes the *binding affinity* of the intermolecular interaction; whereas, the summation of the respective energy potentials and entropy contribution constitutes the system's *free energy*. A docking process is successful when the free energy of the molecular ensemble reaches the global minimum and the binding affinity is high.

2.1.2 Outlining Docking Approaches

The development of a docking solution is an interesting yet difficult endeavour. It necessitates the discovery and utilization of efficient methods and strategies, capable of exploring enormous amounts of potential binding conformations prior to selecting the correct (experimentally verifiable) docking ensemble.

In general, a successful docking solution consists of the following two complementary, functional components: a search algorithm and a scoring method. The main objective of a search algorithm is to explore the receptor/ligand conformational space, and evaluate potential binding poses for chemical and structural complementarity. Likewise, the main objective of a scoring method is to evaluate the binding poses (identified during searching), and select those poses that replicate closely the ones determined experimentally. The success of the docking solution (i.e. how closely the binding pose resembles the native one) depends closely on the searching efficiency and scoring accuracy of these components.

The searching of the conformational space is, in its own right, a difficult problem to tackle. Even in its simplest form (i.e. inflexible molecules), the number of possible docking conformations is vast, which renders an exhaustive search intractably difficult to perform, and this number grows exponentially when molecular flexibility is taken into account. Searching algorithms can be grouped into the following two broad categories: automated and interactive. Automated algorithms utilize sophisticated, pose selection/matching methods and rely only on computer power to carry them through. Conversely, interactive algorithms require human intervention, and their performance depends closely on the underlying human intuition, knowledge and expertise. Given the searching algorithm employed, docking solutions could also be referred to as *automated/algorithmic*, or *interactive*.

The *algorithmic* approaches to docking employ a range of efficient searching algorithms (e.g simulated annealing, geometric hashing, Monte Carlo (MC)) to sample the conformational space, and then use energy-based scoring methods to rank these samples and select those that produce the best fit [MEL$^+$08]. Earlier docking solutions depended mainly on searching algorithms that performed geometry matching, (e.g. geometric hashing, shape descriptors, fast Fourier transformations (FFT)), and accounted only for shape complementarity [LR96, SKB92]. These algorithms treated both molecules as rigid bodies (*rigid-body* docking), and evaluated steric or hydrogen bond interactions for pose scoring [KBO$^+$82, MWS96]. As the computers evolved and became more powerful, researchers developed and proposed equally advanced and sophisticated searching algorithms to address ligand flexibility (*flexible-ligand* docking). These algorithms depended either on stochastic methods (e.g. MC, genetic algorithms (GA), simulated annealing), or on incremental construction methods to sample the pose space for chemical complementarily, and evaluated both bonded and non-bonded interactions for pose scoring [Rit08, WCM97, YAR11]. Recent algorithms addressed receptor flexibility (*flexible receptor-ligand* docking) in a similar manner, by employing methods such as MD simulation, MC, or conformer libraries [BK03]. Despite the significant progress achieved in algorithmic docking, it still remains significantly challenging to handle molecular flexibility, and replicate accurately the respective docking process. Automated docking solutions, in general, predict the correct pose only about 70% of the time [LSP06].

The *interactive* approaches, on the other hand, address the same issues as the algorithmic ones, but unlike them, they depend on rational human thought and human intuition to execute the pose sampling process, rather than raw computational power. Early, advances in molecular visualization and 3D computer graphics paved the way for such interactive approaches. Molecular visualization applications such as RasMol [SMW95], Chimera [PGH$^+$04], Protein Explorer [Mar02], PyMol [Del02], VMD [HDS96] and MidasPlus [FHJL88] enabled scientists to view the tertiary structure of molecules in various styles [Con83b, Con83a, LR71, SG78], identify (visually) possible binding sites and docking conformations, and run molecular-dynamics simulations in real-time. Similar advances in computer-interface devices (e.g. mouse, haptics) enabled the scientists to interact with the virtual world, and facilitated the multimodal exploration of the docking process.

Interactive docking, and specifically haptics-assisted docking, is the focus of this dissertation. As such, the next section provides an analytic view of the issues, rendering models, and applications pertaining to haptics-assisted docking.

2.2 Haptics-assisted Docking

The automated/algorithmic approaches to molecular docking represent the majority of the applications and studies available in the field. Despite, however, their popularity in the research community, there are many unresolved issues. For instance, automated docking methods employ computational intensive searching, posing, and scoring algorithms that are time consuming (solutions to docking problems can take several hours) [OY90, SDINW05] and do not always predict the correct docking conformations [DMR07a, LSP06, SS02]. The effect of the solvent on the intermolecular interactions and protein flexibility are other issues not addressed/modelled properly in these methods [CWL12]. Finally, the automated docking methods deprive their solution sets of another important factor, human knowledge and intuition.

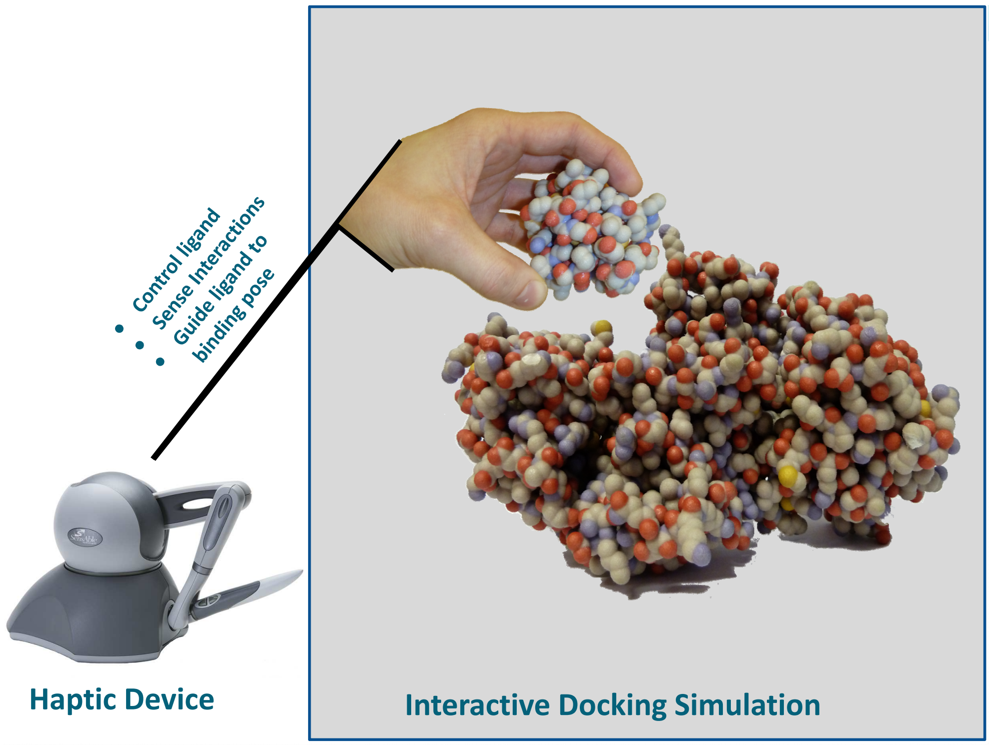

• Control ligand
• Sense Interactions
• Guide ligand to binding pose
Haptic Device
Interactive Docking Simulation

Figure 2.3: A schematic representation of haptics-assisted docking. The user controls the ligand with the haptic device, senses the interaction forces, and uses this visuohaptic information in order to identify and guide the ligand to the docking pose.

Haptic devices and interactive molecular visualization systems transfer the complexity of the molecular binding process from computers to humans. Haptics-assisted docking systems simulate the docking process in a 3D virtual environment, where the user can interact with the molecules, and perform a knowledge-guided search and selection of the final docking ensemble (Figure 2.3). In the late 80s, Ouh-Young [OY90] demonstrated that this knowledge-guided docking approach produces faster and more accurate docking results than an automated approach. Since then, several haptics-assisted docking systems have been proposed and developed [BSA01, DMR07a, DR09, NMT02, STW09]. It is expected that haptic devices are here to stay, and future advancements in the field will facilitate new exciting, intellectual, and commercial explorations.

The next four sections provide an overview of haptic technology and how it is applied to molecular docking problems (Section 2.2.1), describe the energy and force calculation models utilized by haptics-assisted docking solutions (Section 2.2.2), present ways to render these forces haptically (Section 2.2.3), and reviews the advances in the field (Section 2.2.4). The discussion in these sections focuses on CPU-based interactive docking, since it comprises the majority of the existing haptics-assisted docking systems.

2.2.1 Overview

The term *haptics* comes from the Greek word "Haptesthai" which means "to touch". Haptic technology enables the user to interact with a virtual environment through the sense of touch [BS02]. Tactile (e.g., object texture) and kinesthetic (e.g., force sensation) information from the virtual world are transmitted back to the user via the device, allowing one to feel the physical properties of virtual objects. A typical haptic device can render either translational forces along the three axes x,y, and z (3DOF device), or translational and rotational (torque) forces along the same axes (6DOF device). Haptics have many applications in industry and research, especially in areas such as medicine (e.g. surgical simulation, rehabilitation of patients with neurological disorders), entertainment (3D painting, morphing and sculpting), mechanical design (path planning and assembly sequencing), and scientific visualization (geophysical data analysis, molecular manipulation) [BS02, BJ08, JBT04, MPT05, OL05b, SCB04].

In molecular docking, the haptic, force-feedback technology is coupled with 3D-visualization systems to help molecular scientists achieve better and faster docking results. In such systems, the tertiary structures of ligand and receptor molecules comprise the virtual world, and the haptic device interacts with this world through a virtual control point called the Haptic Interface Point (HIP). The user utilizes the visual cues and their structural and biochemical expertise to identify potential active sites and select conformations that appear complementary, both geometrically and chemically. With the ligand often attached on the HIP, one can use the device to move and rotate the ligand around the receptor and assume various binding poses. At the same time, one can measure the respective total energy potential, feel the repulsive or attractive forces of the underlying interactions, and cognitively select the docking pose and site that produces the best fit. This *visuohaptic* representation of the molecular world (from atomic to world scale) enables the user to feel the intensities of the interaction forces and gain a better awareness of the intermolecular binding process.

2.2.2 Calculation Models of Intermolecular Interactions

Molecular mechanics dictate that the total energy and force of the binding depend on the bonded and non-bonded interactions (see Section 2.1.1) developed between the two molecules. When the molecules are treated as rigid, the non-bonded interactions are the only ones required, but when molecules are considered to be flexible bonded interactions (i.e. the stretching, bending and twisting of covalent bonds) must be included either explicitly or implicitly. However, the modelling/simulation of bonded interactions is a very computationally demanding task, and thus most haptics-assisted docking solutions neither model bonded interactions, nor account for molecular flexibility (i.e. treat molecules as rigid bodies). Typically, they model only the VDW interactions [BSA01, HS10, LYL06b], or both the VDW and the electrostatic (Coulombic) interactions [FNM+09, LL04, OY90, SB06, WMJ07].

VDW interactions are often approximated by the Lennard-Jones 6-12 (LJ), energy potential function. This function computes the energy potential between two atoms i and j as,

$$E_{ij}^{VDW} = \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \tag{2.2.1}$$

where A_{ij} and B_{ij} are constants that depend on the type of interacting atoms, and r_{ij} is the distance between these atoms (measured from their centre). In Equation 2.2.1, the first term defines the repulsive part of the force, whereas the second term defines the attractive part - the dispersion force. A_{ij} and B_{ij} can also be written as $A_{ij} = 4\varepsilon\sigma^{12}$ and $B_{ij} = 4\varepsilon\sigma^{6}$ respecively, where ε is the *potential well* depth (i.e. the minimum of potential energy), and σ is the distance at which the inter-atomic potential is zero (see Figure 2.5).

The respective force function is derived from the distance-based differential (gradient) of the LJ potential, as,

$$\begin{aligned} \vec{F}_{ij}^{VDW} &= -\nabla E_{ij}^{VDW} \Rightarrow \\ \vec{F}_{ij}^{VDW} &= \left[12\frac{A_{ij}}{r_{ij}^{13}} - 6\frac{B_{ij}}{r_{ij}^{7}} \right] \hat{\vec{r}}_{ij} \end{aligned} \tag{2.2.2}$$

where A_{ij}, B_{ij}, and r_{ij} have the same meaning as in Equation 2.2.1, and $\vec{r}_{ij}$ is the unit vector in the direction from atom i to j.

The electrostatic interactions are calculated using Coulomb's law. Given two atoms i and j, the electrostatic energy potential between the two is

$$E_{ij}^{ES} = \frac{q_i q_j}{4\pi\epsilon\epsilon_0 r_{ij}} \tag{2.2.3}$$

where 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, and r_{ij} is the distance between these atoms. Analogous to the derivation of the LJ force function, the electrostatic force equation is given by

$$\begin{aligned} \vec{F}_{ij}^{ES} &= -\nabla E_{ij}^{ES} \Rightarrow \\ \vec{F}_{ij}^{ES} &= \frac{q_i q_j}{4\pi\epsilon\epsilon_0 r_{ij}^2}\vec{r}_{ij} \end{aligned} \tag{2.2.4}$$

where q_i, q_j, ϵ_0, ϵ, and r_{ij} have the same meaning as in Equation 2.2.3, and $\vec{r}_{ij}$ is again the unit vector in the direction from atom i to j.

The values of parameters A_{ij}, B_{ij}, q_i, and q_j in Equations 2.2.1, 2.2.2, 2.2.3, and 2.2.4 vary according to the actual empirical force-field model utilized for the respective calculations. There are various force-field models, such as GROMOS [SEC+11], AMBER [WKC+84], CHARMM [BBO+83], MM3 [LA91], MM4 [ACL96], MMFF94 [HN96], and OPLS-aa [DFTRJ97, JMTR96, RJ99]. These models derived their underlying values based on experimental observations, and their values differ because they were designed to accommodate different types of molecular structures (e.g. AMBER for proteins and DNA, CHARMM for small molecules and macromolecules, etc). In addition to molecular docking, these models are also used extensively in molecular-dynamics (MD) simulations.

The majority of haptics-assisted solutions use these equations to compute the total energy potential and force of the binding, and then render this force on the haptic device. Depending on the type of the non-bonded interactions modelled (VDW only, or VDW and electrostatics), the total energy/force of the docking simulation can by derived either by Equations 2.2.5 and 2.2.6 (for VDW only), or Equations 2.2.7 and 2.2.8 (for VDW and electrostatics).

$$E_{Tot}^{VDW} = \sum_{i}^{N}\sum_{j}^{M}\left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}\right) \tag{2.2.5}$$

$$\vec{F}_{Tot}^{VDW} = \sum_{i}^{N}\sum_{j}^{M}\left(\left[12\frac{A_{ij}}{r_{ij}^{13}} - 6\frac{B_{ij}}{r_{ij}^{7}}\right]\hat{\vec{r}}_{ij}\right) \tag{2.2.6}$$

$$E_{Tot}^{VDW+ES} = \sum_{i}^{N}\sum_{j}^{M}\left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{4\pi\epsilon\epsilon_0 r_{ij}}\right) \tag{2.2.7}$$

$$\vec{F}_{Tot}^{VDW+ES} = \sum_{i}^{N}\sum_{j}^{M}\left(\left[12\frac{A_{ij}}{r_{ij}^{13}} - 6\frac{B_{ij}}{r_{ij}^{7}} + \frac{q_i q_j}{4\pi\epsilon\epsilon_0 r_{ij}^{2}}\right]\hat{\vec{r}}_{ij}\right) \tag{2.2.8}$$

Equations 2.2.5 and 2.2.6 comprise an extremely simplified model of the binding interactions (they model shape complementarity only). They contain fewer operands than Equations 2.2.7 and 2.2.8, which makes them faster to compute, but they do not account for the electrostatic interactions, which are responsible for attracting/steering the ligand to nearby binding sites (see Nagata et. al. [NMT02]). Equations 2.2.7 and 2.2.8 take into account electrostatic interactions, but with an extra computational cost.

Both of these equation pairs, however, fail to model the hydrophobic factors of the non-bonded interactions. Though these factors are important to the stability and strength of the binding, there have been no significant works that model them. Moreover, studies such as [OY90], and [SB06] suggest that, when addressing rigid-body docking problems, Equations 2.2.7 and 2.2.8 provide a good approximation of the total energy and force involved. This dissertation utilizes Equations 2.2.7 and 2.2.8 for the same reasons.

Finally, there are some isolated studies that model rigid-docking interactions as follows:

- electrostatic only (Nagata et. al. [NMT02])
- as a force transfer function adapted from the field of volume rendering haptics (Maciejewski et. al. [MCET05])
- as the gradient of a cross-correlation function adapted from the field of signal processing (Birmanns and Wriggers [BW03])

The lack of additional studies on these modelling approaches suggests that they have not gained the same acceptance as the approaches using Equations 2.2.5 and 2.2.6, or 2.2.7 and 2.2.8. In addition to rigid-docking models, a couple of studies tried to model molecular flexibility and to account for both bonded and non-bonded interactions (Daunay et. al. [DMR07a] and Zonta et. al. [ZGAB09]). However, these approaches were either unable to attain the necessary haptic refresh rates (see Section 2.2.3), or limited their scope to very small ligands.

Developing correct models of the intermolecular interactions is crucial to our understanding of the binding mechanics. Science and experimental techniques have produced good models that account for both bonded and non-bonded terms of these interactions. Haptics-assisted docking solutions however, address mostly rigid-body problems, and thus tend to model only the non-bonded interactions. Although rigid-body docking is a popular problem within the community, mainly because of its computational simplicity, it does not reflect accurately the process of intermolecular complex formation. As computers and computational methods evolve, it seems logical that flexible docking problems will attract more attention. When that happens, tertiary-structure deformations and bonded-interaction calculations/renderings should become integral parts of any haptics-assisted docking solution.

2.2.3 Haptically Rendering Intermolecular Interaction Forces

The main goal of a haptics-assisted docking solution is to allow human perception, intuition, and knowledge to assist and accelerate the docking process. To achieve this, any solution must model sufficiently the total energy potential and force of the binding conformations, and render them *visuohaptically* at interactive rates. The users perception of the intermolecular interactions depends closely on the quality and frame rate of these renderings, and any inadequacy on this part will become detectable by the user, and invalidate the usefulness of the docking solution.

Modern haptic technology, allows the user to sense in real time the intermolecular interactions of docking (feedback cues for the total energy potential must be given visually). For a perceptually accurate haptic exploration and manipulation of the docking process, the force-feedback cues have to be updated at a rate of 1 kHz; There are reports suggesting that this requirement can be relaxed down to 500Hz [MFC$^+$14, DDKA06, OL05a], but even then, it remains a challenging constraint to satisfy, since it requires the calculation of the intermolecular interactions within 2ms (milliseconds). When this rate is not met device vibrations and force discontinuities (sensing force gaps) can occur limiting practical use. This requirement can affect not only the size of a docking simulation (i.e. size of interacting molecules), but also the type of interactions (i.e. bonded, non-bonded) modelled in it [WMJ07].

In the case of rigid-body docking, where the docking application models only the VDW and electrostatic interactions (see Equation pair 2.2.7 and 2.2.8), the real-time calculation of the total energy and force is considered infeasible for large molecular structures [WMJ07]. Based on their formulation, both equations demand a great number of pairwise, interatomic calculations between the two molecules. Hence, their time complexity is $O(\mathrm{NM})$ (where N and M are the number of receptor and ligand atoms respectively), which scales linearly with the size of the ligand. As such, when the receptor and the ligand are large molecules, both equations become prohibitively expensive to compute, and thus inappropriate for haptically-driven docking applications [Biv10, SB06].

An initial solution to these real-time-related, computational issues was given by Pattabiraman in 1985 [PLFL85]. Pattabiraman developed a 3D-grid-based approach that pre-computes and stores the total energy potential at predefined 3D-grid cells. His method divides the entire space into equally sized grid cells (see Figure 2.4), and treats the receptors tertiary structure as fixed within that space. Moreover, it assumes that each cell is occupied by a one-atom ligand; based on that assumption, it computes (off-line), and stores (in each cell) the energy potential of the respective receptor/one-atom interactions. During a simulation and as the ligand moves within that grid, the method uses the values stored in this grid to compute the total energy at interactive rates. Specifically, the approach maps all ligand atoms to the closest grid cells, and then adds up the stored energy values, along with the non-bonded parameters of the respective ligand atoms, to produce the total energy at a precision relative to the size of the grid cells. With this approach, Pattabiraman reduced the computation complexity from $O(\mathrm{NM})$ to $O(\mathrm{M})$. Although, there is a memory cost involved in storing the grid data, that cost is outweighed by the underlying performance improvement.

Ouh-Young [OY90], who pioneered the field of haptics-assisted docking, adopted

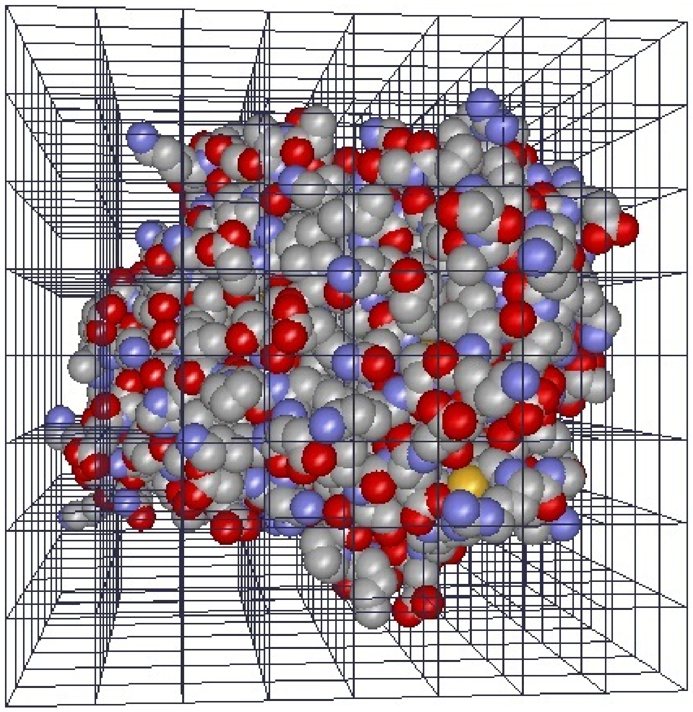


Figure 2.4: Energy/force grid surrounding the receptor molecule 1ADG {Oxidoreductase (Nad(A)-Choh(D))}

Pattabiramans method and used it, not only for energy but for force calculations as well. In addition to the energy-potential grid, Ouh-Young constructed (in a similar way) a force grid of the interatomic forces present at each grid cell. These force quantities were stored in the grid as discrete vectors. To compute the total interaction force, he had to perform a tri-linear interpolation on the appropriate force vectors, and a vector addition on those interpolation results. The total force vector was then haptically displayed back to the user. With these interpolations Ouh-Young smoothed the force bumps rendered on the device (attributed to the discrete forces stored in the grid cells), and improved the overall force feedback sensation of the docking simulation. A similar approach was also taken by Bayazit et. al. [BSA01] in order to compute the VDW interaction in their receptor-ligand docking system called

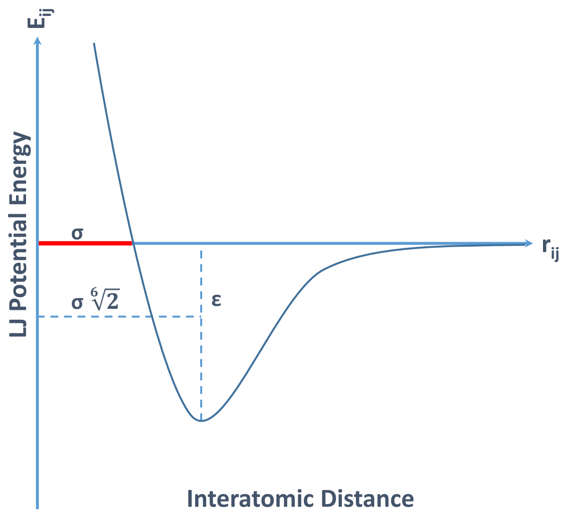
OBPRM.

Although fast, both grid methods provide a coarse approximation of the respective energy and force quantities, mainly because of the assumptions made while pre-computing the VDW grid. Namely, during the offline grid calculation phase the ligand atoms were treated as if they were "ghost" atoms that did not affect the value of parameters A_{ij} and B_{ij} in the Lennard-Jones 6-12 formula (see Section 2.2.2). Therefore, all pre-computed quantities stored in the grid cells, accounted for the VDW interactions between the receptor atoms and a "ghost" atom, rather than between the receptor atoms and the actual ligand ones. Lee and Lyons [LL04] studied this problem and proposed improvements to the pre-computed, force grid method. They suggested the use of separate force grids for each component of the non-bonded interactions (i.e. VDW and electrostatic). Moreover, instead of computing the VDW interaction between receptor atoms and a "ghost" atom, they pre-computed the VDW interactions between receptor atoms and all the different atom types comprising the actual ligand. For example, if the ligand was a water molecule they would have pre-computed the following three force grids:

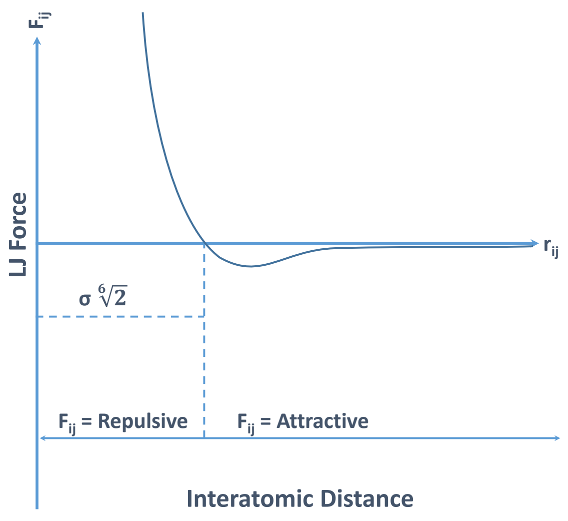
1. a 3D grid storing force vectors pertinent to the electrostatic interactions between receptor and ligand.
2. a 3D grid storing a set of the following two force vectors,
   (a) a force vector pertinent to the VDW interactions between receptor atoms and a hydrogen atom.
   (b) a force vector pertinent to the VDW interactions between receptor atoms and an oxygen atom.

This approach allowed them to not only compute the VDW interactions correctly, but to treat in real-time, and handle independently (e.g. scale, turn on and off), the energy potentials and forces attributed to the VDW and electrostatic interactions. The cost of these improvements was acceptable and required more memory space (for storing the pre-computed grid-values), and more memory lookups (for performing the actual energy/force calculations). This approach has been adopted by the majority of the solutions related to haptics-assisted docking [Biv10, HS10, LYL05, MCET05, SWS^{+}03, STW09, SB06], since it provides a fast, and flexible way to compute and haptically render the underlying docking interactions. In general however, pre-computed force grids, suffer from the following main issues: a) they have high memory requirements, b) they induce rough force transitions at cell boundaries [WMJ07], c) they are impractical for large protein-protein docking problems[RAM^{+}12], and d) they cannot accommodate (by design) receptor flexibility since the grids must be computed at haptic refresh rates after each structural deformation.

Calculating the interaction forces at haptically-acceptable frame rates is a necessary condition for any haptics-assisted docking system, but it is not the only one. The second important condition is the continuous and detectable representation of the scales and intensities of these forces [Biv10]. In his thesis, Bivall asserted that a user should be able to sense the weak (attractive and repulsive), and very strong, forces of intermolecular interactions in a clear, smooth, and distinguishable way. Although it is fairly straightforward to simulate and render electrostatic forces on a haptic device, it is not easy for the VDW forces (modelled with the Lennard-Jones formula) because they are very sensitive to distance changes and can change rapidly in magnitude and direction between successive haptic cycles. As depicted in Figure 2.5, the Lennard-Jones potential is zero when the interatomic distance r_{ij} is σ, and



(a)



(b)

Figure 2.5: Juxtaposing the Lennard-Jones energy and force graphs. a) Depiction of a Lennard-Jones energy graph where σ (red line) is the distance at which the interatomic potential is zero, $\sigma\sqrt[6]{2}$ is the distance at which the energy is minimized, and ε is the potential well depth at $\sigma\sqrt[6]{2}$. b) Depiction of the respective Lennard-Jones force graph where $\sigma\sqrt[6]{2}$ is the distance at which the force becomes zero. The interaction force becomes highly repulsive when r_{ij} is less than $\sigma\sqrt[6]{2}$, and attractive when it is greater than $\sigma\sqrt[6]{2}$.

at its minimum when the distance is $\sigma\sqrt[6]{2}$. Moreover, the VDW force is strongly repulsive when the two molecules are in close proximity, and attractive when they are far apart. Similarly to the energy potential, the repulsive part of the force is attributed to the $\frac{1}{r_{ij}^{13}}$ term of Equation 2.2.2 (see Section 2.2.2), and dominates the attractive force when the distance is less than $\sigma\sqrt[6]{2}$. The attractive part of the force is attributed to the $\frac{1}{r_{ij}^{7}}$ term, and dominates the repulsive force when the interatomic distance is greater than $\sigma\sqrt[6]{2}$. Overall, the force is attractive when the molecules are far apart, it becomes less attractive as the two molecules approach each other, it becomes equal to zero at distance $\sigma\sqrt[6]{2}$, and it becomes strongly repulsive for distances less than $\sigma\sqrt[6]{2}$. The sudden change in force direction and magnitude at distance $\sigma\sqrt[6]{2}$ can cause force rendering instabilities/artefacts, especially when the repulsive force reaches extremely high levels (when the electron clouds overlap). The literature describes the latter case as the "hard-surface problem", which current haptic technology is incapable of dealing with effectively (i.e. rendering such forces).

Ouh-Young [OY90] addressed this problem, and in his solution, GROPE III, he developed a bump checker that provided visual cues (flashing vectors) when two atoms started bumping to each other. His solution did not produce a smooth transition from low intensity attractive forces to highly intensive repulsive forces on the haptic device. It simply augmented the force feedback sensation with visual cues that simulated this transition in the user's mind.

Bayazit et. al. [BSA01] took a simpler approach and reduced any force greater than 1N (the devices limit at that time) down to 1N before rendering it on the haptic device. Their solution addressed the issue of device-rendering stability, but did not enable the user to distinguish the repulsive-force increments when the two atoms collide/overlap, or account for smooth force transitions. Like Bayazit et. al, Wollacott and Merz Jr. [WMJ07] introduced a repulsive-force cut-off value in their haptic rendering loop, but unlike Bayazit et. al, they scaled down all other force values according to this cut-off. Their approach gave them a wider range of repulsive forces to render, but did not help them to address sufficiently the problem of smooth force transitions.

Lee and Lyons [LL04] tried initially to scale down all interaction forces (without cut-offs) in order to be able to haptically-render the repulsive-force increments observed during interatomic penetration. They discovered that the magnitude of the forces, prior to the penetration, were too small to be detectable by the user. For this reason, they implemented a *virtual wall* analogy of haptic-force rendering (using a *god* object). Their solution permitted the rendering of the VDW forces until the interatomic distance becomes less than the sum of the atoms' VDW radii. In that case, their method treated the two atoms as being in contact, and added a spring-based force vector to the total force (equal to depth of penetration multiplied by a spring constant) if the atoms continued to penetrate further into each other. Lee and Lyons' approach provided an effective way to deal with the "hard-surface problem", while allowing the user to become aware of these interatomic penetrations and their underlying force variations. Subasi and Basdogan [SB06] adapted this method in their docking solution as well.

Similarly to Lee and Lyons, Lai-Yuen and Lee [LYL05] proposed a virtual wall approach where the interatomic collision forces were pre-computed and stored in a 3D grid, instead of being calculated on the fly. Their approach traded memory space for faster force computation, and force feedback rendering results. Lastly, Hou and Sourina utilized a linear smoothing method to soften the rapid force transitions. Their method was based on the smoothing function proposed by Gregory et. al.

[GME+00] for virtual-proxy-based 6DOF haptic rendering (the virtual proxy is a similar approach to the god object approach of virtual wall simulation).

In closing, the force-feedback rendering capability of the intermolecular interactions is a central aspect of any haptics-assisted docking application. It is important for the human perception of haptic stimuli that these forces are rendered at a rate of 500Hz (or even faster). Such fast rendering rates necessitate equally fast energy/force computational methods, and the grid-based approaches address this requirement efficiently. Nonetheless, as seen earlier, these approaches can induce rough force transitions at cell boundaries [WMJ07] (if these effects are not reduced appropriately), since VDW forces are exponential in nature, and tri-linear interpolations do not perform well on convex functions. Moreover, they cannot be applied to flexible docking problems (modelling receptor deformations), because of the time-consuming, offline calculations required to construct the grids. Lastly, none of the existing approaches can accommodate force calculations for large molecules. Ongoing research in the field is expected to generate faster and more flexible methods to compute intermolecular interaction forces. It is also expected to involve the discovery of force-rendering methods that can adapt efficiently to the limitations of current haptic technology, while maintaining a smooth, continuous, and distinguishable force feedback display. These premises form the main motivation of this thesis.

2.2.4 Docking Applications with Haptics

The potential benefits of integrating haptic technology in molecular docking solutions have been under investigation since the late 60s [BJOYBJK90]. 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, for many decades, 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.

The first attempts in the field were made by Frederick Brooks' team at the University of North Carolina. Brooks and his team, stimulated by the "Ultimate Display" vision of Ivan Sutherland [Sut65], initiated in 1967 a research project called GROPE, which lasted twenty three years. The project aimed to develop a haptic display system for 6DOF force-rendering of protein-protein interactions [BJOYBJK90]. Project GROPE involved three stages of development (GROPE I, GROPE II, and GROPE III), and started with a simple 2DOF system (force feedback only on x, and y axis), progressed to a 3DOF system and concluded with a 6DOF system, respectively. The last stage of the development was undertaken by Ming Ouh-Young [OY90], as part of his PhD thesis. Ouh-Young developed a 6DOF, haptics-assisted docking system called GROPE III (see Figure 2.6). His system utilized a modified Argonne E-3 Remote Manipulator (ARM) for ligand movement and force feedback display. It enabled the user to move/rotate the ligand around the receptor, and sense the interaction forces and torques on the ARM device. The 1DOF-twisting of ligand-specific, rotatable bonds was also facilitated by the device, but did not induce any force feedback. Force and energy calculations were accelerated based on a pre-computed 3D grid, and torques were derived from the force vectors using a Jacobian matrix. His system suffered from some problems (e.g. feedback rendering was around 60 frames per second, friction problems on the device induced noise on the force renderings) due to the lack of advanced computational and haptic technology. Nonetheless, it allowed Ouh-Young to prove his assertion, that haptic technology augmented by human experience and intuition can accelerate the docking process in an accurate and reliable manner.

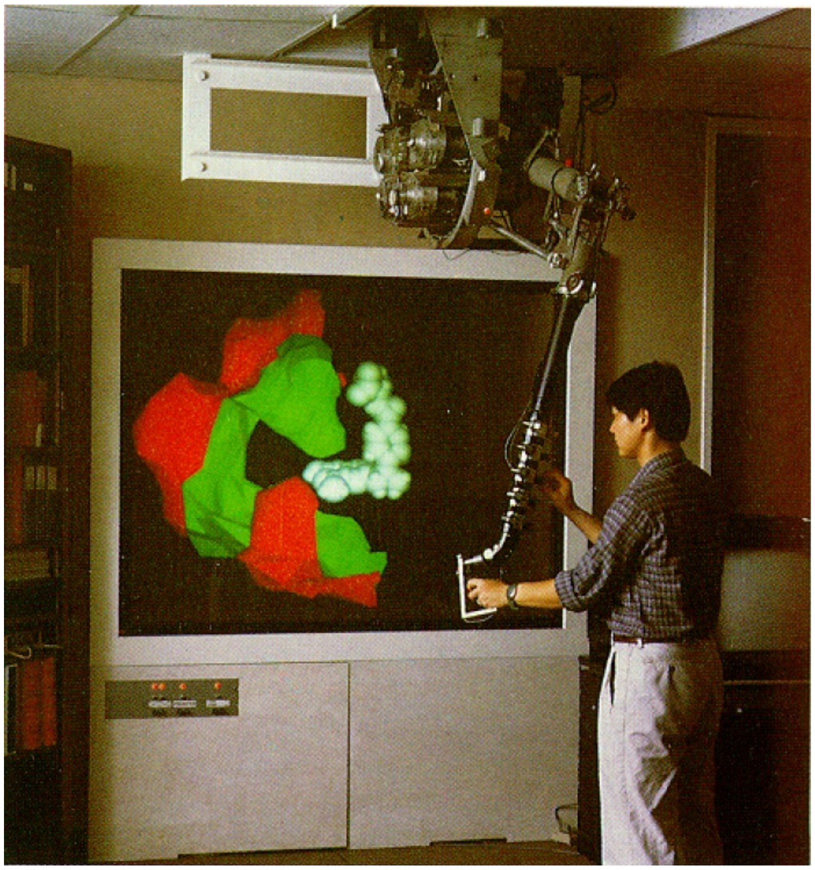


Figure 2.6: The GROPE-III docking system with the Argonne E-3 manipulator [BJOYBJK90]

Taking a different approach from Ouh-Young, Bayazit et al. [BSA01] integrated haptic technology into their motion planning method called obstacle-based, probabilistic roadmap (OBPRM). OBPRM is an automated method that samples the molecular-conformations space for possible docking sites and binding conformations, connects these findings into a roadmap, and then calculates the final docking path based on this roadmap. Like all probabilistic roadmap methods however, their solution suffers from a known problem, commonly referred to as *narrow passage* (it could not sample ligand conformations in narrow C-space regions). Bayazit et al. addressed this problem by coupling OBPRM with haptic technology. In their solution, the haptic device allowed the user to sample the conformations space, sense 3DOF, 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. Energy and force calculations modelled only VDW interactions, and accelerated with the use of a 3D grid. Ligand flexibility was partially addressed by allowing the planner to fine-tune the chosen ligand conformations (modelled as an articulated body) using energy minimization techniques (approximate gradient descent). The authors observed that this haptically-driven user intervention helped the planner to obtain better docking results.

Nagata et. al. [NMT02] designed a 3DOF, docking system for computer-aided drug design. They attempted to compute and render all non-bonded interactions (VDW, electrostatic, and hydrogen bonds) on the haptic device (in real time), without utilizing pre-computed grids. They discovered that they could not achieve these goals, since they lacked the necessary computational power. Nonetheless, they built a limited system that modelled the electrostatic interactions, between a single-charged, globular probe (as the ligand), and twenty, receptor atoms only. They concluded that a brute-force approach cannot compute the respective binding interactions at haptically-acceptable interactive rates without being given additional computer power (about 100-fold). Unlike Nagata et. al., Lee and Lyons [LL04] proposed a 3DOF, haptics-assisted, docking approach that improved the accuracy of grid-based, force/energy calculations, and smoothed the haptic rendering of the Lennard-Jones force field. Their solution pre-computed and utilized multiple 3D-grids for all energy/force calculations, and treated the "hard-surface", force-rendering instabilities (induced by the Lennard-Jones field) using a spring-like model. Their approach produced notable results, and has been applied to many other studies hence after [WMJ07, SB08].

Birmanns and Wriggers [BW03] augmented algorithmic docking with haptics, and developed a 6DOF, rigid-body docking application for biomolecular assembly. They integrated haptic technology into their SenSitus engine (a molecular-fitting and visualization engine), in order to assist the engine (through human intervention) to find the optimum docking pose. Unlike grid-based approaches, their solution utilized a standard cross-correlation function for modelling the energy, force, and torque quantities. Vector quantization techniques computed these quantities in real time, and provided stable haptic rendering rates. Users, who tested this system, reported that the fitting process was found to be surprisingly challenging even with the visuohaptic feedback. The authors attributed these difficulties to the known limitations of the cross-correlation function. Subsequent studies addressed partially these issues, as well as, the issues of molecular flexibility (by utilizing Topology-Representing neural Networks-TRNs [WCAK$^+$04]), and balancing visual and haptic rendering-rate disparities (by utilizing adaptive, visuohaptic rendering, and dynamic, mesh-simplification techniques [BBZW04]).

Lai-Yuen and Lee studied the problem of flexible-ligand docking, and implemented their methods into a haptics-assisted, docking system for computer-aided molecular design and assembly [LYL06a]. Their system utilized a proprietary (lab-built) haptic device capable of 6DOF HIP movement, and 5DOF force-rendering. The user could use the device to navigate the ligand around the receptors tertiary structure and sense the interaction forces. Torque-rendering was available, but only when the ligand collided with the receptor. The system accounted for the VDW interactions only, and it employed 3D-grids to accelerate the energy and force computations. Torques were computed as a function of force vectors acting on a pivot point (referred to as the ligands "center of weight"), and the "hard-surface" problem was addressed using a spring-like model. Molecular flexibility was taken partially into account, since they modelled the ligand molecule as an articulated body with torsional freedom. Unlike the previous approaches, ligand deformations did induce force and torque feedbacks. In addition to the haptic-related features, their solution incorporated an automated binding-pose search engine, called Nano-scale Docking and Assembly Simulator (NanoDAS). NanoDAS enabled the user to generate automatically a docking path, and then haptically explore it with the device. Given a user-specified, initial location for the ligand, NanoDAS utilized real-time, pose-search [LYL06b] and energy-minimization methods [LYL05], in order to generate and return to the user a feasible docking path.

Similarly to Lai-Yuen and Lee, Wollacott and Merz Jr. [WMJ07] developed a rigid-body, docking application for rational drug design. Their Haptic Application for Molecular Structure and Energy Refinement (HAMStER), provided 3DOF, force-rendering capabilities, and was designed to serve as a general purpose tool in drug discovery. HAMStER enabled the user to explore the receptor with the ligand, and receive, at the same time, visuohaptic stimuli. It utilized two different 3D-grids for energy/force calculations (one for VDW and one for electrostatic). The "hard-surface" problem was addressed similarly to Lee and Lyons [LL04], with the only difference that the total force rendered on the haptic device was the sum of the spring force and the interaction force (and not only the spring force). Wollacott and Merz Jr.

implemented this technique in order to achieve force continuity during hard-surface interactions. To achieve haptic stability, the application capped all forces greater than 1.5N down to 1.5N, and then scalled the smaller ones accordingly. In order to achieve interactive, frame rates, HAMStER incorporated several, computational-cost-cutting procedures such as, pre-computing the grid only around the active site (using hydrogen and generic heavy atoms as ligand atoms), treating large receptor molecules as graphic-objects and not as haptic-objects, and allowing only up to 50 haptic-objects rendered in a scene. Wollacott and Merz Jr. characterized as "haptic-objects" (in a virtual scene) only those objects capable of inducing force-feedback cues.

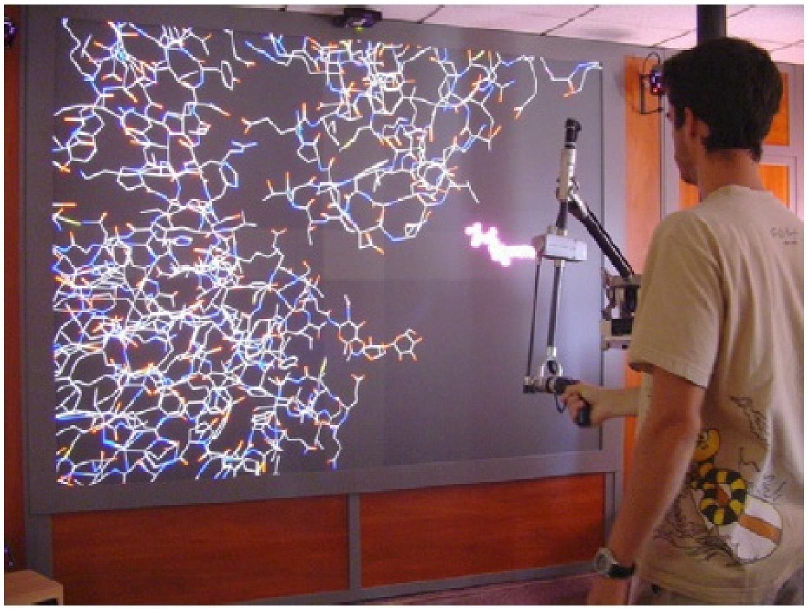


Figure 2.7: The 6DOF, flexible docking system by Daunay and Regnier [DR09] with the Virtuose haptic device

Subasi and Basdogan [SB06, SB08, Sub06], took a slightly different approach and proposed a hybrid system, that combined haptic technology with an off-line MD-simulation engine. It is common for MD simulators to stall in a problematic state known as *trapping* (the simulator is trapped in local, minimum-energy conformations instead of the global ones). Subasi and Basdogan suggested that user intervention could assist MD simulators to overcome such states. In their solution, they allowed the user, via the haptic device, to locate the binding site on the receptor, and to roughly align the ligand within that site. Given that rough alignment, the MD engine could then execute optimum rigid transformations on the ligand, and determine the ligands final pose inside the binding site. To achieve this fine-tuning, the engine had to minimize the total distance error between the current, and the simulated coordinates of the ligand atoms. Energy and force feedback cues were computed based on the 3D-grid method, proposed by Lee and Lyons [LL04]. The "hard-surface" problem was addressed in a similar way, as well. Subasi and Basdogan also proposed a method for haptics-assisted exploration of large molecular structures, rendered in high resolution. Their method was called Active Haptic Workspace (AHW), and enabled the user to explore the molecular surface in small fragments at high resolution. Experimental results showed that, their system helped non-specialist users to successfully identify the binding site, and roughly pose the ligand inside it.

Within project CoRSAIRe, Ferey et. al. [FNM$^+$09] designed and implemented a multisensory virtual reality system for rigid-body, protein-protein docking. Their system combined multimodal (3D mouse, haptic device) interactive technology with the screening power of an automated docking approach. With their system, the user performed the docking process in three stages, two of which required human intervention. The first stage enabled the user (via a multimodal feedback environment) to reduce the pose sampling space using pattern matching skills and protein-protein docking expertise. The second stage allowed the refinement and screening of the conformations selected in the first stage, by an automated docking procedure. Finally, the third stage facilitated the multimodal exploration of the results produced in the second stage. Their system was capable of rendering 3DOF, force feedback cues, and supported two different force calculation modes. The first one calculated the forces using the standard VDW and electrostatic models (i.e. Lennard-Jones and Coulobic), whereas the second one derived them from the depth-value generated when two molecules collide (using a spring-like approach). Unlike other approaches, their system did not account for all interatomic VDW interactions, but only for those involving the surface atoms. To provide to the user an immersive docking experience, Ferey et. al. augmented the visual and force feedback cues with auditory stimuli (e.g. the system used the French word "complementaire" with varying pitch to sonify surface complementarity), as well.

Daunay and Regnier studied the problem of molecular flexibility during docking. Their initial attempts [DMR07a] produced a docking system with 6DOF, force/torque-rendering capabilities but did not account for molecular flexibility. Further improvements to that solution led to a 6DOF, force/torque-rendering docking system in which, both ligand and receptor molecules were considered flexible (see Figure 2.7) [DR09]. Their system modelled both bonded and non-bonded interactions, and utilized a simulation engine for the respective energy computations. An energy minimization process (developed by the authors [DMR07b]) assisted the engine to achieve a better docking fit, and induced receptor and ligand structural deformations. 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 proposed by the authors [DMR07b]. The system could not support haptic refresh rates, since its energy minimization and force calculation techniques (used by the simulator) were computationally intensive and time consuming, and could not provide force/torques updates at 500Hz. To address these issues, the authors utilized wave theory and modelled the transmissions (i.e. ligand movement, force/torque feedback) between the haptic device and the simulator accordingly [DAMR07]. Wave theory states that, if all components of the transmission system (i.e. simulator and haptic device) are passive in the wave domain, and the time delays are constant then, these transmissions will be stable and robust whatever the delays were. Daunay and Regnier exploited this property, developed the appropriate wave transformations, and managed to bridge the rate disparities between haptic rendering and simulation (greater then 400Hz per response) and provide a smooth and continuous force feedback sensation to the user. They were the first ones to build a system for flexible, receptor-ligand docking.

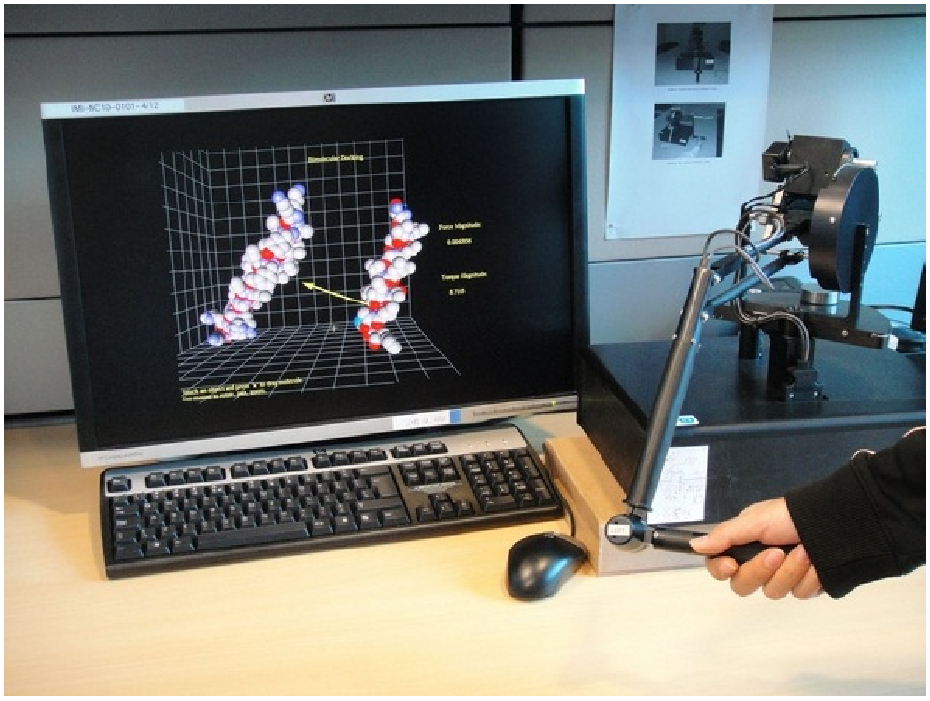


Figure 2.8: The HMolDock system with a 6DOF PHANToM haptic device [HS11]

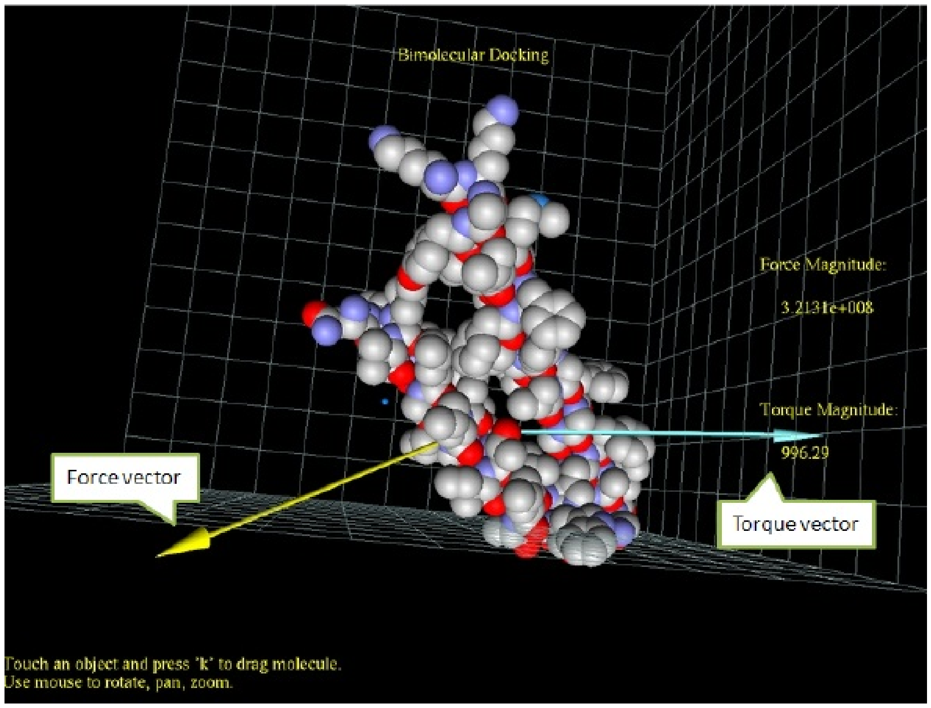


Figure 2.9: An HMolDock screenshot with the force and torque vectors visualized [HS11]

Similarly to Daunay and Regnier, Hue and Sourina [HS11, HS10] proposed and implemented a 6DOF, haptics-assisted, docking system tailored for research in helix-helix docking (see Figure 2.8). Their system, named HMolDock (Haptic-based Molecular Docking), utilized a 6DOF haptic device, and enabled the user to sense the interaction forces/torques while docking Transmembrane a-helices. Earlier versions of HMolDock incorporated the works of Sourina et. al. [STW09], and Hue and Sourina [SH10]. Like its predecessors, the 6DOF version of HMolDock modelled only the VW interactions, did not employ 3D grids to accelerate the energy/force calculations, and treated the molecules as rigid-bodies. Unlike its predecessors however, it provided a 6DOF, haptic rendering environment (instead of a 3DOF one), and addressed the "hard-surface" problem using the *virtual proxy* method proposed by Gregory et. al. [GME^{+}00] (instead of neglecting the problem). Moreover, it augmented the haptic sensations with visual cues reflecting the direction and magnitude of the force/torque vectors (see Figure 2.9), and allowed the HIP to be repositioned dynamically at any location on the ligand's surface. The authors asserted that the repositioning of the HIP helped the users to attain a better understanding of the intermolecular interactions.

Zonta et. al.[ZGAB09] implemented the first affordable, freeware docking application called ZODIAC. Their system addressed ligand flexibility but used a third-party library to accelerate force computations; namely, the OpenBabel library. ZODIAC integrated OpenBabel, and utilized the library for force calculations and for molecular modelling/rendering. Zonta et. al. did not address the "hard-surface" problem directly. They used instead the VDW repulsive forces (i.e. relied on their steepness) as a means to emulate rigid body clashes. Their system was able to model ligand flexibility at haptic refresh rates, but only for very small ligands (i.e. comprising 16 atoms).

Other studies pertinent to molecular docking with haptics included the works of Krenek, [Kre01], Sankaranarayanan et. al. [SWS+03], Sauer et. al. [SHO04], and Bivall et. al [PCT+07, BAT11]. These studies investigated primarily the importance of haptic technology in e-learning and education (e.g. teaching structural biology to users/students). Krenek identified the functional characteristics of a successful, haptics-assisted, docking application, and explored how these characteristics assist a user/student during docking. Sankaranarayanan et. al. developed a multi-modal system with haptic rendering capabilities, and evaluated its usefulness in teaching molecular biology to high school students. Sauer et. al. developed a system that enabled students to haptically sense atomic interactions (atomic bonds) and investigated the benefits of such a system in teaching molecular assembly courses. Bivall et. al implemented a 3DOF, haptics-assisted docking system and studied whether or not haptic technology could help inexperienced students understand the concept of protein-ligand docking, and identify the correct docking conformations. These studies provided valuable feedback pertaining to why and how haptic technology could support their education.

Lastly, some authors utilized haptic devices to interact with molecules during molecular dynamics simulations and sense the respective forces [FDGB08, SGS01], explore interactively (with a water-probe) the solvent accessible surface (ISAS) of a receptor and sense the receptor's hard surface [SHL09], deform an elastic network model of a biomolecule by applying forces to individual atoms [SLH11], or visualize in a web browser the potential energy surfaces and wave-packet dynamics of molecular systems, and render the underlying forces back to the user [DJMH05]. Although these studies did not address the problem of molecular docking directly, they did propose interactive methods that could be adapted "as is" by haptics-assisted docking solutions.

Table 2.1: A comparison of existing haptics-assisted docking systems. *(VDW=van der Waals, ES=Electrostatic)

Published Works	**Description**	**# of Atoms (Rec./Lig.)**	**Force Model***	**Calculation Method**	**Compute Unit**	**Flexibility**	**Haptic H/W**	**Haptic Feedback**	**Haptic Frame Rates (Hz)**	**Software Availability**
Brooks et.al. [OY90]	An interactive receptor-ligand docking system	600/60	VDW+ ES	Pre-comp. Grids	CPU	Ligand	Proprietary	6DOF	60	Proprietary
Bayazit et.al. [BSA01]	A hybrid system, that uses haptic feedback to guide an obstacle-based probabilistic roadmap motion planning algorithm for receptor-ligand docking	2500/20	VDW	Pre-comp. Grids	CPU	Ligand	Phantom	3DOF	1000	Proprietary
Nagata et.al. [NMT02]	An interactive receptor-ligand docking system	20/20	ES	Brute Force	CPU	Rigid	Phantom	3DOF	1000	Proprietary
Lee and Lyons [LL04]	An interactive system for studying receptor-ligand docking interactions	532/30	VDW+ ES	Pre-comp. Grids	CPU	Rigid	Phantom	3DOF	1000	Proprietary
Birmanns and Wriggers [BW03]	A system for interactive fitting atomic structures into low-resolution EM density maps and biomolecular assembly	5000/1	A fitting cross-correlation function	Vector Quantization	CPU	Rigid	Phantom	6DOF	1000	Proprietary
Lai-Yuen and Lee [LYL06b]	An interactive receptor-ligand docking system for computer-aided molecular design	2430/16	VDW	Pre-comp. Grids	CPU	Ligand	Proprietary	5DOF	1000	Proprietary
Wollacott and Merz [WMJ07]	An interactive receptor-ligand docking system for rational drug design	18/1	VDW+ ES	Pre-comp. Grids	CPU	Rigid	Phantom	3DOF	1000	Proprietary
Subasi and Basdogan [SB08]	An interactive receptor-ligand docking system	3400/24	VDW+ ES	Pre-comp. Grids	CPU	Rigid	N/A	3DOF	1000	Proprietary
Ferey et.al. [FNM$^+$09]	A hybrid, multisensory system, for protein-protein docking. Haptic feedback combined with automated docking	N/A	VDW+ ES	Pre-comp. Grids	CPU	Rigid	Virtuose	3DOF	1000	Proprietary
Daunay and Regnier [DR09]	A hybrid system, for protein-ligand docking. Haptic feedback combined with automated docking	N/A	Bonded+ Non-bonded	MOE S/W Package	CPU	Receptor + Ligand	Virtuose	6DOF	N/A	Proprietary
Hou and Sourina [HS11]	An interactive helix-helix docking system	266/154	VDW	Brute force	CPU	Rigid	Phantom	6DOF	1000	Proprietary
Zonta et.al. [ZGAB09]	An interactive protein-drug docking system	5537/16	VDW	OpenBabel Lib.	CPU	Ligand	Novint	3DOF	1000	Free
Anthopoulos et.al. [APGB14]	An interactive system for drug design	30000/60	Bonded+ Non-bonded	Cut-off based grid-cell traversals	GPU	Receptor + Ligand	Phantom	6DOF	33	Proprietary
Iakovou et.al. [IHL15]	An interactive system for the study of large protein-protein and protein-ligand docking interactions	200000/200000	VDW+ ES	Cut-off based proximity querying	GPU	Rigid	Phantom	3DOF	1000	Free

Despite all this research effort however (see Table 2.1), these haptics-assisted docking systems suffer from issues related to pre-computed grids [OY90, BSA01, LL04, Biv10, LYL06b, MCET05, SWS$^+$03, SB08, WMJ07] (see section 2.2.3), utilize proprietary hardware [LYL06b], or even fail to address the 2ms time constraint effectively [DR09, ZGAB09]. Moreover, none of these systems can accommodate the study of large protein-protein docking, which limits further the scope and usefulness of such applications for the molecular docking community. The fact that only very few of these systems are freely available to the community [ZGAB09] is another reason why the adoption rate of this technology has been slow.

2.3 Conclusions

The molecular docking field can benefit by the use of haptic technology. There are good indications that haptics-assisted, docking solutions can improve the docking results produced by their algorithmic counterparts. However, haptics-assisted docking neither received substantial attention from the research community, nor has it been exploited commercially. Existing interactive docking systems do not model sufficiently the binding interactions, and thus are often considered to be unfit for commercial use. Moreover, they cannot manage docking problems (rigid or flexible) of large biomolecules, and they have thus been limited to a) rigid protein-ligand docking problems of molecules comprising a couple of thousand of atoms each, and b) rigid receptor-flexible ligand docking problems of very small ligand molecules. Proprietary haptic devices and rendering software raise additional barriers, and constrain further the commercial applicability of their docking solutions. The next chapters discuss the design and implementation of a haptics-assisted docking application that attempts to address many of these issues.

Chapter 3

Building a haptics-assisted docking application

3.1 Introduction

Molecular visualization, haptic navigation, force calculation and rendering are the core functional blocks of any autonomous haptics-assisted docking system. 3D molecular visualization enables the user to depict on screen chemical and spatial information about the structures, and visually identify potential binding sites or docking conformations, during the simulation. Haptic navigation on the other hand defines how the user interacts with the virtual world. It sets the dimensions of the virtual workspace, and dictates the rules governing this interaction, i.e. user movement and accessibility within the workspace. Finally, the haptic rendering of the interaction forces allows the user to sense the repulsive or attractive forces acting between the two molecules at various potential docking poses, and use this input in order to score these poses (e.g. score for chemical complementarity) and select the most probable one. Consequently, an equally important aspect of the system is the force field used for modelling these interaction forces. A haptics-assisted docking system has to address all of the above aspects, since failure to do so can hinder drastically the effectiveness of the given application, e.g. inability to view and access parts of the molecular structure, improper modelling of the interaction forces etc.

This chapter describes the design and implementation of the first two building blocks (i.e. visualization and navigation) within Haptimol_RD (Figure 3.1), part of which is a novel haptic navigation technique suitable for the haptic exploration of large virtual environments. The chapter also states the force field and the file format used for modelling (calculating) the interaction forces, and for describing the 3D structure of the molecules, respectively. The discussion in this chapter begins with an outline of the potential uses of Haptimol_RD.

3.2 Potential Application of Haptimol_RD

Haptimol_RD is a software application, developed from scratch for the purpose of this research, capable for the haptics-assisted rigid docking of very large biomolecules. It is a part of the Haptimol suite which, in addition to Haptimol_RD, features the following two applications: a) Haptimol ISAS [SHL09] and b) Haptimol ENM [SLH11]. The former application allows the user to interact with the solvent accessible surface of biomolecules, whereas the latter one allows the user to deform biomolecules by applying forces to atoms in an elastic network model. Unlike Haptimol ISAS and Haptimol ENM, Haptimol_RD facilitates the study of protein-protein and protein-small molecule interactions, enabling the user to understand biomolecular interactions at a fundamental level. As such its applicability can be wide-ranging and far-reaching within all areas of biomolecular research.

In that regard, Haptimol_RD can be used in structure-based computer-aided drug design. Even though the system cannot screen a large number of compounds for

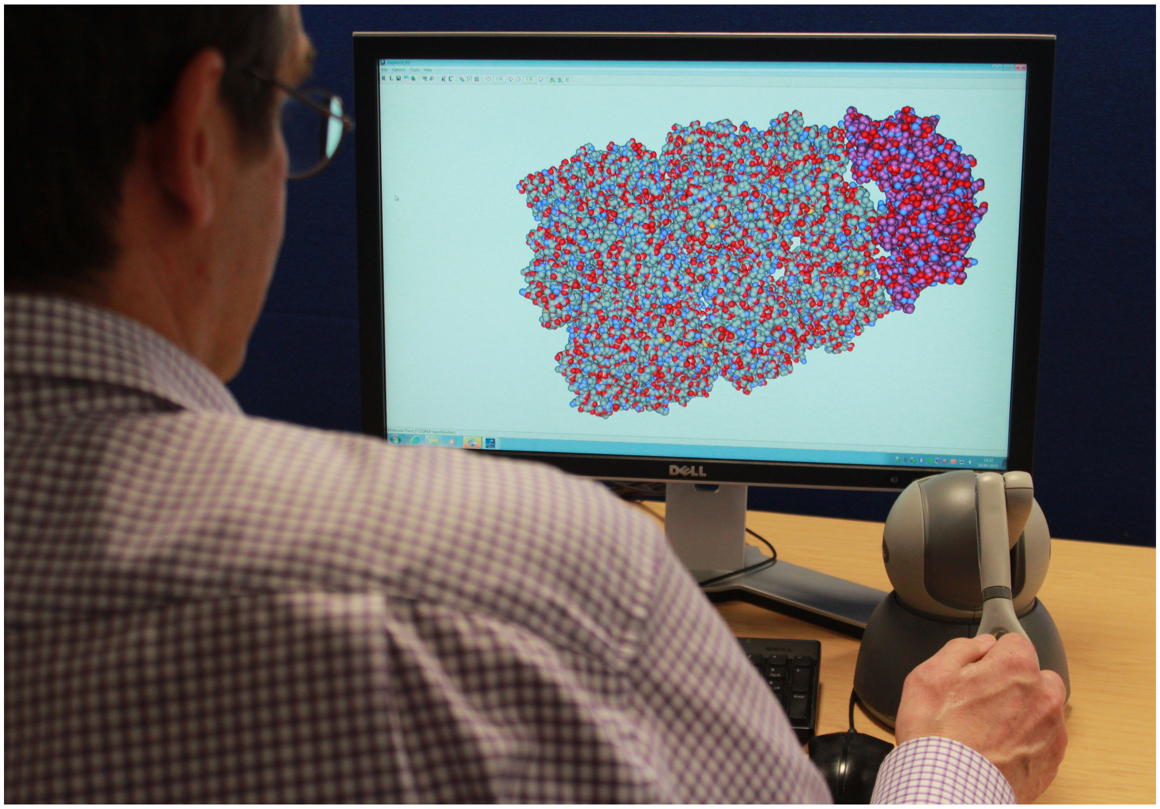


Figure 3.1: Conducting an interactive rigid docking simulation with proteins GroEL (larger molecule) and GroES (smaller molecule), Haptimol_RD, and the 3DOF Geomagic Touch haptic device. Both molecules are defined in the PDB file with accesion code 1GRU where they are in a bound conformation. The user controls GroES and feels the interaction forces using the haptic device.

a particular target (as automated methods do in virtual screening), it can be used subsequent to automated methods when a small number of lead compounds have been identified. Namely, the user can utilize the software to visualize these lead compounds in their docking conformations, feel the underlying interaction forces and improve upon or reject these conformations based on user knowledge, experience and expertise.

In academic context, Haptimol_RD can be used as a highly engaging and informative tool for teaching students about the nature of molecular interactions and biomolecular function. Experience with existing biomolecular haptics software [SWS$^+$03, SHO04, SB08, BAT11] has demonstrated that interactive docking systems are excellent tools for helping students understand the process of molecular binding.

Haptimol_RD can also be used by researchers, both in academic and industrial contexts, in order to investigate protein function at a molecular level. Biologists and biochemists are often interested in particular protein-protein interactions as they underpin biological processes. In such cases, Haptimol_RD provides an interactive environment with which expert users can test new ideas and hypotheses. As stated in [far14], automated methods are very poor at teaching us about the process of docking itself, e.g. whether electrostatic steering is involved in the binding process. Haptimol_RD has already proven that can be very useful in this respect [ILH16].

Lastly, biotechnology is another area of application for Haptimol_RD. For example, enzymes are used in the production of paper, in the food and drinks industry, as detergents, in textile production and in the production of biofuels. In the case of the latter, an enzyme can be used to catalyse the conversion of plant cellulose to glucose. Molecular-level understanding of the mechanism of this enzyme could lead to genetically engineered enzymes with improved efficiency. Haptimol_RD can be the test-bed for such investigations.

3.3 Atomic coordinates

The atomic coordinates describe the 3D structure of the molecule, and facilitate molecular visualization. X-Ray Crystallography and NMR spectroscopy are the two most common techniques used for obtaining these coordinates. The former identifies atom positioning by measuring the diffraction patterns of an X-ray beam sent through a molecule in crystalized form; whereas the latter obtains detailed information about the structure, by exploiting the phenomenon of nuclear magnetic resonance and its effects on the intramolecular magnetic field surrounding each atom. Upon creation, the atomic coordinates along with additional information about the structure (such as atom names, residue names, primary and secondary units, chain IDs, author/experiment details etc) are saved within simple text files, and stored in large databases for further study and research.

One such database is the Protein Data Bank (PDB) founded in 1971. PDB acts as an open-access worldwide archive of structural information pertaining to biological molecules [BWF$^+$00]. It also defines a data format (with the .pdb extension) used for storing atomic coordinates and other molecular information. Although coordinate information can be found in various file formats (the Jmol Wiki [Jmo] provides an extensive list of them), pdb is perhaps the one format most widely used. Haptimol_RD utilizes this format in order to obtain atom-based force parameters from the *pdb2gmx* tool of Gromacs [vdSLHtGdt13], and to visualize the molecular structures. The next two sections describe how this is done.

3.4 Force field

As for most haptics-assisted, interactive docking approaches, Haptimol_ RD models only the vdW and electrostatics interactions. The vdW interactions are modelled by the Lennard-Jones potential and the electrostatic interactions by Coulomb's law using Equation 2.2.8. The torques acting on those molecules due to the VDW and electrostatic interactions are not modelled, as most low-cost haptic devices are unable to render them. A graphical depiction of the torques might be one way to address this issue but it is not examined in this thesis. To obtain values for the parameters A_{ij}, B_{ij}, q_i and q_j in Equation 2.2.8 the Gromos54a7[SEC+11] force field is used, as specified and implemented in Gromacs version 4.6.2 [vdSLHtGdt13]. Specifically, Gromos54a7 provides values for the Lennard-Jones parameters A_i, B_i and A_j, B_j, and the Coulombic parameters q_i and q_j of atoms i and j, respectively. Using these Lennard-Jones values and the formulas $A_{ij} = \sqrt{A_i \times A_j}$ and $B_{ij} = \sqrt{B_i \times B_j}$, the terms A_{ij} and B_{ij} are finally computed. Moreover, the Coulomb constant $\frac{1}{4\pi\epsilon_0}$ is set equal to 138.935485 $kJ\ mol^{-1}\ nm\ e^{-2}$, and the ϵ is set equal to 1.0, assuming interactions take place in vacuo. The total force is measured in $kJ\ mol^{-1}\ nm^{-1}$. The actual command used for obtaining these values is the following,

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

where xxx is the molecules pdb code. *pdb2gmx* is a Gromacs tool, that processes a pdb file, adds the necessary hydrogens in the molecular structure, and returns the actual Gromos54a7 force field topology file (*.top) containing the nonbonded parameters (information about this tool can be found in Gromacs manual [vdSLHtGdt13].) It should be noted that Gromos54a7 models implicitly the hydrogen-bond interactions, i.e. it accounts for those interactions within the values of its electrostatic terms.

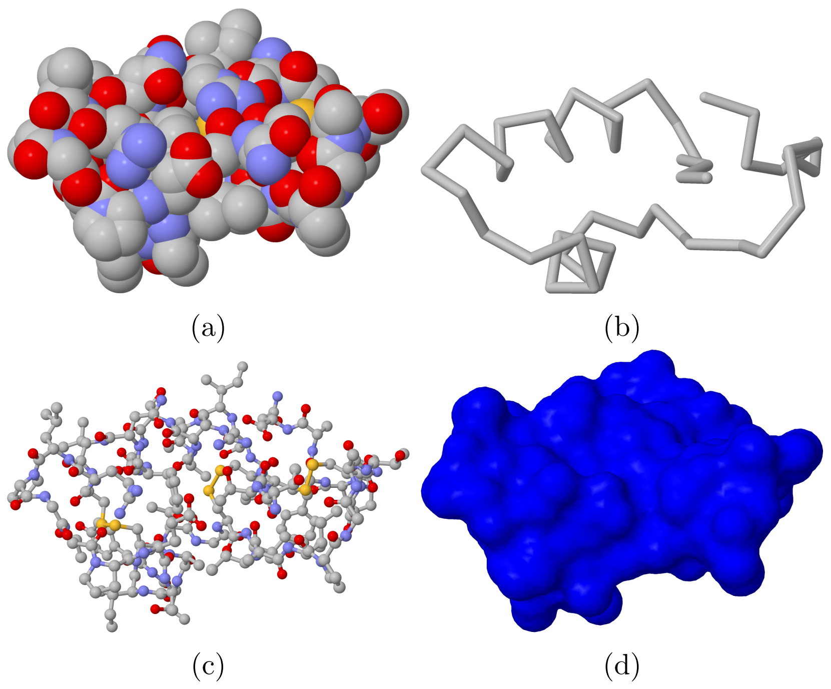


Figure 3.2: The protein Crambin (PDB code: 1CRN) visualized using the: (a) space-filling, (b) backbone, (c) ball ans stick, and (d) surface models. All models were visualized using the JSmol viewer provided by the PDB website (*http://www.rcsb.org*).

3.5 Visualizing molecular structure

The field of molecular graphics provides several approaches for visualizing molecular structure, each one of which renders and describes different chemospatial characteristics of the structure (Chimeras online user guide [chi] provides a list). In interactive haptics-assisted molecular docking however, the models most commonly used are the space-filling [LL04, LYL06a, WMJ07, STW09], backbone [ZGAB09], ball and stick [FDGB08], and surface models [FNM$^+$09], or in some cases a combination of them [BJOYBJK90, DMR07a, SB08]. The space-filling method models the molecules as clusters of spheres of radius equal to the atoms van der Waals radii [Bon64] (Figure 3.2a); whereas, the backbone method renders the structural skeleton of the molecule, i.e. C_α atoms connected with rods (Figure 3.2b). Likewise, the ball and stick method models atoms and atom bonds using coloured spheres, and cylindrical sticks, respectively (Figure 3.2c); whereas the surface method provides a volumetric representation of the molecule, and renders the structure as a continuous 3D mesh (Figure 3.2d). Haptimol_RD implements the first two models (i.e. space-filling, backbone), and allows the user to choose between them at runtime (Figure 3.3c,d) via menu or button commands.

As any real-time graphics system, the rendering performance (i.e. graphics frame rates attained) of Haptimol_RD is affected by the size of the geometry visualized. Namely, the larger and more complex the molecules are the slower the graphics refresh rates become. For this reason, three different rendering approaches were examined during the implementation of the space-filling and backbone models (Figure 3.3). The goal of that investigation was to identify an approach that would enable the application to render the largest molecules possible, at refresh rates greater than or equal to 30Hz. The first approach utilizes the OpenGL API (Application Programmable Interface) and its primitives (i.e. spheres, cylinders) in order to render atoms and rods (Figure 3.3a,b). This approach, although trivial to implement, performed worst than the other two, since it necessitated the transfer of the entire geometry (i.e. slow memory operations) from CPU to GPU, as well as the rendering of a large number of triangles (i.e. 272 per sphere), at each frame. To reduce this geometry-transferring overhead, the second approach utilizes the Vertex Array Object (VAO) and Vertex Buffer Object (VBO) features of OpenGL, and describes directly on the GPU the geometry of a single atom and rod. Atom/rod population is then achieved by invoking repeatedly these predefined objects, and using the OpenGL Shading Language (GLSL) for colouring and transforming the underlying geometry on the GPU (Figure 3.3c,d). This method produced better visual results than the initial approach, and

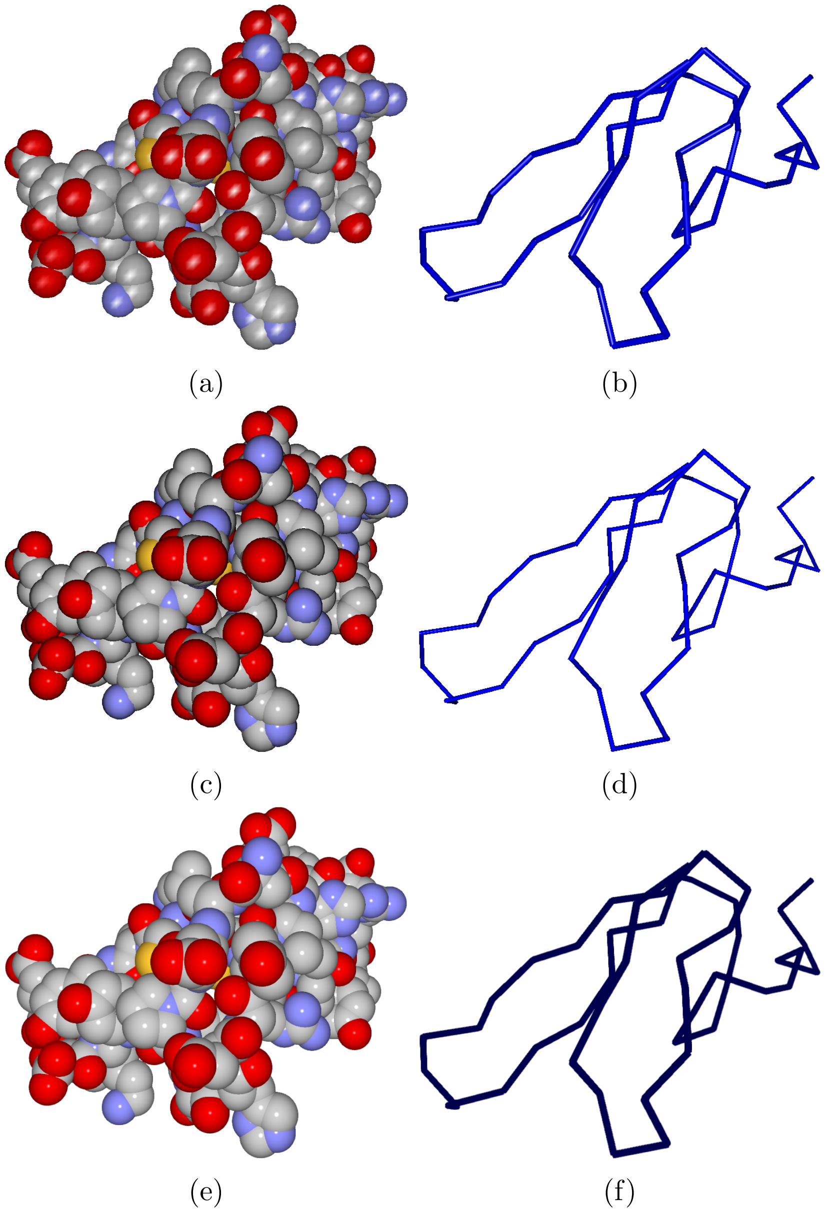


Figure 3.3: The protein Epidermal Growth Factor (PDB code: 1NQL) displayed in space-fill (a,c,e) and backbone (b,d,f) modes using the three rendering approaches examined in this thesis. (a)(b) The molecule is rendered using standard OpenGL commands and primitives. (c)(d) VAO and VBO objects describe the structure of a single atom and rod on the GPU, and the molecule is rendered by populating and adjusting the size and positioning of these structures using OpenGL and GLSL. Haptimol_RD visualizes molecular structure using this approach. (e)(f) Molecular structure is described as quads and ray traced on the GPU using the impostor-based method proposed by Easdon [Eas13].

achieved performance improvements by a factor of more than 2.5 times. However, it could not accommodate structures comprising more than forty thousands atoms each, due again to the performance penalties incurred by the large amount of triangles (the same as before) rendered by the graphics card and resident CPU-GPU memory transfers, i.e. transferring atom coordinates, radius and colour. The last approach addresses both of these issues (i.e. number of triangles and CPU-GPU memory transfers) by employing the quad, impostor-based ray tracing method proposed by Easdon [Eas13] (Figure 3.3e,f). Using VAOs and VBOs, this method describes all atoms/rods on the GPU as quads (i.e. impostors), and then uses the GLSL to ray trace (render) the actual spheres/cylinders on these quads directly on the GPU. This method achieved the best results both visually (i.e. anti-aliased smooth spheres) and performance wise out of all, and was able to attain the targeted refresh rates for molecules comprising almost two hundred thousand atoms each. Unlike the other rendering approaches the performance of this method is affected by how the molecules are positioned along the z axis. A zoomed in view of the molecules increases the number of screen pixels to be ray traced, which penalizes the method's rendering performance, and vice versa. For molecules up to 190k atoms however, the impostor approach was able to attain refresh rates higher than 30Hz, indifferent of the zooming levels applied.

Figure 3.4 displays the performance measurements recorded during the testing of the three rendering approaches with molecules of various sizes. All tests were conducted on a 8GB, 64bit Windows 7 PC with a 2.93GHz Intel Core i7 CPU and an NVIDIA GTX580 GPU. The values reported (per method and test case) are the averages of the frames per-second recorded at each second, within one minute of display time, rounded to the first significant digit. Evidently the impostor-based ray-tracing approach outperforms consistently the other two rendering methods in all test cases.

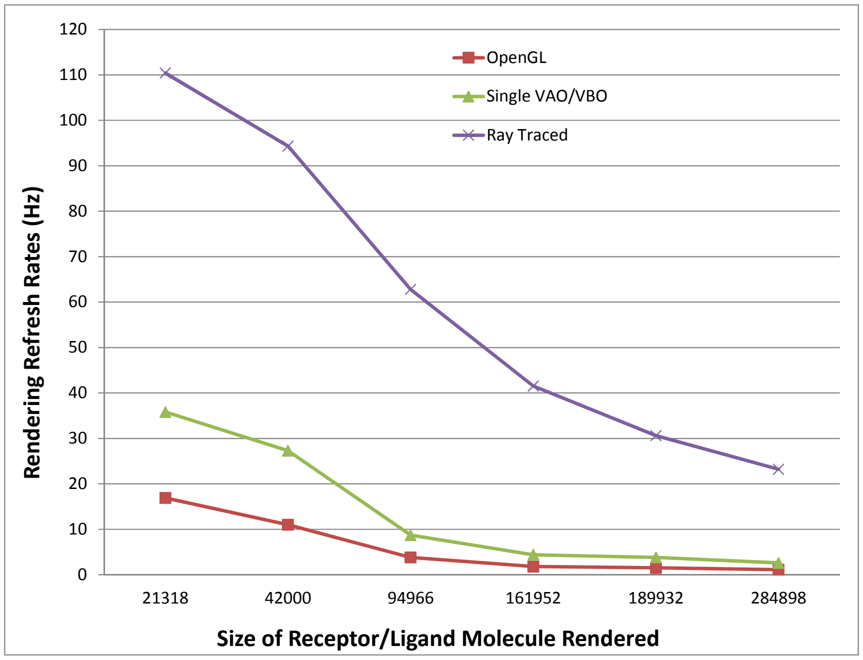


Figure 3.4: Performance measurements, in frames per seconds, for the three rendering techniques discussed in this chapter, as executed on a 2.93GHz Intel Core i7 CPU and an NVIDIA GTX580 GPU. The rendering involved receptor/ligand molecules of sizes ranging from 20 up to 285 thousand atoms each. *OpenGL* refers to the first method, *Single VAO/VBO* refers to the second method, and *Ray Traced* refers to the third method, as they are described in Section 3.5 respectively.

However, this approach has a negative impact on the performance of the GPU-based force calculation method discussed in Chapter 5 (i.e. performance penalties range from 0.4ms up to 3ms), since both methods compete for the same GPU resources. A system with a dual GPU configuration can address this issue by assigning one GPU to the graphics rendering and the other to the force calculation routines. However, a typical desktop/laptop configuration comes with only one GPU. In order to accommodate the widest user base possible, Haptimol_RD renders molecular structure using the second approach and not the third one.

3.6 Haptic Navigation

The haptic navigation of very large and geometrically complex structures is not a trivial task. The workspace dimensions of a standard 3DOF haptic device (e.g. 3DOF Geomagic Touch) is limited, which often leads to size mismatches between the haptic and the virtual world workspaces. A straightforward solution to this is to scale up/down the virtual workspace and map it to the haptic workspace. However, this solution is suitable only for virtual objects with smooth and continuous surfaces (i.e. without small bumps, grooves or cavities), since otherwise the resultant scaling could lead to substantial surface-detail distortions. For instance, a large virtual object with surface bumps or grooves, after being scaled down, might have these surface details compressed to a point that they become almost continuous and thus unrenderable to the haptic device. In the case of haptics-based molecular docking, this could easily mean that the pockets and grooves on the protein surface might become undetectable to the user, hindering drastically the haptic exploration of the potential binding sites, as well as the effectiveness of the respective docking simulation.

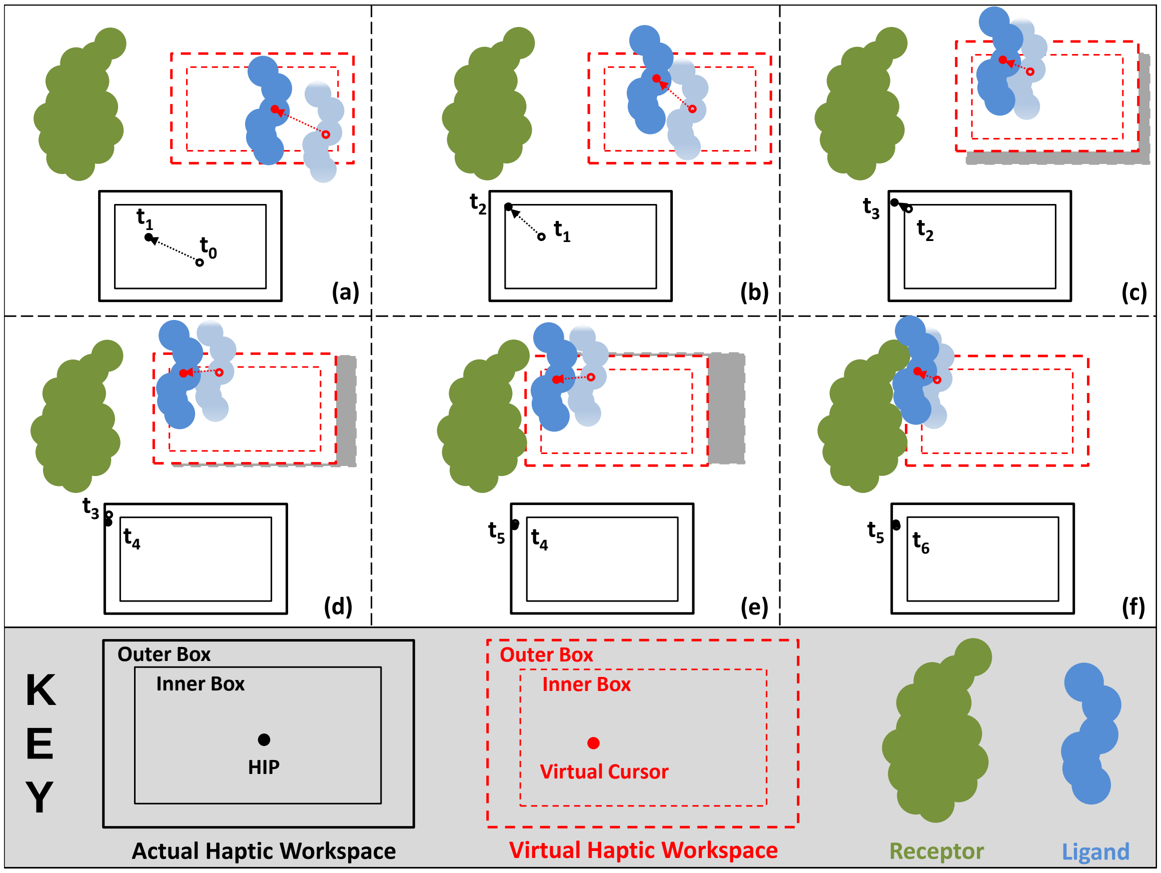


Figure 3.5: A 2D conceptual illustration of the Virtual Haptic Workspace (VHW) implemented in Haptimol_RD. The HIP moves within the actual haptic workspace. This movement influences virtual cursor movement within the VHW, which in turn induces ligand, and/or VHW movement. The black and red arrows give the direction of the HIP and cursor displacements, respectively. Likewise, the black and red unfilled circles, the light blue structure, and the grey box display the last positions of the HIP, cursor, ligand and VHW, respectively. All displacements are sampled at consecutive haptic frames. (a) The HIP moves within its inner box, causing an equivalent position control displacement of the cursor and ligand within the VHW and virtual world respectively. (b) The HIP moves and intersects the borders of its inner box. This is translated to position and rate displacements, and the result is applied to the cursor/ligand. (c)(d)(e) The HIP moves within its outer box. The resultant rate control displacement causes the cursor to intersect/overrun the borders of the VHW inner box. As a result, the VHW is translated towards the same direction (see the displacements of the red and grey boxes) and brings the cursor back within the inner box. (f) Again the HIP moves within its outer box, and the resultant rate control displacement causes the cursor to overrun the borders of the VHW inner box. In this case however the ligand collides with the receptor at multiple points, and as such no VHW position updating takes place. From this point on, HIP displacement will result in cursor/VHW displacement only if the given ligand displacement produces collision free results.

Haptimol_RD addresses this issue by implementing the concept of a Virtual Haptic Workspace (VHW). The VHW is a movable virtual workspace of the same size as the device workspace. Movement within the VHW is controlled by a virtual haptic cursor attached to the ligand's centre of mass. The VHW boundaries are updated (when necessary) in real-time as the haptic interface pointer (HIP) moves within the actual device workspace and updates the cursor's position within the VHW; as such the VHW is not constrained to the 3D coordinate space of the device (Figure 3.5). This movable haptic workspace allows the ligand to explore/interact with receptors of arbitrary size (while keeping the receptor fixed in space), and enables the method discussed in Chapter 6 to resolve efficiently intermolecular collisions at multiple points. Real-time rotation (using the Arc ball method described by Stocks [Sto10]) of the receptor, ligand, or both (i.e. global rotation of the scene) is provided to ensure that all parts of the receptor/ligand structures are viewable and accessible to the user. The method allows for position and rate control displacements. To decide whether to apply a position or rate control displacement, the method uses the idea of a navigation cube [SHL09]. The navigation cube is defined by two concentric boxes, an outer and an inner. The boundaries of the outer box match in size the boundaries of the actual haptic workspace, with the inner box being a scaled down version (approx. 80% of the dimensions) of the outer box. A similar navigation cube is applied to the actual haptic workspace as well. Within the virtual workspace, position control displacements are induced when the position of the virtual haptic cursor does not exceed the inner box boundaries, and rate control displacements are induced when it does. During rate control, the displacement vector updates the coordinates of both the virtual object (ligand) attached to the cursor and of the virtual workspace(i.e. inner and outer box), whereas during position control, only the coordinates of the virtual object are updated (Figure 3.6). The virtual workspace coordinates are also updated when the user applies a global rotation to the scene. Similar to the VHW, position control displacements, in device workspace, are prompted when the HIP moves within the respective inner box boundaries, and rate control displacements are prompted when it does not. As stated earlier, the virtual cursor moves according to the displacement of the HIP. When the HIP induces a position control displacement, the resultant displacement vector is applied to the virtual cursor without any modification. Under rate control however, the displacement vector **r** applied to the virtual cursor is given by Equation 3.6.1,

$$\mathbf{r} = < \frac{(HIP_x - IB_x)}{IB_x}, \frac{(HIP_y - IB_y)}{IB_y}, \frac{(HIP_z - IB_z)}{IB_z} > \tag{3.6.1}$$

where HIP_i are the HIP coordinates in x, y, and z, and IB_i is either the minimum or maximum coordinates of the inner box in x,y,z depending on the location of the HIP, i.e. i=x,y,z. If HIP_i is less than or equal to the minimum i coordinate of the inner box IB_i^{min} then IB_i equals IB_i^{min}, and when it is greater than or equal to the maximum i coordinate of the inner box IB_i^{max} then IB_i equals IB_i^{max}. The actual displacement applied to the virtual object is given by the product of **r** with the scalar s_v, which scales device/VHW units into virtual world units and depends on the size of the molecules simulated, i.e. the larger the molecule the larger the s_v value becomes. It is possible for the HIP to move without updating the position of the cursor. This occurs when the molecules collide with each other (Figure 3.5f). Under molecular collision, the HIP will update the cursor ONLY if the given displacement moves the cursor/object to a valid (i.e. collision free) position (see Section 6.3). As such the method decouples virtual object movement from HIP movement completely, unlike the virtual coupling approach which connects (constrains) the HIP and virtual object with a spring [BJ08]. This decoupling is the main advantage of this method since it allows unconstrained object/VHW movement(i.e. unconstrained by the spatial resolution of the actual haptic workspace) within the visual world, and enables the efficient handling of intermolecular collisions during a docking simulation. By keeping the receptor fixed in space and moving only the ligand, the method differentiates itself (in addition to the VHW) from those of Subaci and Basdogan [SB08], and Stocks et.al [SHL09], both of which apply rate and position control displacements to the receptor and ligand molecules respectively. Since it feels natural to place a key into a steady lock rather than a movable one, the author finds this type of haptic navigation to be more intuitive than the previous ones for molecular docking. Nonetheless, Haptimol_RD has also implemented the other mode of receptor/ligand movement, for the user accustomed to that type of navigation.

3.7 Conclusion

This chapter describes the initial stages of development of an interactive haptics-assisted docking application called Haptimol_RD, including references to the force field and molecular-structure-defining format used. The focus during this stage was given on designing and developing the application's molecular visualization and haptic navigation routines. Emphasis was also given on the size of molecules supported, since the docking of very large biomolecules was one of the design goals for Haptimol_RD. For that purpose, three molecular rendering techniques were examined with increasing levels of design and implementation complexity. The first technique rendered structures using OpenGL, the second one using OpenGL and GLSL (OpenGL/GLSL), and the third one using GLSL. Based on performance measurements, the GLSL approach outperformed the other two significantly, achieving real-time frame rates for

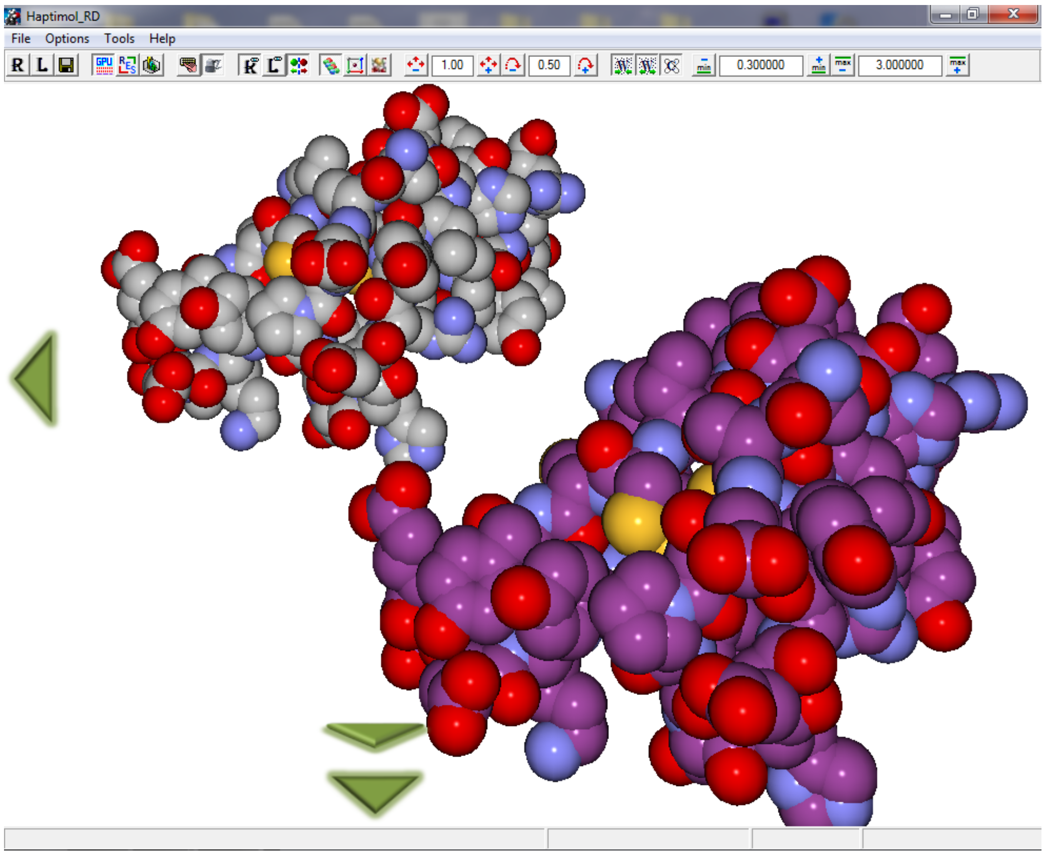


Figure 3.6: Haptic navigation of the ligand (molecule in purple) around the receptor using Haptimol_RD. The green arrows indicate that the ligand (and the VHW) moves along the negative x, y and positive z axes under rate control displacements. The arrows are not displayed when the ligand moves under position control displacements.

molecules comprising up to 190k atoms each, with the OpenGL approach performing the worst. Nonetheless, execution conflicts (discovered later during this thesis) between the GLSL method and the force calculation method discussed in Chapter 5 prohibited us from applying this rendering technique in Haptimol_RD. For this reason, Haptimol_RD implements the OpenGL/GLSL method which can render in real-time molecules comprising up to 40k atoms each.

In addition to the molecular rendering techniques, a novel haptic navigation method is also presented here. The method utilizes two haptic workspaces (i.e. device and virtual) in order to decouple HIP movement from virtual object movement. This decoupling enables the unconstrained haptic navigation of large virtual environments, and facilitates the execution of the multipoint collision response method discussed in Chapter 6. With the molecular visualization and haptic navigation routines implemented, the development effort can now be steered towards the design and implementation of Haptimol_RD's force calculation routine. The next two chapters discuss the work done towards this direction, and describe novel methods for computing the interaction forces on the CPU and GPU. The discussion begins, in the next chapter, with the description of an efficient CPU-based force calculation approach.

Chapter 4

Real time calculation of the docking forces on the CPU

4.1 Introduction

A fundamental part of haptics-assisted interactive docking is the calculation of the interaction forces, at haptic refresh rates. As mentioned in Section 2.2.3, most of the existing interactive CPU-based docking applications utilize pre-computed force grids in order to satisfy this requirement. Such grids, however, have high memory requirements, induce rough force transitions at cell boundaries [WMJ07], and, by design, cannot accommodate receptor flexibility since the grids must be computed at haptic refresh rates after each structural deformation. Furthermore, none of the existing CPU-based force calculation approaches can attain force updates within 2ms for large molecules (comprising several thousands of atoms each). Hence, a real-time force calculation approach that can address these issues effectively still remains elusive.

This chapter describes the steps taken towards the design and development of a force calculation approach capable of computing (on the CPU) in real time, and at haptic refresh rates the intermolecular forces of docking. The final result is a novel approach that addresses efficiently and successfully all issues discussed earlier and can facilitate the haptics-assisted docking of large molecular structures. The discussion starts with a brief description of the brute force approach in the next section.

4.2 CPU-based Brute Force approach

An intuitive method to compute the total interaction force is the brute force approach. Using Equation 2.2.8, this approach accounts for all interatomic interactions (in real-time) between the receptor and the ligand. The method does not require any pre-computations (e.g. precomputed force-grids), but has a time complexity of O(NM), where N and M are the number of atoms in the receptor and ligand respectively. Although it can, in principle, facilitate molecular flexibility (since it has no pre-computation requirements), it is computationally very demanding especially when applied to large structures. This approach has been implemented by Nagata et. al. [NMT02] and Sourina et. al. [STW09], but their docking simulations were constrained to molecule sizes no larger than a couple of hundred of atoms each. The main reason for this was the lack of the necessary CPU processing power capable of supporting brute force calculations for larger structures.

The brute force approach is revisited in this thesis in order to examine whether or not current CPU architectures command the necessary processing power. The following two subsections describe our implementation and results respectively.

4.2.1 Computing the force

Algorithm 1 outlines the brute force approach used in this study. The approach traverses sequentially, in the first loop, each receptor atom a_R, and accumulates in the total force variable f_{Tot} the pairwise interaction forces between a_R and each ligand atom a_L traversed in the second loop. To compute the force the method updates the position of each ligand atom, calculates its pairwise atom distance with a_R, and then applies Equation 2.2.8 (i.e. *computeForce*) for each (a_R, a_L) pair. Position updates of the ligand atoms were done using the combined viewing transformation matrix $\boldsymbol{T_{New}}$ discussed in Section 4.3.4.

Algorithm 1 Brute Force

```
Require: Receptor, array of atom structures
Require: Ligand, array of atom structures
Require: $T_{New}$, combined viewing transformation matrix
Ensure: $f_{Tot}$, total interaction force
1: for all atoms $a_R$ in Receptor do
2:    for all atoms $a_L$ in Ligand do
3:       // adjust atom coordinate
4:       $a_L.centre_{xyz} \leftarrow a_L.centre_{xyz} * T_{New}$
5:       d $\leftarrow$ distance($a_R$, $a_L$)
6:       $f_{Tot} \leftarrow f_{Tot}$ + computeForce($a_R$, $a_L$, d)
7:    end for
8: end for
9: end
```

4.2.2 Performance

Performance testing for the brute force method was trivial. Each experiment utilized two artificial, one-dimensional arrays of atoms (one for the receptor the other for the ligand) of the same size. During each experiment, the method was executed ten times (to average possible scheduling delays caused by background processes) using the same data sets, and the response time of each trial was recorded. If the response times, on average, were faster than the 2ms threshold then the size of both arrays was increased by ten atoms, and the experiment was repeated. The testing was concluded when the method could not satisfy the less than 2ms condition. For these tests, the trials started with an array size of two hundred atoms per molecule, and were concluded when the arrays reached two hundred and sixty atoms each. Therefore, the method was able to achieve force updates, at haptic refresh rates, for molecules up to two hundred and fifty (**250**) atoms each (i.e. the array size in the last valid set of trials). This result reaffirms that the brute force approach although intuitive and easy to implement, 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. All of the experiments were conducted on a 2.93GHz Intel Core i7 PC, running a 64bit version of Windows with 8GB RAM. The use of artificial atoms (i.e. atom arrays) is justified by the fact that the method's performance depends only on the size of the interacting molecules, and not on the atom values used in Equation 2.2.8.

4.3 Using a cut-off distance

As seen earlier (see Sections 2.2.3 and 4.2), it is infeasible to compute in real time the total interaction force for large molecules, based on all interatomic interactions due to the O(NM) complexity of Equation 2.2.8. Researchers have studied this issue and proposed a set-reduction technique that can accelerate significantly all force computations, while providing an acceptable approximation of the total interaction force. The technique reduces the number of interatomic interactions accounted for in Equation 2.2.8 using a cut-off distance as an interaction threshold (Figure 4.1). Specifically, the technique identifies the set of receptor/ligand atom pairs within a given cut-off, and then applies Equation 2.2.8 only on this set (all remaining atom pairs are discarded from the calculation). The main idea behind this method is that as the interatomic distance passes a certain limit (cut-off) the denominators of the VDW and electrostatic interactions in Equation 2.2.8 become very large forcing their terms to converge to zero. As such, all interatomic interactions beyond a certain

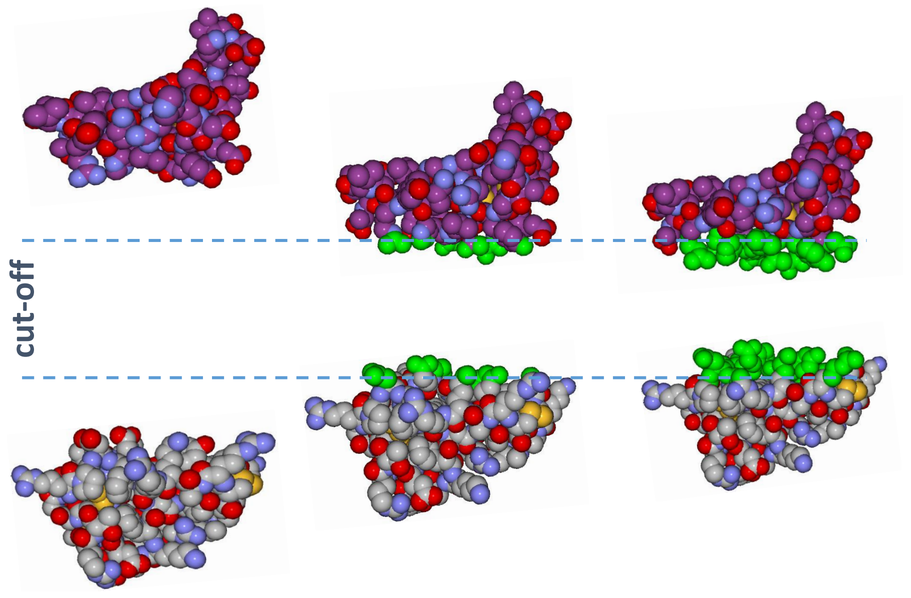


Figure 4.1: Representing visually the concept of a cut-off distance. As the receptor and ligand (in purple) molecules come in close proximity, the pairwise interatomic distance in some of their atoms becomes less than or equal to the cut-off (atoms coloured in green). Force calculation will be based only on this set of atom pairs.

distance contribute to the total force infinitesimally (close to zero), and therefore can be discarded. Cut-off distances have been used extensively in MD [SPF+07] and automated docking simulations [MHL+09] in order to accelerate force calculations, with distances between 8-12Å [SPF+07, AGB13] being the most common ones used.

The work described in thesis examines whether or not this concept can help a haptics-docking application achieve real-time force updates, at haptic refresh rates, for large molecules. The following subsections and Chapter 5 describe the work done towards this direction.

4.3.1 Force calculations on the CPU using proximity querying

Because of the cut-off distance, the calculation of the total interaction force can be viewed alternatively as a proximity querying problem, rather than a simple accumulation problem. As such, the force calculation problem can be stated anew as,

> *Given a receptor and a ligand molecule with N and M atoms respectively and a cut-off distance, identify the subset of K and L atoms (where $K \subseteq N$ and $L \subseteq M$) with pairwise interatomic distances less than or equal to the cut-off distance. Compute the total force using the sets K and L.*

A crucial aspect, therefore, in a cut-off-based force calculation approach is to identify the set of the interacting atoms. If this identification can be done at haptic refresh rates (and for large molecules) then a real-time force calculation approach can be possible. The research presented in this thesis investigated this problem and addressed it using proximity querying techniques on two spatial subdivision structures; namely, regular grids and octrees [IHL14]. Regular grids/octrees are spatial partitioning structures that divide the geometry of an object into smaller subunits. Regular grids subdivide uniformly the geometry's tightest bounding volume into grid cells (uniform subunits), whereas octrees divide recursively the object's geometry into octants (non-uniform subunits) until they reach a certain subdivision depth. Although trivial to construct, regular grids are memory inefficient, since they cannot discard grid cells empty of geometry. On the other hand, octrees do discard empty octants (very memory efficient) but there are more elaborate to construct. Specifically, octree construction starts by subdividing the object's bounding volume into eight child octants of the same size, and by continuing this subdivision recursively L times, where L is the subdivision depth, for all child octants still containing parts of the geometry. This recursive subdivision results in a tree structure of maximum degree 8 and depth L that represents the object's geometry, and in which each leaf octant contains part of that geometry (Figure 4.3). Both spatial partitioning structures simplify the implementation, and accelerate the execution of costly operations[CH88] such as object intersection discovery, neighbour finding, proximity querying etc. With regular grids, these operations are often implemented as searches over a range of cells, whereas with octrees they are often implemented as simple, recursive tree traversals of the underlying structures. Accessing a grid cell is a constant time operation, whereas, accessing a leaf octant is a logarithmic operation on the height of the octree. Nonetheless, the use of regular grids is often avoided in large scale problems due to their inefficient memory requirements.

This thesis, however, examines both structures, and constructs two real-time force calculation methods using regular grids and octrees. Both methods use their respective partitioning structures in order to identify quickly the set of interatomic interactions within a given cut-off distance d_{cutoff}. The grid-based method identifies this set, by decomposing the tertiary (3D) structure of the largest molecule (usually the receptor) within a grid, and having each atom a_j from the second molecule (i.e. the ligand) query that grid. During grid querying, each a_j creates a search range of cells based on d_{cutoff}, and checks whether or not the atoms a_i stored within these cells are within d_{cutoff} (Figure 4.6a). On the contrary, the octree-based method constructs, stores and decomposes within different octrees the 3D structure for both receptor and ligand molecules, and then uses these tree structures to efficiently query the 3D

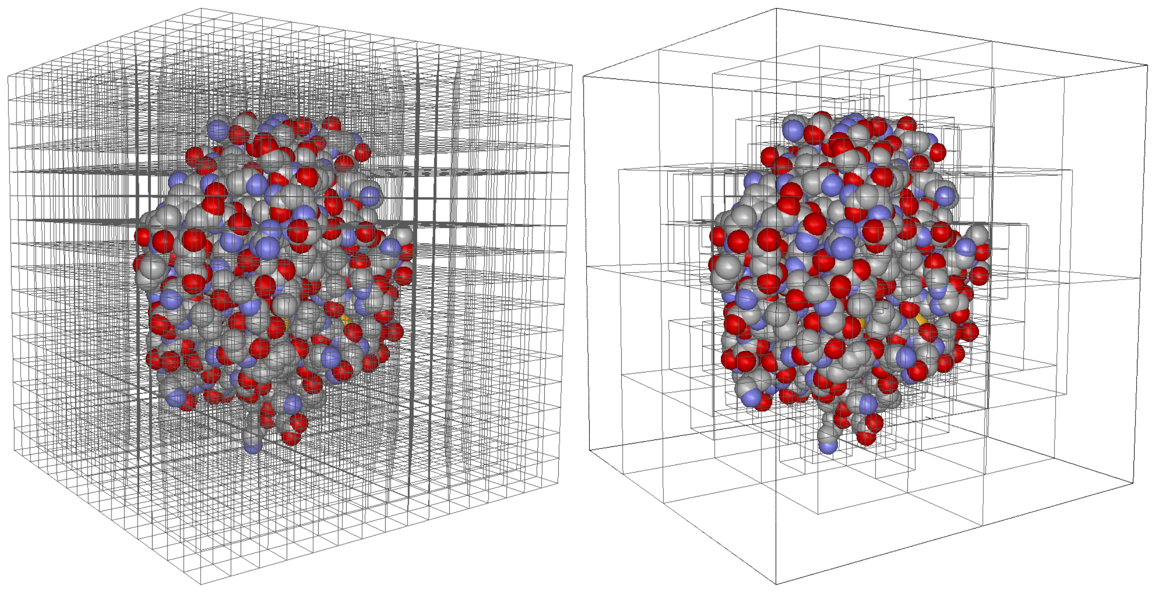

(a) (b)

Figure 4.2: The molecule Trypsin subdivided with the same level of detail by a regular grid and an octree. (a) The constructed octree structure with all of its octants displayed. (b) The constructed regular grid structure with all of its cells displayed. The total number of octants is far less than the total number of cells, resulting in a smaller memory footprint for the octree.

space and identify all atom pairs a_i and a_j whose r_{ij} distance is within d_{cutoff} (Figure 4.6b). Regular grid traversals (during construction and querying) are implemented as a nested loop, whereas the respective octree traversals are performed recursively in a depth-first order starting from the root. Both approaches calculate r_{ij} based on the center coordinates of atoms a_i and a_j. Lastly, both approaches calculate the total interaction force, in real time based on the resultant set. The next four subsections describe the CPU-based regular grid/octree construction and querying techniques examined in this thesis.

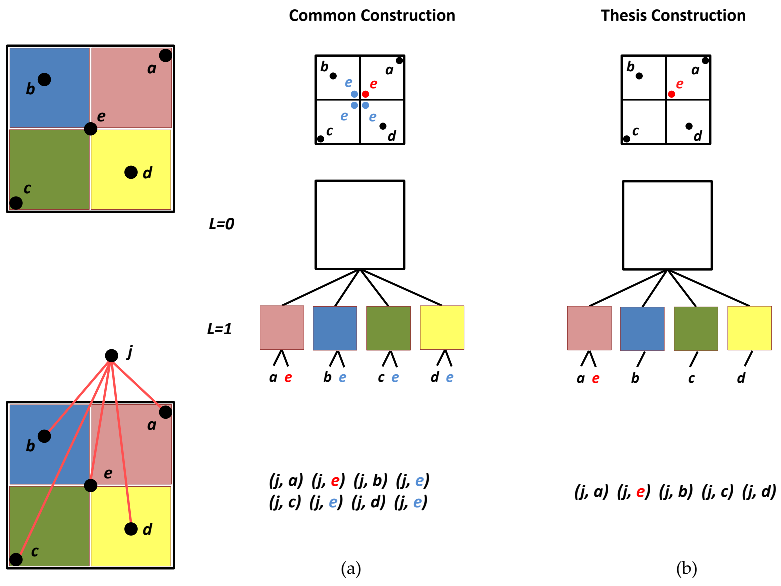

Common Construction
Thesis Construction
a
b
c
d
e
j
L=0
L=1
a e
b e
c e
d e
a e
b
c
d
(j, a) (j, e) (j, b) (j, e)
(j, c) (j, e) (j, d) (j, e)
(j, a) (j, e) (j, b) (j, c) (j, d)
(a)
(b)

Figure 4.3: Top row: A conceptual 2D visualization of a molecule R comprising of atoms a, b, c, d, e, and the respective regular grid and octree (of depth 1) structures constructed. Atom e intersects all of the cells/octants in both structures. Bottom row: R is intersected by a molecule L comprising of atom j. Atom j is within cut-off distance from (interacts with) atoms a, b, c, d, and e, forming the respective sets of interacting atom pairs. (a) A typical construction method assigns e to all cells/leaf octants, i.e. all e coloured in blue. As a result, the set of interacting atom pairs produced contains duplicates of (j,e). (b) The construction method described here assigns e to the first cell/leaf octant traversed, i.e. e coloured in red. Under this construction method the set contains only one instance of (j,e).

4.3.2 Regular grid Construction

The regular grid is constructed similarly to the approach taken by Fang and Piegl [FP93]. The method subdivides uniformly the geometry's tightest bounding box (and the geometry) into grid cells of size equal to a target size c_g. The result is a 1D array of grid cells each one of which contains either a list of atom pointers or an empty list of atoms (Figure 4.2a). Algorithm 2, ***CPUConstructRegularGrid***, outlines these construction steps,

Algorithm 2 CPUConstructRegularGrid

```
Require: $xyz_{min}$, the object's minimum bounding box coordinates
Require: $xyz_{max}$, the object's maximum bounding box coordinates
Require: $atomList$, array of atom structures
Require: $c_g$, the desired size of a grid cell side
Ensure: $G$, the regular grid as a 1D array of grid cells
 1: $\ell_{xyz} \leftarrow xyz_{max} - xyz_{min}$
 2: $n_x \leftarrow \text{floor}(\ell_x / c_g)$
 3: $n_y \leftarrow \text{floor}(\ell_y / c_g)$
 4: $n_z \leftarrow \text{floor}(\ell_z / c_g)$
 5: // calculate the actual grid cell size in x,y,z
 6: $c^{act}_{xyz} \leftarrow \ell_{xyz} / n_{xyz}$
 7: for all atoms $a_i$ in atomList do
 8:     // calculate the 3D grid cell indices for atom $a_i$
 9:     $xyz^G_i \leftarrow \text{floor}((a_i.centre_{xyz} - xyz_{min}) / c^{act}_{xyz})$
10:     // compute the 1D grid cell index for the 3D indices
11:     $c^{xyz}_i \leftarrow z^G_i n_x n_y + y^G_i n_x + x^G_i$
12:     $\text{G}[c^{xyz}_i] \leftarrow \text{pointer}(a_i)$
13: end for
14: end
```

According to Algorithm 2, the method starts by obtaining the number of grid cells in x,y,z, using Equation 4.3.1

$$n_x = \left\lfloor \frac{\ell_x}{c_g} \right\rfloor, n_y = \left\lfloor \frac{\ell_y}{c_g} \right\rfloor, n_z = \left\lfloor \frac{\ell_z}{c_g} \right\rfloor \tag{4.3.1}$$

where n_x, n_y and n_z are the number of cells in x,y and z directions, and 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. It then adjusts the cell size per each dimension (i.e. $c_x^{act}, c_y^{act}, c_z^{act}$) by dividing the bounding box's dimensions with the resultant number of grid cells (i.e. n_x, n_y, n_z). To assign atoms into grid cells, the method transforms initially the centre coordinates of each atom into a 3D index using Equation 4.3.2,

$$x_i^G = \left\lfloor \frac{x_i^a - x_{min}}{c_x^{act}} \right\rfloor, y_i^G = \left\lfloor \frac{y_i^a - y_{min}}{c_y^{act}} \right\rfloor, z_i^G = \left\lfloor \frac{z_i^a - z_{min}}{c_z^{act}} \right\rfloor \tag{4.3.2}$$

where x_i^G, y_i^G and z_i^G form the 3D cell index for atom i, x_i^a, y_i^a and z_i^a are the atom's coordinates, and x_{min}, y_{min} and z_{min} are the minimum coordinates of the bounding volume. It then maps the 3D index into an 1D index, and assigns the atom to the cell located at this index position, within the regular grid (i.e. an 1D array of cells). For the 3D to 1D cell index mapping, the method uses Equation 4.3.3,

$$c_i^{xyz} = z_i^G n_x n_y + y_i^G n_x + x_i^G \tag{4.3.3}$$

where c_i^{xyz} is the 1D mapping of the 3D cell index $[x_i^G, y_i^G, z_i^G]$ for atom i. This construction method avoids the insertion of an atom into multiple cells (Figure 4.3). This is the case when the atom's center intersects multiple cells (e.g. lies on cell borders). In such cases it is customary to insert the given atom to all intersected cells in order to ensure its proper traversal during cell querying (especially during collision detection queries). The construction described here does not favour multiple atom insertions, and relies on the query algorithm (see Section 4.3.5) for ensuring that all such atoms will be considered for the final query set, irrespective of which cells they were assigned to. This one-to-one relationship between atoms and cells accelerates the performance of the query algorithm because each interatomic pair is considered only once, and thus the cost of handling duplicate pairs is avoided (Figure 4.3).

4.3.3 Octree Construction

The octree-based method constructs the tree structure (as a linked-list of linked-lists of octants), and populates it with atoms simultaneously. It requires as input a target subdivision level L, and the dimensions of the geometry's bounding box. Initially, the method turns the bounding box into a bounding cube (with sides of size equal to max(ℓ_x, ℓ_y, ℓ_z)), and sets this cube as the tree's root octant. It then utilizes Algorithm 3, ***ConstructOctantLevel***, to subdivide the tree into equally-sized child/leaf octants, and to populate them with atoms (Figure 4.2b).

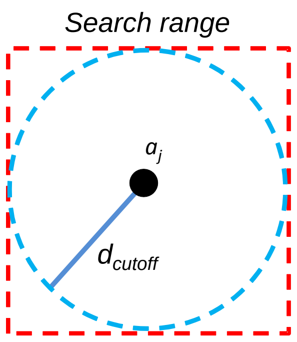
Given an atom, Algorithm 3 traverses the octree, and assigns the atom to the leaf octant that either intersects or contains its centre. If the path to this leaf octant (or the leaf itself) does not exist, the method creates the required octants (forming this path) dynamically, and subdivides the tree structure 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. Since an octant is created only when an atom intersects it, the resultant octree contains no empty octants (octants with no atoms), and thus the structure is compact and memory efficient, and helps the query algorithm avoid unnecessary octant traversals. Similar to the grid method, when the atom is intersected by more than one octant the construction algorithm assigns the atom to the first leaf octant traversed, and not to all of them (Figure 4.3). Again, the query algorithm (see Section 4.3.6) ensures that the given atom will be considered for the final query set, and thus avoids the cost of handling duplicate pairs (Figure 4.3). The octants have to be of uniform size in order to facilitate the proper execution of the querying algorithm, and minimize the number of false positives traversed. All octrees constructed in this chapter are of depth equal to four, because of the performance balances attained between octree construction and querying times (see Section 4.4.1).

Algorithm 3 ConstructOctantLevel

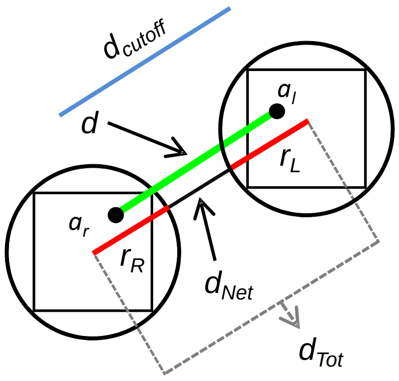
```
Require: $L$, the octree's targeted level
Require: $L_{curr}$, the octant's current level in the octree
Require: $xyz^{box}_{min}$, the octant's minimum box coordinate
Require: $xyz^{box}_{max}$, the octant's maximum box coordinate
Require: $a_i$, the atom object to add
Ensure: $octantsChildrenList$, a linked-lists of octants
1: if isOctantNew = true then
2:   $oct.xyz^{box}_{min} \leftarrow xyz^{box}_{min}$
3:   $oct.xyz^{box}_{max} \leftarrow xyz^{box}_{max}$
4:   $oct.xyz_{centre} \leftarrow (xyz^{box}_{min} + xyz^{box}_{max})*0.5$
5:   $oct.radius \leftarrow$ distance$(xyz^{box}_{max} - xyz^{box}_{min})*0.5$
6:   isOctantNew ← false
7: end if
8: $L_{child} \leftarrow L_{curr}+1$
9: // reached a tree leaf node, thus assign to this octant the given atom
10: if $L_{child} > L$ then
11:   octantsAtomList.AddEnd(pointer($a_i$))
12:   isOctantNew ← true
13: else
14:   for all 8 child octants $oct^{Ch}_i$ in $oct$ do
15:     // compute and assign the cube coordinates for each child octant
16:     $oct^{Ch}_i.xyz^{box}_{min} \leftarrow$ min cube coordinates for $oct^{Ch}_i$
17:     $oct^{Ch}_i.xyz^{box}_{max} \leftarrow$ max cube coordinates for $oct^{Ch}_i$
18:     if $a_i.centre_{xyz}$ within $[oct^{Ch}_i.xyz^{box}_{min}, oct^{Ch}_i.xyz^{box}_{max}]$ then
19:       if $oct^{Ch}_i$ not created then
20:         $oct^{Ch}_i \leftarrow$ new Octant Node
21:         isOctantALeaf ← false
22:         // add this child at the end of the octant's children list
23:         octantsChildrenList.AddEnd($oct^{Ch}_i$)
24:       end if
25:       // forward/add the atom to $oct^{Ch}_i$ octants recursively
26:       $oct^{Ch}_i$.ConstructOctantLevel($L$, $L_{child}$, $oct^{Ch}_i.xyz^{box}_{min}$, $oct^{Ch}_i.xyz^{box}_{max}$, $a_i$)
27:     end if
28:   end for
29: end if
30: end
```

4.3.4 Updating atom coordinates during querying

As the two molecules move in space, their geometry changes position and orientation. These changes are stored in the two viewing transformation matrices $\boldsymbol{T_R}$ and $\boldsymbol{T_L}$, for the receptor and ligand molecules respectively. In order to function correctly, the querying algorithm must apply these matrices to all structures involved, i.e. partitioning structures and atom coordinates. To save a substantial amount of matrix-vector multiplications, both querying methods combine $\boldsymbol{T_R}$ and $\boldsymbol{T_L}$ into one matrix using the relation $\boldsymbol{T_{New}} = \boldsymbol{T_R^{-1}T_L}$, and then apply $\boldsymbol{T_{New}}$ only to the ligand-related structures (they use $\boldsymbol{T_{New}} = \boldsymbol{T_L^{-1}T_R}$ and apply $\boldsymbol{T_{New}}$ to the receptor-related structures if the ligand is larger than the receptor). This optimization technique maintains the relative orientation of both molecules intact, and instead of transforming both molecules (and their partitioning structure(s)) in space, it keeps the receptor geometry fixed and transforms only the ligand geometry with respect to it (or vice versa).



(a)



(b)

Figure 4.4: A 2D visualization of Algorithm 4 and Equation 4.3.4. (a) During regular grid querying each atom a_j creates a search region, mapped to grid cells, based on d_{cutoff} (red dotted box). This region, is always larger than the one required (blue dotted circle), since the latter is always inscribed in the former. (b) Two leaf octants that belong to different octrees and contain atoms a_r and a_l, respectively. These atoms are within the cut-off distance and thus interact. In cases like this, 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. In this example d_{Tot} is larger than d_{cutoff}.

4.3.5 Regular grid Querying on the CPU

Algorithm 4 GetSearchRange

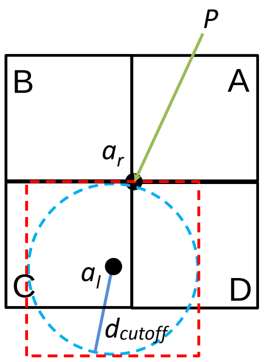
```
Require: $RG_{Info}$, regular grid query info structure
Require: $centre_{xyz}$, atom coordinates
Require: $d_{cutoff}$, the cut-off distance
Ensure: $x^G$ $y^G$ $z^G$, returned min-max range structures for x,y,z
 1: // get the min-max coordinates of the cube
 2: // centred within $d_{cutoff}$ from atom $a$
 3: $x^C_{min} \leftarrow centre_x$-$d_{cutoff}$
 4: $y^C_{min} \leftarrow centre_y$-$d_{cutoff}$
 5: $z^C_{min} \leftarrow centre_z$-$d_{cutoff}$
 6: $x^C_{max} \leftarrow centre_x$+$d_{cutoff}$
 7: $y^C_{max} \leftarrow centre_y$+$d_{cutoff}$
 8: $z^C_{max} \leftarrow centre_z$+$d_{cutoff}$
 9: // get the search range in x,y,z coordinates
10: $x^G.min \leftarrow (x^C_{min}\text{-}RG_{Info}.x_{min})/RG_{Info}.\ell_x$
11: $y^G.min \leftarrow (y^C_{min}\text{-}RG_{Info}.y_{min})/RG_{Info}.\ell_y$
12: $z^G.min \leftarrow (z^C_{min}\text{-}RG_{Info}.z_{min})/RG_{Info}.\ell_z$
13: $x^G.max \leftarrow (x^C_{max}\text{-}RG_{Info}.x_{min})/RG_{Info}.\ell_x$
14: $y^G.max \leftarrow (y^C_{max}\text{-}RG_{Info}.y_{min})/RG_{Info}.\ell_y$
15: $z^G.max \leftarrow (z^C_{max}\text{-}RG_{Info}.z_{min})/RG_{Info}.\ell_z$
16: end
```

Regular grid querying is implemented as a series of nested loops. The first loop accesses sequentially the atoms a_j of the smallest molecule. For each a_j visited, the method updates the atom's coordinates with $\boldsymbol{T_{New}}$, and then uses Algorithm 4, ***GetSearchRange***, to identify a search region of grid cells (Figure 4.4a). Algorithm 4 computes the tightest bounding cube of a sphere with centre equal to the updated coordinates of atom a_j, 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. The inner loops traverse these x,y,z ranges, and each x,y,z loop step is mapped into an 1D index using Equation 4.3.3 (where the x^G_i, y^G_i, and z^G_i terms take the respective x,y,z loop-step values). Using this 1D index (if valid), the method accesses the respective cell and checks whether or not the cell contains atoms a_i that lie within d_{cutoff} distance from atom a_j. All atom pairs (a_i, a_j) found within the d_{cutoff} are stored inside set S_{Pairs}. The total force is then calculated based on this set S_{Pairs} (Section 4.3.7).

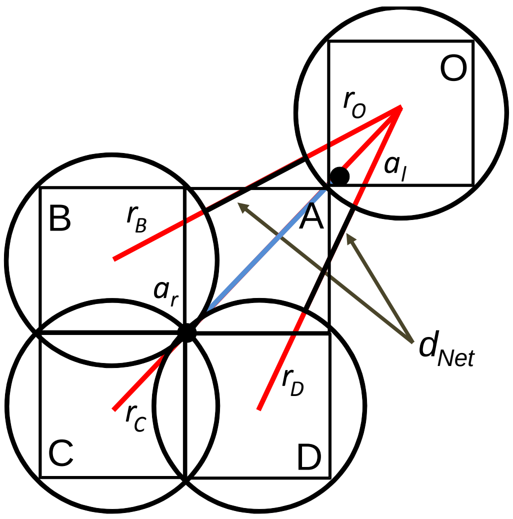
Algorithm 5 CPUQueryRegularGrid

```
Require: $atomList$, array of atom structures
Require: $G$, the regular grid as a 1D array of grid cells
Require: $T_{New}$, combined viewing transformation matrix
Require: $RG_{Info}$, regular grid query info structure
Require: $d_{cutoff}$, the cut-off distance
Ensure: $S_{Pairs}$, set of interacting atom pairs
1: for all atoms $a_i$ in atomList do
2:   // adjust atom coordinate
3:   $a_i.centre_{xyz} \leftarrow a_i.centre_{xyz} * T_{New}$
4:   // execute Algorithm 1
5:   $GetSearchRange(RG_{Info}, a_i.centre_{xyz}, d_{cutoff}, x^G, y^G, z^G)$
6:   for l=$x^G$.min to l≤ $x^G$.max with l++ loop do
7:     for k=$y^G$.min to k≤ $y^G$.max with k++ loop do
8:       for j=$z^G$.min to j≤ $z^G$.max with j++ loop do
9:         indx ← l*$RG_{Info}.n_x$*$RG_{Info}.n_y$+k*$RG_{Info}.n_x$+j
10:        gridCell ← G[indx]
11:        if gridCell not empty then
12:          for all atoms $a_i^G$ in gridCell do
13:            d ← distance($a_i$, $a_i^G$)
14:            if d ≤ $d_{cutoff}$ then
15:              $S_{Pairs}.add(a_i, a_i^G)$
16:            end if
17:          end for
18:        end if
19:      end for
20:    end for
21:  end for
22: end for
23: end
```

Algorithm 4 allows each ligand atom to concentrate its search to a small section of the regular grid structure. Because of it, the underlying rejection/acceptance tests are computationally inexpensive since they use a constant time operation in order to traverse the candidate cells. Additional computational-cost savings result from the fact that the method does not have to handle duplicate atom pairs. As stated in Section 4.3.2, the grid-construction method does not allow insertions of the same atom to different cells. Therefore receptor atoms intersecting multiple cells are inserted in only one of these cells. To ensure the proper handling of such special cases, the querying method must be cell indifferent, meaning that it should query the atoms regardless of which cells they were placed in. The method described here can accommodate such special cases because the search area defined for each ligand atom is larger than the one required, i.e. a sphere inscribed in a cube. Specifically, given a ligand atom a_l and a receptor atom a_r, a_l must query a_r only if the distance from their centres is within d_{cutoff}. But if it is within d_{cutoff}, that means that a_r lies either on, or is inside in the search area defined by the bounding cube. Since the bounding cube and atom a_r share the same coordinates, Algorithm 4 will incorporate the respective cell into the range of cells returned back to the querying algorithm (Figure 4.5a). As such, a_r will be queried as expected, regardless of its placement within the regular grid during construction. Hence, the query method will function correctly under all construction cases, special or trivial. Apparently, the larger search area means that a ligand atom might have to perform more distance-based inclusion/exclusion tests with receptor atoms than necessary. This is a common issue in grid-based querying (popular approaches such as [SPF$^+$07] have it), and is usually addressed by making the size of the cells smaller, i.e. increased grid subdivision. Namely, regular grids with a finer partition granularity will reduce this cost (less receptor atoms to test per cell), and those with a coarser partition granularity will increase it. This logic has been tested in this thesis during benchmarking (see Section 4.4.1).



(a)



(b)

Figure 4.5: A diagram-based proof that both querying algorithms will handle the atom pair (a_r, a_l) correctly, irrespective of the cell/leaf octant (A, B, C or D) atom a_r was initially assigned to during construction. a) Atom a_l maps its search range (red dotted cube) on the regular grid. Even though a_l is not within d_{cutoff} from a_r, the query will examine a_r (indifferent of cell placement) since the range and a_r share the same coordinates at position P (in this case the search range encloses all four cells). When the atoms are within d_{cutoff}, a_l's range will always enclose/overlap with a_r and therefore such a P will always exist. b) Octant O queries octants A, B, C, and D to identify whether or not a_r and a_l are within d_{cutoff}. Even in the worst case scenario (a_r was assigned to C), the d_{Net} of octants C and O will be equal to d_{cutoff} (blue line), whereas in the other three cases it will be less than d_{cutoff}. In this example octant A is the best insertion case for a_r since it is in zero d_{Net} distance from octant O.

4.3.6 Octree Querying on the CPU

Octree querying is implemented as a series of tree traversals. Given $\boldsymbol{T_{New}}$ and the two octree structures the method applies Algorithm 6, ***CPUQueryOctant***, 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 computes then the total force (Section 4.3.7). 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 $\boldsymbol{T_{New}}$, and then computes the net distance d_{Net} between the octant centres using:

$$d_{Net} = d_{Tot} - (r_R + r_L) \tag{4.3.4}$$

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 4.4b). If d_{Net} is less than or equal to the cut-off distance, 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 4.3.4, the cut-off distance, and the octree hierarchy, the query strategy performs quick rejection tests on the underlying molecular geometry, and converges rapidly to those leaf-octants containing the interacting atom pairs. The octant rejection test (i.e. $d_{Net} > d_{cutoff}$) 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 4.3.3 (i.e. when atoms are intersected by more than one octant). As previously mentioned, the construction

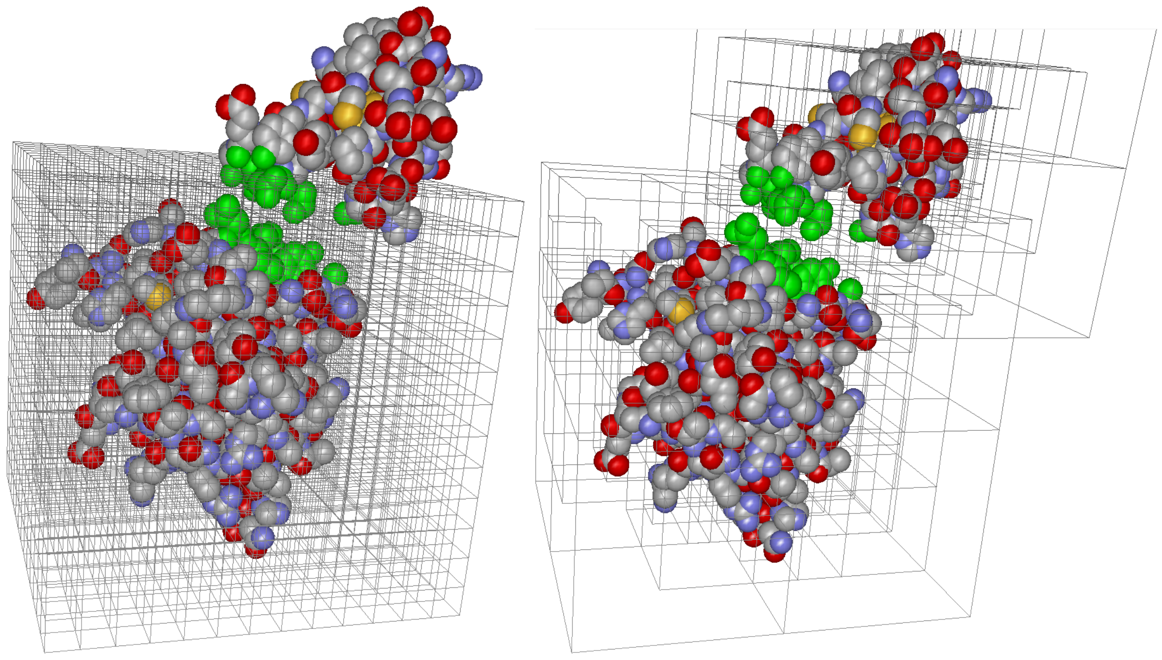
Algorithm 6 CPUQueryOctant

```
Require: $T_{New}$, combined viewing transformation matrix
Require: $oct^{First}$, an octant from the first octree structure
Require: $oct^{Secnd}$, an octant from the second octree structure
Require: $d_{cutoff}$, the cut-off distance
Ensure: $S_{Pairs}$, set of interacting atom pairs
1: if both $oct^{First}$ AND $oct^{Secnd}$ are leaf-octants then
2:   for all atoms $a_r$ in $oct^{First}$ AND all atoms $a_l$ in $oct^{Secnd}$ do
3:     d ← distance($a_r$, $a_l$)
4:     if d ≤ $d_{cutoff}$ then
5:       $S_{Pairs}.add(a_r, a_l)$
6:     end if
7:   end for
8: else
9:   if $oct^{First}$ OR $oct^{Secnd}$ is a leaf-octant then
10:    // set non-leaf octant to tmpNLOctant and leaf octant to tmpOctant
11:    if $oct^{First}$ is a leaf-octant then
12:      tmpNLOctant ← $oct^{Secnd}$
13:      tmpOctant ← $oct^{First}$
14:    else
15:      tmpNLOctant ← $oct^{First}$
16:      tmpOctant ← $oct^{Secnd}$
17:    end if
18:    for all child octants $oct_c$ in tmpNLOctant do
19:      $d_{Net}$ ← net-distance($oct_c$, tmpOctant)
20:      if $d_{Net} \leq d_{cutoff}$ then
21:        CPUQueryOctant($T_{New}$, $oct_c$, tmpOctant, $d_{cutoff}$)
22:      end if
23:    end for
24:  else
25:    for all child octants $oct_r$ in $oct^{First}$ AND all child octants $oct_l$ in $oct^{Secnd}$
   do
26:      $oct_{small} \leftarrow min(oct_r, oct_l)$ in size
27:      $oct_{small}.centre_{xyz} \leftarrow oct_{small}.centre_{xyz} * T_{New}$
28:      $d_{Net}$ ← net-distance ($oct_r$, $oct_l$)
29:      if $d_{Net} \leq d_{cutoff}$ then
30:        CPUQueryOctant($T_{New}$, $oct_r$, $oct_l$, $d_{cutoff}$)
31:      end if
32:    end for
33:  end if
34: end if
35: end
```

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 4.5b). As such, a_r will be queried and inserted in S_{Pairs} as expected, regardless of its placement within the octree during construction (similarly to grid-based querying). Again, the rejection test will prune correctly the octrees (and their underlying geometry), under all construction cases, special or trivial.

4.3.7 Calculating the Force

Upon the completion of the respective proximity querying method, the force calculation procedure traverses sequentially the S_{Pairs} set, and for each atom pair found in S_{Pairs}, it computes the VDW and electrostatic force contributions and adds them to the total force. Since all force calculations are performed in real time, both methods can facilitate independent handling of the electrostatic and VDW forces in a manner similar to the one reported in Lee and Lyons[LL04]. 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.



(a) (b)

Figure 4.6: Two proteins interact during a docking simulation. Green colours denote the atoms with a pairwise interatomic distance less than or equal to a given cut-off, as identified by the two proximity querying methods. (a) The regular grid-based method, in which all ligand atoms query the regular grid (applied only on the receptor) to identify the interacting atom pairs. (b) The octree-based method, in which the interacting atom pairs are identified by querying both receptor/ligand octrees recursively and pairwise.

4.4 Performance Testing of the CPU-based methods

A series of docking simulations were conducted in order to evaluate whether or not the two real-time force calculation approaches could satisfy the requirements set earlier. As such, both approaches had to be implemented and integrated into Haptimol_RD. Using Haptimol_RD the following set of experiments were conducted:

1. benchmarking construction and querying performances
2. measuring querying performance during real-time rigid-docking simulations

Similarly to brute force testing (Section 4.2.2), each benchmarking experiment was executed ten times, and the values reported was the average of those results. Again, all of the 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 these simulations was the Geomagic Touch (formerly known as SensAble Technologies Phantom Omni). For the purpose of benchmarking arbitrary force parameters were used, since the timing of the respective force-computations does not depend on the values of these parameters. For the haptics-assisted rigid-docking simulations, however, the force parameters were obtained using the process described in Section 3.4.

4.4.1 Benchmarking Performance on the CPU

Benchmarking experiments were conducted in order to measure the scalability of these proximity querying methods, and identify their limitations. To achieve that, both approaches were subjected to various artificial docking simulations of demanding computational workloads and different molecular complexities. At this stage emphasis was given in finding those molecules that can stress test the two proximity querying

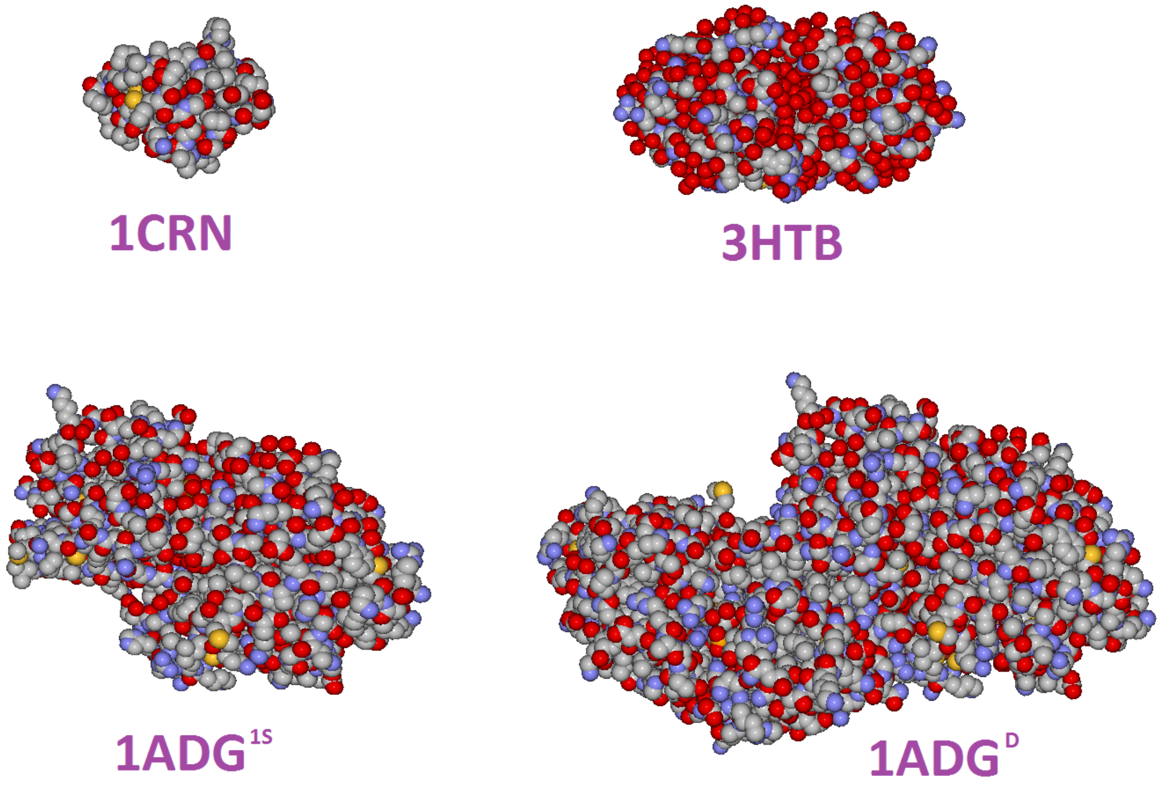
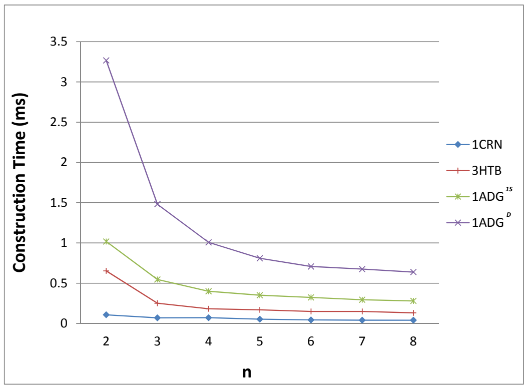


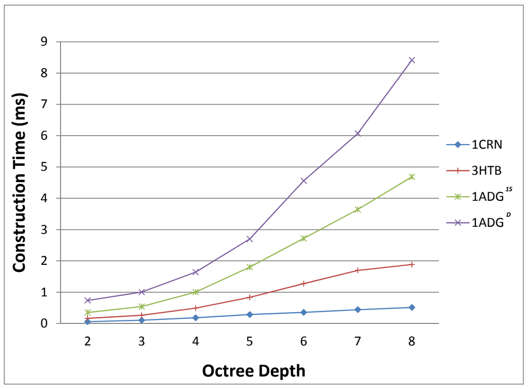
Figure 4.7: The four molecules used for benchmarking the two CPU-based force calculation approaches, while showing their relative sizes. The largest molecule is $1ADG^D$ with 7k atoms (see Table 4.1).

algorithms effectively. Since proximity queries are sensitive to the atom granularity of the underlying cells/octants, the molecules were selected with different sizes and shapes (e.g. compact, extended). Although unrealistic, the molecules were also allowed 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.

In addition to querying times, the experiments accounted for grid/octree construction times. Construction times are important when receptor flexibility is modelled, given that the respective structures 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. Both construction methods were benchmarked using the proteins *Crambin* (PDB code: 1CRN), *Lysozyme* (3HTB), *Alcohol Dehydrogenase* [containing only one of the two subunits, $1ADG^{1S}$], and *Alcohol Dehydrogenase* [containing both subunits (dimer), $1ADG^{D}$] (Figure 4.7), as defined in the PDB database [BWF+00]. Seven regular grids and octrees of different granularities were constructed for each test. Specifically, the regular grids were constructed using targeted cell sizes c_g equal to nr_C, where n=2,3,..8 and r_C is the radius of a carbon atom, i.e. 1.7Å . Likewise the octrees were constructed using subdivision levels L=2,3,..8. Table 4.1 lists these results.

As expected, the grid construction method outperformed the octree construction method, achieving millisecond/sub-millisecond grid-construction times in almost all test cases and subdivision sizes (due to its execution simplicity). The grid construction method was unable to build the grid structure within 1ms only when n was set



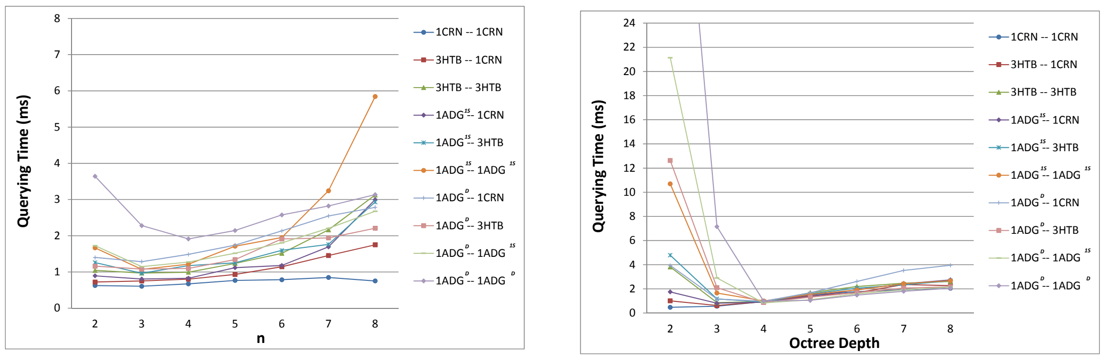


(a) (b)

Figure 4.8: Regular grid and octree construction times per molecule in milliseconds. (a) The regular grids were constructed with c_g values equal to nr_C, where n=2,..,8 and r_C is the radius of a carbon atom. (b) Similarly the octrees were constructed at depth levels L=2,3,..,8.

equal to 2 and 3 for the molecule 1ADG^D. Unlike grid construction, the octree construction method achieved 1ms responses, in all test cases, only for depths lower than four. At depth four the method supported millisecond/sub-millisecond construction times for molecules 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 4.8 depicts these construction times per protein, for all seven different n/L values. Construction times for grids and octrees with finer partition granularity (i.e. regular grids with n set equal to 1 and octrees with L set greater than or equal to 9) are not reported here, since none of them achieved better performance results during querying.

To benchmark the querying methods ten test cases were devised using the same four proteins, as shown in Figure 4.9. 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



(a) (b)

Figure 4.9: Regular grid and octree querying times of the ten interacting molecular pairs tested, in milliseconds. (a) The grid querying times at the same different cell sizes, i.e. at c_g values equal to nr_C, where n=2,..,8 and r_C is the radius of a carbon atom. (b) The octree querying times for the depths L=2,3,..,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.

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 it was the distance that generated a substantial amount of interacting atom pairs (as the ligand moved over/intersected the receptor) to test adequately the performance of both querying methods in relation to different n/L values (i.e. n,L=2,3..8) and protein sizes. 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 both querying methods. In all cases the cut-off distance was 8Å . Moreover, for each test case the values recorded were the δ_T distance used, the querying response time, the cardinality of the S_{Pairs} set, the total number of cells or child and leaf octants traversed and the total number of inter-atomic distance calculations (Tables 4.2 and 4.3). The querying response time is the time to determine the set, S_{Pairs}, the time to perform the

Table 4.1: Regular grid and octree construction times per molecule in milliseconds, for cell sizes c_g equal to nr_C (where n=3,..5 and r_C is the radius of a carbon atom) and octree depth levels 3, 4, and 5, respectively. Common to both construction methods, the table lists the name of and number of atoms comprising each molecule. It then groups under grid construction the value of n used, the total number of cells created, and the grid-construction times obtained, and under octree construction the tree level L, the total number of child/leaf octants created, and the octree-construction times obtained. *D* stands for Dimer and 1*S* for one subunit.

		REGULAR GRIDS			**OCTREES**			
Molecule	# of heavy atoms	n	Tot. # of cells	Constr. time (ms)	L	Tot. # of octants	Tot. # of leaf octants	Constr. time (ms)
		5	27	0.0542	3	132	97	0.1023
1CRN	327	4	64	0.0719	4	372	240	0.1837
		3	216	0.0698	5	699	327	0.2837
		5	216	0.1698	3	165	124	0.2617
3HTB	1388	4	343	0.1827	4	683	518	0.4879
		3	1000	0.2519	5	1810	1127	0.8329
		5	343	0.3511	3	190	143	0.539
1ADG^{1S}	3445	4	729	0.4002	4	888	698	1.0025
		3	1728	0.5468	5	3122	2234	1.8006
		5	1331	0.8101	3	137	108	1.0035
1ADG^{D}	7046	4	2744	1.0072	4	644	507	1.6403
		3	6859	1.4815	5	3135	2491	2.6987

force calculation being negligible in comparison. Figure 4.9 depicts these querying times per test case, for all seven different n/L values. According to these results, the grid-based method achieved sub-millisecond querying responses only in the first five cases and only when n$\geq$3, whereas the octree method achieved sub-millisecond responses for all test cases when the respective octree depths were set to four. Faster response times were attained for larger cell sizes and smaller octree depths in several cases, however, the measurements indicated that at n/L values equal to four both querying approaches maintained a performance balance between their construction and querying times. Evidently, the grid/octree construction and querying costs are geometry and cell-size/tree-depth dependent, and impose a trade-off between construction speed and querying performance. This relationship is depicted in Tables 4.1, 4.2, and 4.3 which list the construction and querying measurements pertinent to each method and test case, for n/L values ranging between 3 and 5 inclusive. Measurements for n/L values equal to 2, 6, 7 and 8 were not included for table clarity. Overall, octree-based querying performed better than grid-based querying, and for this reason is the default querying method used in all CPU-based force calculations.

4.4.2 Haptics-assisted Interactive Rigid-Docking Simulations on the CPU

In addition to benchmarking, several proximity querying performance tests were conducted based on rigid-docking simulations of known compounds, using the octree-based proximity querying method. Although receptor flexibility was not modelled in these simulations, the respective octree-construction times are reported for the reader who wants to take into account these construction overheads (necessary if molecular flexibility is addressed). The tests were based on three well known complexes, related to protein-protein and protein-drug docking. Namely, they utilized 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. Each one of these files contains the structures of the receptor and the ligand in their bound conformation. Out of these files, the 3D geometry of the receptor and ligand molecules was extracted and saved into a separate PDB file, i.e. two new PDB files were created from the original PDB file, one for the receptor and one for

Table 4.2: Regular grid querying times for ten interacting molecular pairs in milliseconds, for cell sizes c_g equal to nr_C, where n=3,..5 and r_C is the radius of a carbon atom. The table lists the displacement distance used in these experiments, the number of interacting atom pairs (S_{Pairs}) returned, the total number of grid cells traversed, and the total number of atom pairs examined in order to generate the set S_{Pairs}. D stands for Dimer and 1S for one subunit.

Interacting Molecules	δ_T (nm)	S_{Pairs}	L	Tot. # of cells traversed	Tot. # of atom pairs exam.	Querying time (ms)
			5	1840	39685	0.7674
1CRN–1CRN	1.55	4146	4	5751	27918	0.6702
			3	24903	20305	0.6058
			5	18250	50126	0.9324
3HTB–1CRN	2	4040	4	30369	38699	0.8002
			3	100073	28284	0.7528
			5	73428	55808	1.2364
3HTB–3HTB	2.9	3224	4	125096	41393	0.9916
			3	412640	27535	0.9696
			5	27790	64290	1.1200
1ADG^{1S}–1CRN	2.8	4792	4	64183	44595	0.8256
			3	165102	35720	0.8045
			5	114234	68268	1.2520
1ADG^{1S}–3HTB	3.88	4338	4	267998	48215	1.1753
			3	688456	35376	0.9649
			5	289674	80875	1.7106
1ADG^{1S}–1ADG^{1S}	4.29	3168	4	677707	51464	1.2149
			3	1738281	33526	1.0733
			5	98626	74067	1.7400
1ADG^{D}–1CRN	3.27	8853	4	215020	57827	1.4842
			3	560324	44108	1.2838
			5	441178	65854	1.3403
1ADG^{D}–3HTB	4.2	2952	4	958867	46459	1.0973
			3	2516965	31440	1.0811
			5	1135553	66859	1.5174
1ADG^{D}–1ADG^{1S}	5.21	2569	4	2468593	49741	1.2731
			3	6520432	25650	1.1471
			5	2336673	60754	2.1461
1ADG^{D}–1ADG^{D}	5.58	2452	4	5094195	42432	1.9123
			3	13425727	23565	2.2802

Table 4.3: 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 these 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 $1S$ for one subunit.

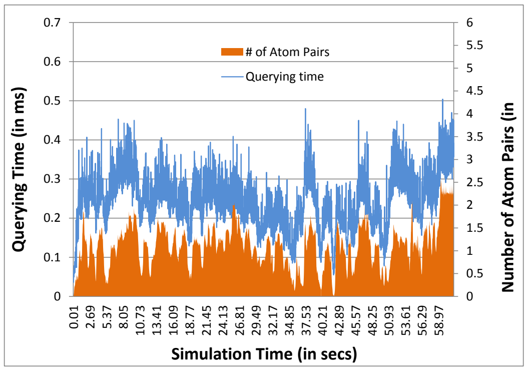
Interacting Molecules	δ_T (nm)	S_{Pairs}	L	Tot. # of octants traversed	Tot. # of leaf octants traversed	Tot. # of atom pairs exam.	Querying time (ms)
			3	2841	2450	30427	0.5666
1CRN–1CRN	1.55	4146	4	10405	7564	13792	0.9428
			5	18542	8137	8137	1.5023
			3	2076	1729	70700	0.6236
3HTB–1CRN	2	4040	4	8117	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

the ligand. As stated earlier, the pdb2gmx tool was used in order to obtain the Gromos54a7 non-bonded force parameters from each PDB file, except the file containing the drug sorafenib. For sorafenib these parameters were obtained through PRODRG server (http://davapc1.bioch.dundee.ac.uk/programs/prodrg/)[SVA04]. Using Haptimol_RD, and the respective geometry and force parameter files, three docking simulations were conducted. 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 4.10a, 4.10b, and 4.10c depict these docking simulation results, whereas Table 4.4 lists the corresponding construction results. These simulations utilized octrees of depth 4, and a cut-off distance of 8Å.

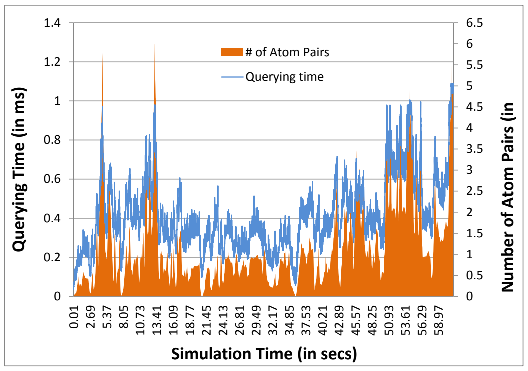
Table 4.4: Octree construction times for the six molecules used in the real-time docking simulations.

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
EGFr	5836	615	468	1.2572

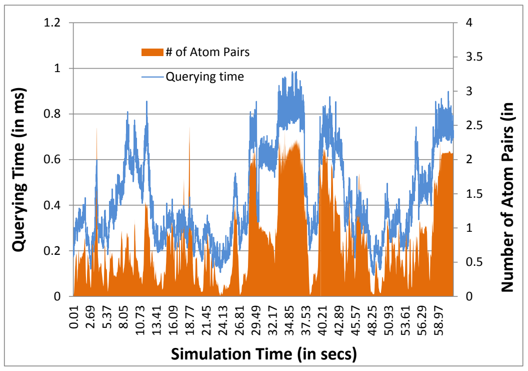
The results show that the method attained sub-millisecond response times for the majority of the simulation period. The querying times exceeded slightly the 1ms barrier only when BPTI assumed its final docking position. At that position the method performed a significant number of octant comparisons, induced by substantial



(a)



(b)



(c)

Figure 4.10: 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*. The graph depicts the querying times attained, at 10ms intervals, and the respective sets of interatomic interactions accounted for by the approach during the simulation.

octree overlapping, because BPTI was docked deep into Trypsin's binding pocket. The response times in that case fluctuated between 1.015 and 1.092 ms.

4.5 Conclusion

The main focus in this chapter is the design and development of a real-time, CPU-based force calculation approach that can accommodate the haptics-assisted docking of large molecules, and overcome the computational limitations of pre-computed force grids. As demonstrated, current CPU technology cannot facilitate real-time force calculations for such large structures, when all interatomic interactions between the receptor and ligand molecules (i.e. brute force) are accounted for. This motivated the application of a cut-off-based, set-reduction technique in order to reduce the number of interatomic interactions considered during a force calculation. Two different proximity querying methods were examined in order to identify at haptic refresh rates this reduced set of interacting atom pairs. The first method relied on a regular grid, whereas the second one relied on octrees as the means to accelerate distance queries in the 3D space. Both methods were implemented and tested using different molecular structures. Even though grid construction was significantly faster than octree construction, the octree-based querying method performed consistently better than the grid-based method in most of the test cases, and as such, it is set as the default CPU-based force calculation method in Haptimol_RD.

Using the octree-based method Haptimol_RD can compute in real-time (and at haptic refresh rates) the electrostatic and VDW force contributions for interacting molecules comprising up to 7k atoms each. As such it can facilitate the haptics-assisted rigid docking of large protein-protein and protein-drug complexes. This offers more than a threefold improvement on the size of the molecules existing docking systems can accommodate, and it addresses effectively all issues related to pre-computed force grids. However, it falls short when the docking case involves very large molecules. Therefore, there is a need to push that size limit further, i.e. to molecules comprising tens or even hundreds of thousands atoms each. High-end GPUs might have the answer to this question, offering an alternative execution platform for both querying approaches. Given that both querying methods exhibit high levels of execution parallelism during grid/octree traversals, it becomes apparent that they can benefit from the many-core processing capabilities of modern GPU's. The next chapter describes the adaptation of both real-time force calculation methods on the GPU.

Chapter 5

Accelerating interaction force calculations on GPU

5.1 Introduction

Chapter 4 explained how the concepts of a cut-off distance and spatial decomposition can assist haptics-assisted docking to attain real-time force updates (within 2ms) for large molecules on the CPU. Although modern CPU processing power limits this method to molecules no larger than 7k atoms each, the emergence of general purpose programmable GPU architectures might offer the additional computing power needed in order to increase this size substantially, i.e. to very large molecules. Over the years, several approaches tried to harness the many-core processing capabilities of GPUs in order to accelerate force computations in molecular interactions. Recently Anthopoulos et. al. [AGB13] reported a GPU-accelerated cut-off-based force calculation approach applicable to interactive docking. Anthopoulos et. al. applied the approach to their haptics-assisted molecular modelling simulator [APGB14] 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 33 Hz (30 ms response time). Furthermore, by design, the method cannot be applied to the docking of large molecules. Other GPU-accelerated force calculation approaches have been proposed and applied in systems pertinent to MD and automated docking (Stone et. al. [SHUS10] provides a review of the different work conducted in this area). These methods often employ grid-based proximity querying algorithms [SPF$^+$07, AGB13] or very efficient 2D energy/force matrices [HKR12] to minimize the overall computational cost and accelerate the pose scoring functionality of automated rigid-docking systems [SH09]. They compute inter and/or intra molecular interactions based on a cut-off distance or using Fast Fourier Transformations, and in some cases model molecular flexibility to some degree. However, these methods cannot be applied to haptics-assisted docking (even for large molecules) since they necessitate execution times substantially larger than 2ms [SPF$^+$07, SH09, HKR12], and/or have costly constructions/update requirements for the underlying regular grid structures [AGB13, SH09]. As it stands, existing GPU-accelerated force calculation approaches fail to address successfully many of the issues (e.g. size of molecules, force refresh rates) related to interactive, haptics-assisted molecular docking.

This chapter discusses the design and implementation of two novel GPU-accelerated force calculation approaches applicable to haptics-assisted docking [IHL15]. Both methods are a GPU-based adaptation of the respective CPU-based methods discussed in Chapter 4. A GPU-accelerated version of the brute force approach is also described here and used as a performance baseline. The next section provides a quick overview of the GPU computing environment and outlines the main design principles involved.

5.2 GPU Computing

All methods discussed in this chapter have been implemented using the Open Computing Language [OM12] (OpenCL) parallel programming framework and executed on an NVIDIA GPU [NVI]. OpenCL provides a C-like programming environment that facilitates the programmability of GPUs, and makes it easy for the developer to harness the computational power of many-core processors. OpenCL was used in order to maximize the portability of Haptimol_RD to different GPU architectures. The following paragraphs outline the nomenclature of NVIDIA GPUs, relate it to the choices made during method design, and map it to the OpenCL programming paradigm.

GPUs are high-performance, streaming processors that favour data-parallel execution and coherent memory access patterns. A modern GPU provides hundreds of computing cores that can support high rates of parallel execution (Figure 5.1). A typical GPU consists of several streaming multiprocessors (SM), each one of which contains multiple computing cores (a.k.a. Scalar Processors). The cores are responsible for the execution of a program or kernel. The basic computation unit is a thread. Each thread executes an instance of the kernel. Threads are grouped together to form thread blocks, and each block is assigned to a specific SM. A SM processes the block threads in sets of 32, called a warp, and executes them in parallel on its cores in a Single Instruction Multiple Data (SIMD) fashion. The number of resident threads against the theoretical number of threads supported by the GPU is described as occupancy. In general, high occupancy indicates better execution performance.

Thread control/access can be attained programmatically via a global/block-specific

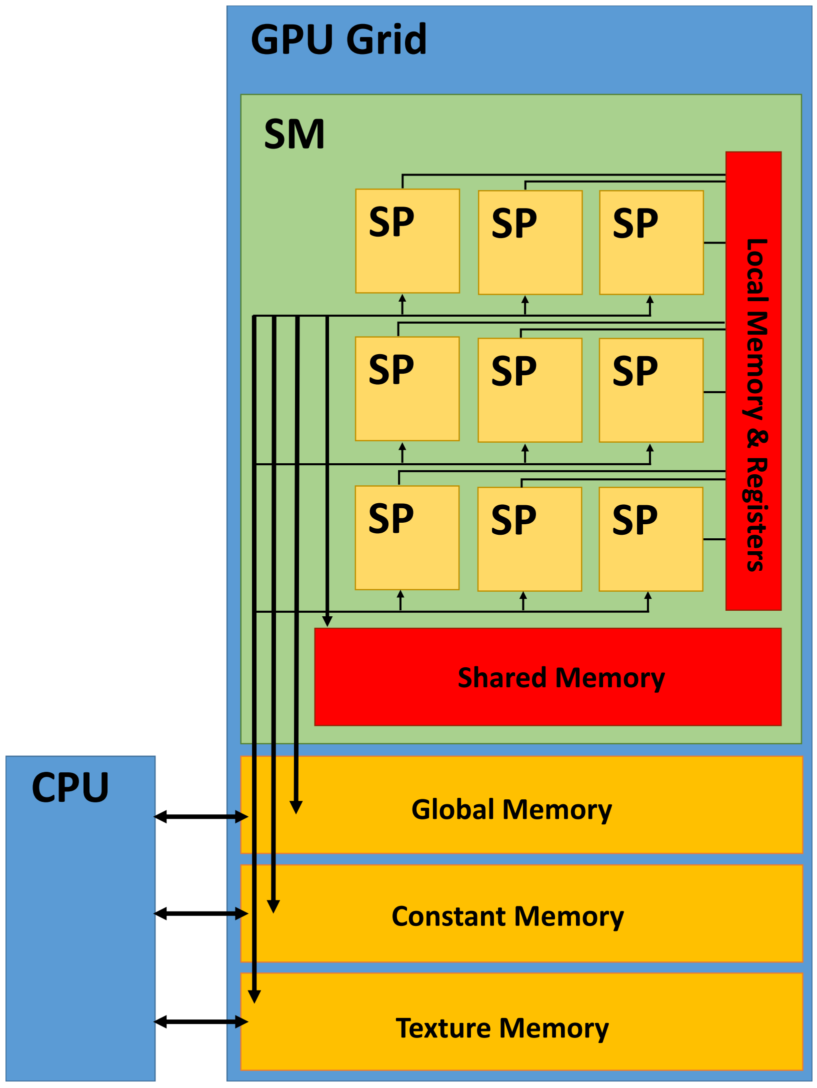


Figure 5.1: An abstraction of the main structural/functional components of a modern NVIDIA GPU (adapted from van Oosten [vO11]). The diagram shows the Scalar Processors (SP) stacked within a Streaming Multiprocessor (SM), and interfacing with the various levels of GPU memory. CPU-GPU communication can achieved only through slow, on-board memory.

ID assigned sequentially to each thread upon creation. Maximum execution parallelism is achieved when all threads within a warp execute the same kernel instruction (execution convergence). Execution divergence occurs when threads within a warp execute different kernel instructions (e.g. due to conditional statements), which forces the SM to serialize the execution of these threads, and can penalize substantially kernel performance. Thread management (e.g. creation, switching etc) is done completely in hardware with almost zero overhead. Thread communication and synchronization is possible at block level via a small (32-48KB), high-speed, on-chip (i.e. SM) memory, also referred to as shared memory. In addition to shared memory, threads have global gather/scatter, read/write access to a large (0.8-4GB), slow, on-board memory, and private access to a set of registers (very fast private memory). Global memory accesses are costly operations in terms of latency. A SM can hide memory latency by executing a large number of warps (high occupancy) and switch among them when one or more of its warps wait on a memory transaction. Another way to reduce latency cost is memory-access coalescing. When the threads in a warp access consecutive global memory locations then the hardware can combine (coalesce) these requests into a single request, and hence reduce the total number of memory transfers.

Consequently, to achieve scalability and execution efficiency a GPU-based algorithm should be very conscious of its thread utilization, instruction execution and memory access patterns [NVI10]. Namely, the algorithm should maintain a high number of occupancy at all times to maximize execution performance and hide latency, (b) attain fine-grained data parallelism to minimize execution divergence, (c) utilize shared memory whenever possible to reduce global memory accesses and (d) avoid scattered global memory reads and writes (i.e. uncoalesced memory accesses). Both cut-off-based approaches described here takes into account all of the aforementioned design principles in order to attain optimized performance.

In closing, OpenCL abstracts the different characteristics of rival GPU architectures and unifies them under a singular programming paradigm. In OpenCL terminology the GPU is referred to as a device, the SMs as compute units and the cores as processing elements. Moreover, threads are defined as work-items, thread-blocks as workgroups, and shared memory as local memory. In the following sections OpenCL and GPU terminology will be used interchangeably in order to describe the specifics of the force calculation approaches.

5.3 GPU-based Brute Force approach

Similarly to the CPU-based brute force approach (see Section 4.2), the GPU-accelerated approach accounts for all interatomic interactions between the receptor and the ligand molecules, using Algorithm 7. Algorithm 7 achieves that, by computing in parallel the pairwise interaction force between each receptor atom a_i and all ligand atoms a_j^L. Upon execution, the kernel spawns one work-item for each receptor atom, arranged in workgroups of 256 items each. Position updates are applied on the receptor atoms using the combined matrix $\boldsymbol{T_{New}}$ (see Section 4.3.4).

Algorithm 7 was implemented and then tested using the same testing methodology described in Section 4.2. Again the goal here was to discover how many atom pairs could be computed with Algorithm 7 within 2ms. The testing started with an array size of two hundred and fifty atoms per molecule (the CPU limit), and were concluded when the arrays reached one thousand four hundred and thirty atoms each. Initially the array increments were one hundred atoms long and were gradually reduced to ten atoms as the tests proceeded. The performance times measured included kernel

Algorithm 7 GPU-accelerated Brute Force

```
Require: $Receptor$, array of atom structures
Require: $Ligand$, array of atom structures
Require: $T_{New}$, combined viewing transformation matrix
Ensure: $f_i$, receptor atom force subtotal
 1: for all atoms $a_i$ in $Receptor$ do in parallel
 2:     ltID ← GetLocalThreadID(); grID ← GetGroupID()
 3:     $f_i \leftarrow 0$
 4:     // adjust receptor atom coordinate
 5:     $a_i.coord \leftarrow a_i.coord * T_{New}$
 6:     for all atom structures $a_j^L$ in Ligand do
 7:         d ← distance($a_i$, $a_j^L$)
 8:         $f_i \leftarrow f_i$ + computeForce($a_i$, $a_j^L$, d)
 9:     end for
10: end for
11: end
```

queuing and submitting times but not the time taken for uploading the 1D arrays of atoms on the GPU (since this information will never change). All of the experiments were executed on an NVIDIA GTX580 GPU with 1.5GB RAM.

According to these results, the GPU version of the brute force approach can accommodate real-time force calculations for molecules comprising up to 1.5k atoms each. This is an almost sixfold improvement over the CPU-based brute force method, and a good indication that analogous performance gains might be attainable for the cut-off-based force calculation approaches as well. The next section discusses the adaptation of the two CPU-based proximity querying methods, discussed in Chapter 4, on the GPU.

5.4 Force calculations using GPU-accelerated proximity querying

The GPU-based force calculation methods described in this section identify the set of interacting atom pairs within a cut-off distance using the same space partition structures (i.e. regular grids and octrees) and querying logic as their CPU analogues. Both GPU-based approaches employ a parallelized version of their respective querying algorithm, designed to exploit the execution parallelism of modern GPUs. Given that there are different GPU architectures and that memory transfers from CPU to GPU (and vice versa) are costly operations, extra design considerations (other than the ones outlined in Section 5.2) were made in order to address that. Namely, both methods should: a) be capable of supporting GPUs of different memory sizes and computing capabilities, and b) necessitate minimal precomputation/construction requirements on the underlying molecules. Existing GPU-based proximity querying algorithms such as the one described in Lauterbach et. al. [LMM10] cannot be applied here, because they perform distance queries only on geometry located on the surface of an object and not within the inner part of the object as required in our case, i.e. identifying those atoms within a cut-off distance residing inside the molecular surface. The following subsections describe the respective grid/octree construction and querying algorithms.

5.4.1 Constructing Spatial Partitioning Structures

To minimize construction overhead, the methods construct the regular grid and octree structures only for the molecule with the least number of atoms, thus leaving the larger molecule (often the receptor) free of any spatial partitioning constraints. Both partitioning structures are built on the CPU, and then transferred to the GPU as

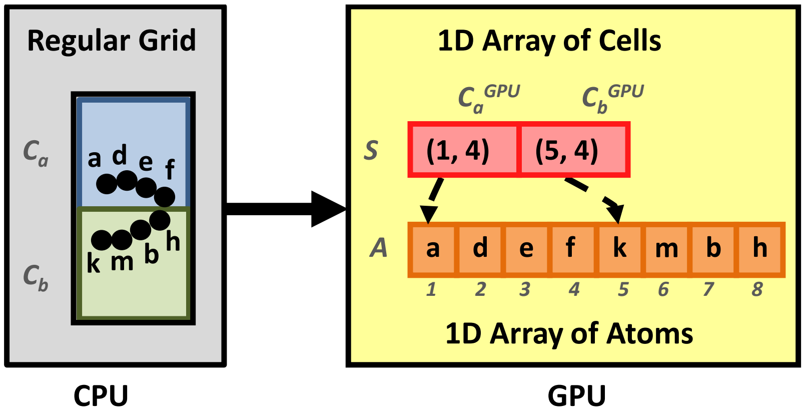


Figure 5.2: A 2D depiction of a regular grid built on the CPU and transferred to the GPU as a 1D array of cell records S and 1D array of atoms A. The initial grid consisted of the cells C_a and C_b containing the atoms a,d,e,f and k,m,b,h, respectively. Both cells are represented in the S array as cell 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). A similar 1D array is built for the octree as well.

a 1D array of cells 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 5.2).

The regular grid is constructed on the CPU using the approach taken in Section 4.3.2. The 1D cell array S for the GPU, is then obtained by a) looping through the grid in an x first, y second, z last order, b) mapping the 3D grid cell index into a 1D index, and c) using this index to assign the cell in the 1D array. A cell is 16 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. During construction the cell size c_g is set based on the formula $c_g = nr_C$ (r_C is 1.7Å, the radius of a carbon atom), where n is determined empirically (see Section 5.5). The actual cell size used might change slightly in order to divide the bounding box into an integer number of subdivisions.

Likewise, the pruned octree is constructed on the CPU using the method described in Section 4.3.3. The 1D octant array S is then obtained by executing a breadth-first traversal of the tree and assigning the respective octants in the array in that order. An octant is a 32-byte structure, and contains an index to 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), 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). Unlike the octrees constructed for the CPU-based querying method, the number of octree levels, L, constructed here depends on the size of the molecule, and is decided dynamically using Equation 5.4.1,

$$L = min\left(\left\lfloor log_2\left(\frac{max(\ell_x, \ell_y, \ell_z)}{c_o}\right)\right\rfloor, L_{max}\right) \tag{5.4.1}$$

where L is the octree subdivision target, c_o is the targeted side-length of a leaf octant (i.e. the length of one of the bounding cube sides), and L_{max} is the maximum subdivision level the GPU-based query algorithm can support, i.e. 7 due to memory constraints. The equation has as numerator the maximum side of the bounding box because the 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 (see Section 5.5). 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, this construction strategy allows both force calculation approaches 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 the 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, the approach can facilitate, in principle, docking problems that model receptor flexibility.

5.4.2 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 5.3). Again, atom position updating is done using the combined viewing transformation matrix $\boldsymbol{T_{New}}$, with the only difference that the formula used here is $\boldsymbol{T_{New}} = \boldsymbol{T_L}^{-1}\boldsymbol{T_R}$, since $\boldsymbol{T_{New}}$ is applied to the smallest molecule (ligand) and not to the larger one (receptor) as before (see Section 4.3.4).

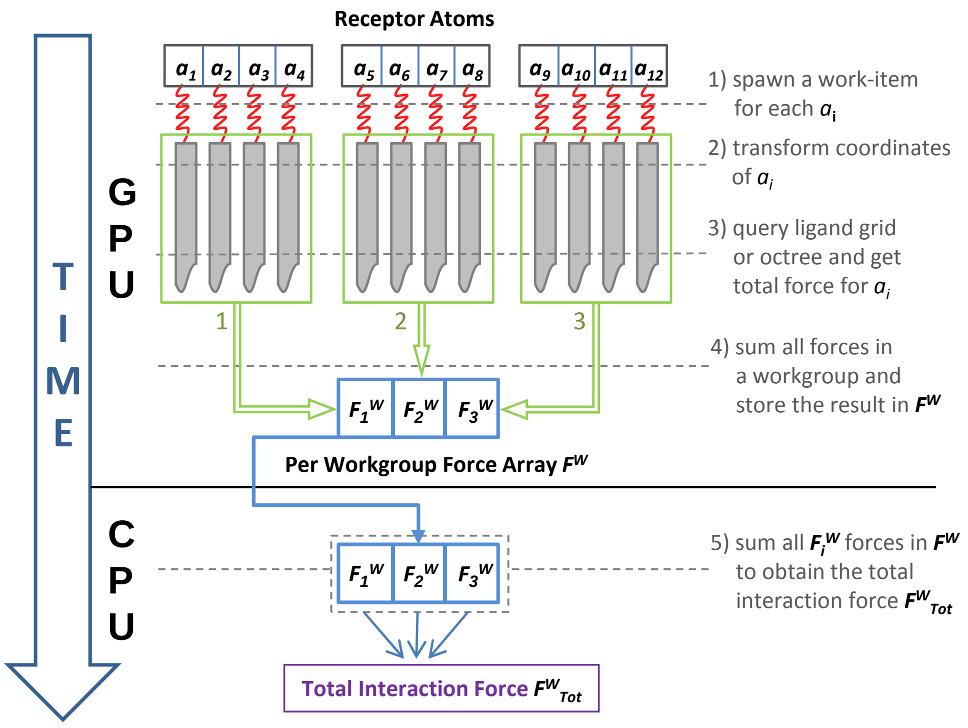

Receptor Atoms
a_1 a_2 a_3 a_4
a_5 a_6 a_7 a_8
a_9 a_{10} a_{11} a_{12}
1) spawn a work-item for each a_i
2) transform coordinates of a_i
3) query ligand grid or octree and get total force for a_i
4) sum all forces in a workgroup and store the result in F^W
TIME
GPU
1
2
3
$F_1{}^W$ $F_2{}^W$ $F_3{}^W$
Per Workgroup Force Array F^W
CPU
5) sum all $F_i{}^W$ forces in F^W to obtain the total interaction force $F^W{}_{Tot}$
Total Interaction Force $F^W{}_{Tot}$

Figure 5.3: A visualization of the 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 the 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 $\boldsymbol{F_i^W}$ in a global number-of-workgroups-long force array $\boldsymbol{F^W}$ (execution step 4). Array $\boldsymbol{F^W}$ is transferred back to the CPU, where its entries are accumulated to produce the total interaction force $\boldsymbol{F_{Tot}^W}$ (execution step 5).

The following list outlines the key execution steps of both approaches.

1. Spawn a work-item for every atom a_i within the largest molecule and group them into workgroups.

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.

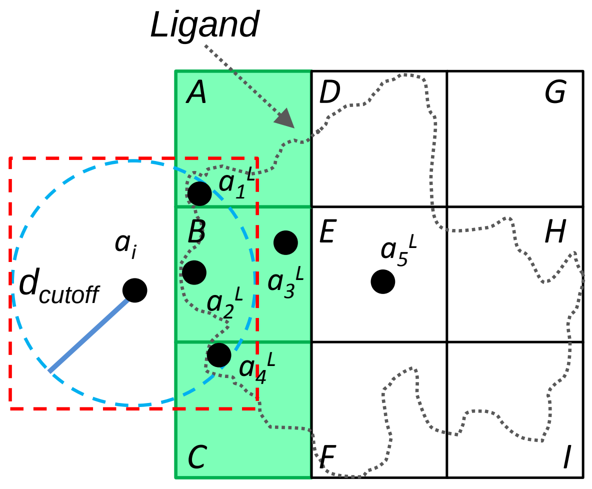
   (b) Compute the force for all pairs in the set.

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.

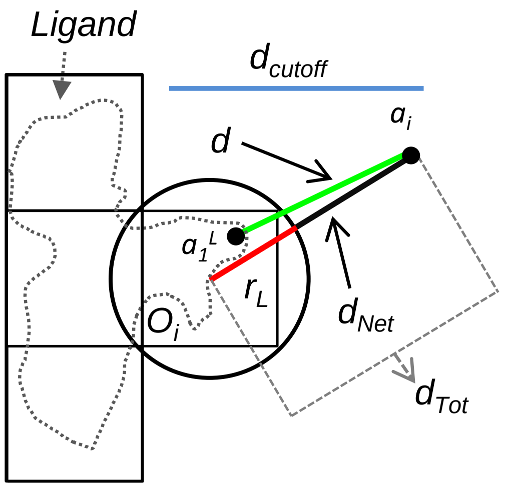
5. Sum the partial forces in $\boldsymbol{F^W}$to obtain the total force $\boldsymbol{F_{Tot}^W}$.

Steps 1), 2), 4) and 5) are steps common to both partitioning structures. The execution flow differs in Step 3) because each method queries its respective structure (i.e. regular grid and 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 5.4). Like their CPU-based counterparts, both methods can facilitate the independent/dynamic handling of the electrostatic and VDW forces (i.e. scale/switch on-off electrostatics, vdW repulsive and/or vdW attractive parts), and thus enable the user experiment with different types of interactions easily.

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



(a)



(b)

Figure 5.4: A conceptual 2D visualization of the 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 (similarly to the CPU-based grid querying). 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 $a_i a_1^L$).

Querying and Calculating Forces Using a Regular Grid

The method utilizes 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 computes the total force on this set. It begins 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 atom coordinates, a search region of grid cells is identified using Algorithm 4, ***GetSearchRange*** (see Section 4.3.5).

Like in CPU-based grid querying, Algorithm 4 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 5.4a). 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 $\boldsymbol{F^W}$. The total force is computed by accumulating the entries in $\boldsymbol{F^W}$. 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 this accumulation is done on the CPU since the CPU can perform this summation faster than the 0.2ms

Algorithm 8 GPUQueryRegularGrid

```
Require: $Receptor$, array of atom structures
Require: $A$, regular grid query info structure
Require: $S$, regular grid query info structure
Require: $RG_{Info}$, regular grid query info structure
Require: $FC_{Info}$, general force calculation info structure
Require: $d_{cutoff}$, the cut-off distance
Ensure: $F^W$, array of work-group-force subtotals $F_i^W$
1: for all atoms $a_i$ in $Receptor$ do in parallel
2:   ltID ← GetLocalThreadID(); grID ← GetGroupID()
3:   $f_i \leftarrow 0$
4:   // adjust receptor atom coordinate
5:   $a_i.coord \leftarrow a_i.coord * FC_{Info}.T_{New}$
6:   // execute Algorithm 4
7:   $GetSearchRange(RG_{Info}, a_i.coord, d_{cutoff}, x^G, y^G, z^G)$
8:   for l=$x^G$.min to l≤ $x^G$.max with l++ loop do
9:     for k=$y^G$.min to k≤ $y^G$.max with k++ loop do
10:      for j=$z^G$.min to j≤ $z^G$.max with j++ loop do
11:        indx ← l*$RG_{Info}.n_x$*$RG_{Info}.n_y$+k*$RG_{Info}.n_x$+j
12:        gridCell ← $S$[indx]
13:        if gridCell not empty then
14:          for all atom indices $s_{IN}$ in gridCell do
15:            $a_i^L \leftarrow A[s_{IN}]$
16:            d ← distance($a_i$, $a_i^L$)
17:            if d ≤ $d_{cutoff}$ then
18:              $f_i \leftarrow f_i$ + computeForce($a_i$, $a_i^L$, d)
19:            end if
20:          end for
21:        end if
22:      end for
23:    end for
24:  end for
25:  lclForce[ltID] ← $f_i$
26:  synchronize threads
27:  if ltID is first thread in workgroup then
28:    $F^W$[grID] ← sum(lclForce)
29:  end if
30: end for
```

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. $\lceil \frac{receptoratoms}{256} \rceil$. 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 the method to perform the majority of force calculations on the GPU in a memory-coalesced fashion, and hence optimize the performance of this method. Algorithm 8, ***GPUQueryRegularGrid***, outlines the aforementioned key execution steps.

Querying and Calculating Forces Using an Octree

Similar to the grid-based algorithm, the 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 [NVI], 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 a stack-based, octree querying method was developed 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). The size of the stack is set 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 the maximum subdivision level supported 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. d_{Net} is computed using Equation 5.4.2,

$$d_{Net} = d_{Tot} - r_L \tag{5.4.2}$$

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 the regular grid method) only for those receptor/ligand atom pairs with $d \leq d_{cutoff}$ (Figure 5.4b). 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 the 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 9, ***GPUQueryOctree***, describes the main steps of this octree-based force calculation approach.

Algorithm 9 GPUQueryOctree

```
Require: $Receptor$, array of atom structures
Require: $A$, regular grid query info structure
Require: $S$, regular grid query info structure
Require: $FC_{Info}$, general force calculation info structure
Require: $d_{cutoff}$, the cut-off distance
Ensure: $F^W$, array of work-group-force subtotals $F_i^W$
 1: for all atoms $a_i$ in $Receptor$ do in parallel
 2:    ltID $\leftarrow$ GetLocalThreadID(); grID $\leftarrow$ GetGroupID()
 3:    stack[56] $\leftarrow$ -1; next $\leftarrow$ 0 (i.e. index of root octant); $f_i \leftarrow 0$
 4:    // adjust receptor atom coordinate
 5:    $a_i.coord \leftarrow a_i.coord{*}FC_{Info}.T_{New}$
 6:    do
 7:       octant $\leftarrow$ $S$[next]; next $\leftarrow$ -1; count $\leftarrow$ 0
 8:       // breadth-first traversal to identify level overlaps
 9:       for all child octant indices $o_{IN}$ in octant do
10:          $d_{Tot} \leftarrow$ distance($a_i$, $S[o_{IN}]$)
11:          $d_{Net} \leftarrow d_{Tot}$-$r_L$
12:          if $d_{Net} \leq d_{cutoff}$ and $S[o_{IN}]$.isLeaf then
13:             for all atom indices $s_{IN}$ in leaf $S[o_{IN}]$ do
14:                $a_i^L \leftarrow A[s_{IN}]$
15:                d $\leftarrow$ distance($a_i$, $a_i^L$)
16:                if d $\leq d_{cutoff}$ then
17:                   $f_i \leftarrow f_i$ + computeForce($a_i$, $a_i^L$, d)
18:                end if
19:             end for
20:          else
21:             // depth-first traversal to move down the tree per level
22:             if next == -1 then
23:                next $\leftarrow o_{IN}$
24:             else
25:                stack.push($o_{IN}$) in reverse
26:             end if
27:          end if
28:       end for
29:       // use the stack if level traversing is completed
30:       if next == -1 then
31:          next $\leftarrow$ stack.pop()
32:       end if
33:    while (next $\geq$ 0)
34:    lclForce[ltID] $\leftarrow f_i$
35:    synchronize threads
36:    if ltID is first thread in workgroup then
37:       $F^W$[grID] $\leftarrow$ sum(lclForce)
38:    end if
39: end for
```

5.5 Performance Testing of the GPU-based methods

Both methods were implemented using Visual C++ and OpenCL 1.1, and integrated within Haptimol_RD. Similarly to the CPU-based methods, a series of experiments were conducted in order to benchmark the performance of these methods (against demanding simulation loads), compare them to the CPU-based implementation, and measure their efficiency during interactive rigid-docking simulations on known complexes. Again, all tests were executed on a 2.93GHz Intel Core i7 PC running under a 64bit version of Windows 7 with an NVIDIA GTX580 GPU, and a 3DOF Geomagic Touch haptic device. The PC was equipped with 8GB RAM, and the GPU with 1.5GB RAM. Likewise, arbitrary force parameters were used in benchmarking and GPU-CPU performance comparison tests, whereas actual force parameters (see Section 3.4) were used during the interactive rigid-docking simulations. The emphasis here is on measuring the performance of the two proximity querying methods executed on the GPU, and therefore the construction of the partitioning structures are not reported as they are pre-computed on the CPU.

5.5.1 Benchmarking Experiments on the GPU

The benchmarking experiments conducted here, follow the same logic as the one described in the previous chapter, e.g. use artificial docking simulations, allow atom overlap, model molecules as rigid structures. The experiments utilized the molecules *Alcohol Dehydrogenase* dimer (1ADG), *Aspartate Carbamoyltransferase*, (1AT1), *GroEL-E434K Mutant* (2YEY) and *Clathrin* (1XI4), as defined in their respective PDB files (Figure 5.5). Using these proteins the following four artificial protein-protein docking test cases were generated: 1ADG-1ADG (i.e. 1ADG with 1ADG), 1AT1-1AT1,

Table 5.1: 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.

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

2YEY-2YEY and 1XI4-1XI4. For each test case seven rigid docking simulations were conducted using regular grids of different cell sizes (c_g values equal to nr_C were used, 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 a 4×4 matrix specified the position and orientation of the ligand in respect to the receptor. Each matrix ensured that the ligand would overlap with the receptor, and generate a substantial set of interatomic interactions to benchmark sufficiently the respective force calculation method. Unlike CPU-related benchmarking (were only 10 repetitions sufficed), each simulation recorded 10000 different response times (i.e. a simulation time of about 10ms) in order to even out the effect of spurious performance spikes (explained below). The percentage of those responses found below 1ms, within 1-2ms(inclusive), within 2-4ms and above 4ms (Figures 5.6c and 5.6d) was also computed, since there are reports suggesting that acceptable haptic refresh rates in some cases can go as low as 250-300Hz[MFC$^+$14, DDKA06]. Table 5.1 reports test-case/construction specific data, whereas Figures 5.6a-5.6d and Tables 5.2 and 5.3 report simulation specific data. In all experiments, d_{cutoff} was set equal to 8Å.

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

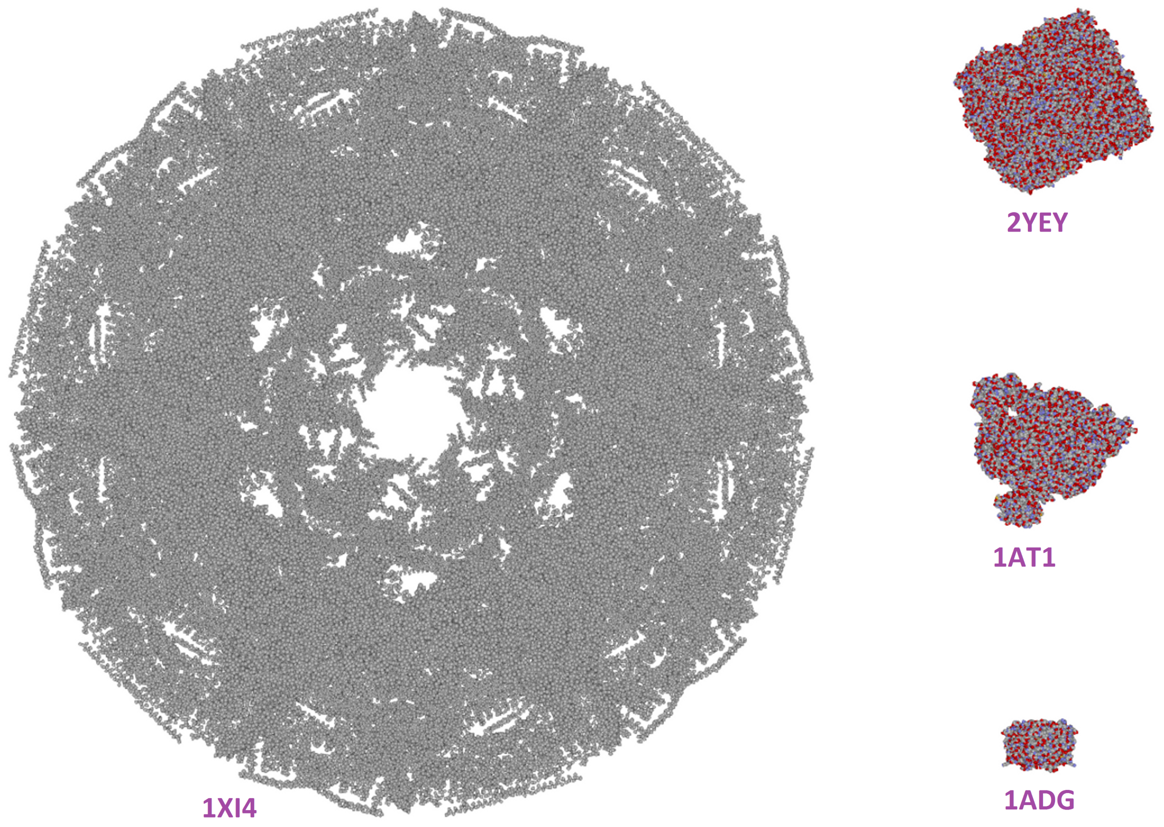
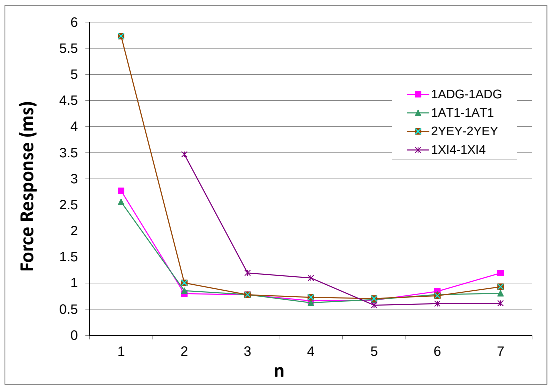
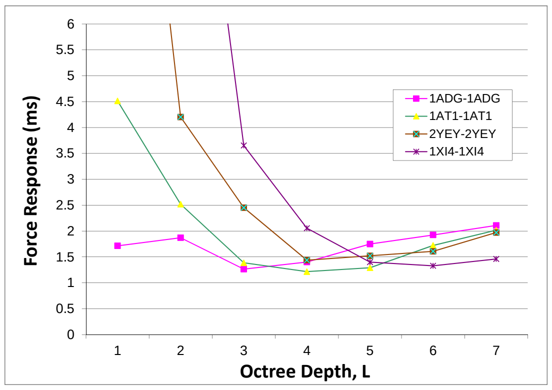


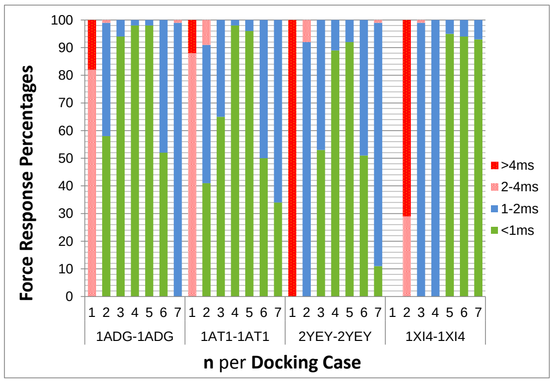
Figure 5.5: The four molecules used in these benchmarking experiments, showing their relative sizes. 1XI4 is the largest one with 184k atoms, and a bounding box with largest axis of 747.22Å in z (see Table 5.1).



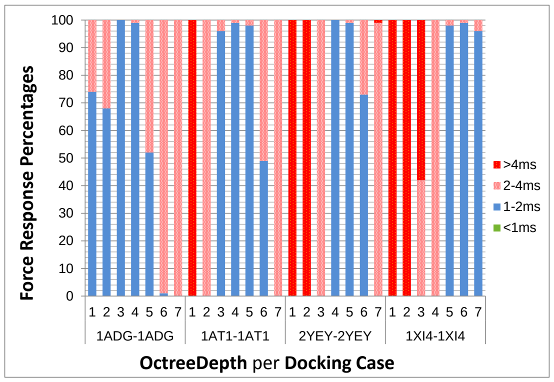
(a)



(b)



(c)



(d)

Figure 5.6: 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 slightly 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 5.2: Benchmarking the regular-grid-based method. The grids are constructed with cell sizes c_g equal to nr_C, where n=1,2,..7 and r_C is the radius of a carbon atom. The table lists the total number of interacting atom pairs (S_{Pairs}), the GPU memory allocated for the grid, the actual size of the cell in Å, the value of n used, the best response time attained, and the percentages (rounded to the nearest integer) of those 10000 response times found below 1ms, within 1-2ms(inclusive), within 2-4ms and above 4ms.

Complex	S_{Pairs}	**Memory (bytes)**	**Cell Size (Å)**	**n**	**Best (ms)**	**<1ms**	**1-2ms**	**2-4ms**	**>4ms**
1ADG-1ADG	20024	6144	12.5	7	1.2	0%	99%	1%	0%
		8748	11.2	6	0.84	52%	48%	0%	0%
		15972	8.6	5	0.67	98%	2%	0%	0%
		32928	7.0	4	0.66	98%	2%	0%	0%
		82308	5.3	3	0.78	94%	6%	0%	0%
		292668	3.5	2	0.80	58%	41%	1%	0%
		2464548	1.7	1	2.77	0%	0%	82%	18%
1AT1-1AT1	20042	15972	12.5	7	0.81	34%	66%	0%	0%
		26364	10.8	6	0.78	50%	50%	0%	0%
		49152	8.9	5	0.69	96%	4%	0%	0%
		96000	6.8	4	0.63	98%	2%	0%	0%
		236196	5.2	3	0.78	65%	35%	0%	0%
		827052	3.4	2	0.86	41%	50%	9%	0%
		6861444	1.7	1	2.56	0%	0%	88%	12%
2YEY-2YEY	20032	32928	12.3	7	0.93	11%	88%	1%	0%
		58956	10.3	6	0.76	51%	49%	0%	0%
		96000	8.8	5	0.71	92%	8%	0%	0%
		187500	6.8	4	0.73	89%	11%	0%	0%
		471648	5.1	3	0.78	53%	47%	0%	0%
		1591812	3.4	2	1.01	0%	92%	8%	0%
		12734496	1.7	1	5.73	0%	0%	0%	100%
1XI4-1XI4	20034	2859936	12.1	7	0.62	93%	7%	0%	0%
		4668204	10.2	6	0.61	94%	6%	0%	0%
		7902036	8.6	5	0.58	95%	5%	0%	0%
		15540348	6.9	4	1.10	0%	100%	0%	0%
		37345632	5.1	3	1.20	0%	99%	1%	0%
		126041508	3.4	2	3.47	0%	0%	29%	71%
		1015254228	1.7	1	N/A	N/A	N/A	N/A	N/A

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, fluctuations between the best and worst response times (in all simulations) were observed, reaching in some instances a difference of up to 1.5ms. These performance fluctuations are attributed to intervening CPU/GPU workloads (e.g. background processes, display rendering). 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 5.6a). 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 the grid-based approach as a universal subdivision criterion for regular grids. In the case of octree-based querying, the first step was to identify the subdivision levels L with the fastest response times (Figure 5.6b). Given this information, the formula $max(\ell_x,\ \ell_y,\ \ell_z)/2^L$ was then used in order to obtain the actual side-length of the leaf octants, and relate this length to r_C, i.e. found those multiples of r_C that would cause the method to construct a tree of level L. Using Equation 5.4.1 L was determined 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 Table 5.3) when c_o=5r_C. In all cases, the octrees occupied less memory, at the respective leaf/cell sizes than the regular grids (Tables 5.2 and 5.3), 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, both approaches were able to construct their respective partitioning structures on the GPU. It became impossible (for the GPU used) to find a PDB file that would force Haptimol_RD to construct an octree instead of a regular grid. To overcome this, a test molecule was created out of four 1XI4 molecules. The molecules were aligned along the main diagonal of the new 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, the docking case 4_1XI4-4_1XI4 was generated and benchmarked, at a targeted octant size c_o=5r_C. For this test case, Haptimol_RD utilized an octree (of 1.4MB), since the GPU could not allocate the 463.5MB of continuous memory needed for storing the regular grid. Again, both molecules were allowed to overlap along their longest surface, and as such generated a set S_{Pairs} of 27151 interacting atom pairs. In this case, the octree-based method averaged force response times at 4.6ms, indicating that simulation cases of such size pose an upper limit to this method. More memory on the graphics card would have allowed, of course, Haptimol_RD to attain relatively faster response times for the same docking case (i.e. 4_1XI4-4_1XI4) by letting it utilize the grid-based method instead of the octree-based method, but such tests could not be performed here due to the memory limitations of the NVIDIA GTX580 GPU.

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

Table 5.3: Benchmarking the octree-based method at subdivision levels 1-7. The table lists per subdivision level the total number of interacting atom pairs (S_{Pairs}), the GPU memory allocated for the octree, the actual leaf octant's size in Å, the subdivision level, the best response time attained, and the percentages (rounded to the nearest integer) of those 10000 response times found below 1ms, within 1-2ms(inclusive), within 2-4ms and above 4ms.

Complex	S_{Pairs}	**Memory (bytes)**	**Leaf Size (Å)**	**Level**	**Best (ms)**	**<1ms**	**1-2ms**	**2-4ms**	**>4ms**
1ADG-1ADG	20024	288	56.1	1	1.71	0%	74%	26%	0%
		928	28.0	2	1.87	0%	68%	32%	0%
		4384	14.0	**3**	1.27	0%	100%	0%	0%
		20608	7.0	4	1.40	0%	99%	1%	0%
		100320	3.5	5	1.75	0%	52%	48%	0%
		282272	1.8	6	1.92	0%	1%	99%	0%
		507072	0.9	7	2.11	0%	0%	100%	0%
1AT1-1AT1	20042	288	75.3	1	4.51	0%	0%	0%	100%
		1568	37.6	2	2.52	0%	0%	100%	0%
		7744	18.8	3	1.39	0%	96%	4%	0%
		36864	9.4	**4**	1.22	0%	99%	1%	0%
		183904	4.7	5	1.29	0%	98%	2%	0%
		643840	2.4	6	1.72	0%	49%	51%	0%
		1318400	1.2	7	2.02	0%	0%	100%	0%
2YEY-2YEY	20032	288	92.3	1	14.92	0%	0%	0%	100%
		20820	46.1	2	4.20	0%	0%	0%	100%
		10784	23.1	3	2.45	0%	0%	100%	0%
		56352	11.5	**4**	1.44	0%	100%	0%	0%
		299232	5.8	5	1.52	0%	99%	1%	0%
		1251008	2.9	6	1.61	0%	73%	27%	0%
		2864576	1.4	7	1.97	0%	0%	99%	1%
1XI4-1XI4	20034	288	373.6	1	74.11	0%	0%	0%	100%
		2208	186.8	2	12.96	0%	0%	0%	100%
		12448	93.4	3	3.65	0%	0%	42%	58%
		65440	46.7	4	2.06	0%	0%	100%	0%
		313760	23.4	5	1.40	0%	98%	2%	0%
		1351584	11.7	**6**	1.33	0%	99%	1%	0%
		5003808	5.8	7	1.46	0%	96%	4%	0%

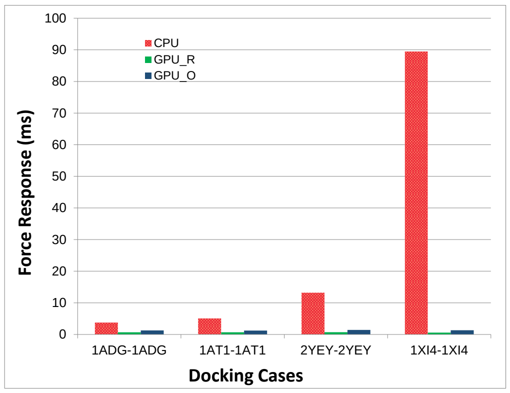
response times regardless of the underlying molecule sizes. The one-to-one work-item-per-atom strategy adapts very well to the Single Instruction Multiple Data execution model of the GPU, and thus utilizes efficiently the GPU's computational resources.

5.5.2 GPU-CPU Comparisons

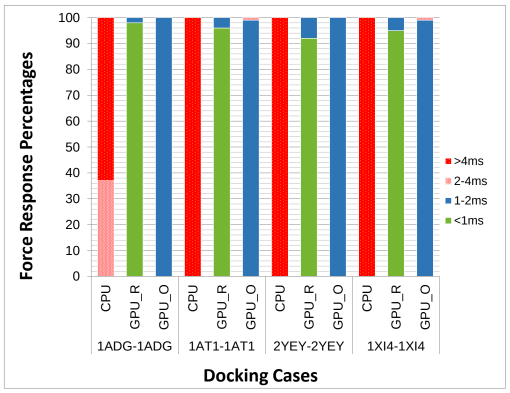
Both GPU-accelerated methods were compared to the CPU-based method using octrees. The purpose of these tests was to identify/measure the performance gains attained by the GPU methods over the CPU method. The methods were not compared to other current CPU-based approaches (e.g. brute force, pre-computed force-grid based etc.) since reportedly they cannot manage molecular systems of more than a couple of thousand of atoms each [RAM+12].

To obtain comparable results, the CPU-based method was tested 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). In these tests, the subdivision levels of all octrees were set equal to 4, as stated in the Section 4.3.3. Comparison results, per test case and querying method (CPU, GPU-Regular grid, and GPU-Octree), were then reported as follows: a) best response times obtained (Figure 5.7a), and b) best response-time intervals for these 10000 iterations (i.e. <1ms, 1-2ms, 2-4ms, >4ms) as percentages (Figure 5.7b).

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



(a)



(b)

Figure 5.7: GPU-CPU force response comparisons between the two GPU-accelerated force-calculation methods (i.e. regular grid/GPU-R and octree/GPU-O) and the octree-based CPU-force-calculation described in Section 4.3.6. 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 slightly 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 5.4.1 with $c_o = 5r_c$ and CPU-based octrees of Level, L=4.

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.

5.5.3 Haptics-assisted Interactive Rigid-Docking Simulations on the GPU

In addition to the above, performance tests for the grid-based method (the best performing GPU-based method of the two) were also conducted using rigid-docking simulations of know compounds. Similar tests were conducted in the previous chapter for the octree-based approach (Section 4.4.2). Like the CPU tests, the purpose of these simulations was: 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. To obtain comparable results, these simulations utilized the three complexes (i.e. *EGF* with *EGF receptor*, *BPTI* with *Trypsin*, anticancer drug *BAY43-9006* with *B-raf*) used in the CPU tests with the addition of the *GroES*-*GroEL* complex as defined in 1GRU PDB file. Using these complexes six rigid-docking simulations were conducted related to protein-protein and protein-drug docking. The first four (one simulation per complex) captured the relation between force response times and number of interacting atom pairs using the methodology described in Section 4.4.2, i.e. the ligand was moved around and steered to its binding conformation. Again, each simulation ran

Table 5.4: Structural information for the eight molecules used in the 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
EGFr	5836	113.09
GroES	6321	102.70
GroEL	66451	374.17

for approximately one minute, and the values recorded (at 10 millisecond intervals) were the force response times and the number of interacting atom pairs. Unlike the first four simulations, the last two were conducted in order to identify how the number of interacting atom pairs affects the rendering stability of the force (especially when the molecules become very large). The complexes B-raf-sorafenib and GroEL-GroES (the smallest and largest case) were used in these simulations each of which lasted for approximately 6ms, i.e. just moving the ligand around the receptor. During these tests the values recorded at each haptic frame were the total force, and the number of interacting atom pairs. Table 5.4 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 5.8 and 5.9 illustrate graphically the results obtained from these simulations. The force parameters and the PDB files (containing the interacting molecules) used in these simulations were obtained with the same technique described in Section 4.4.2.

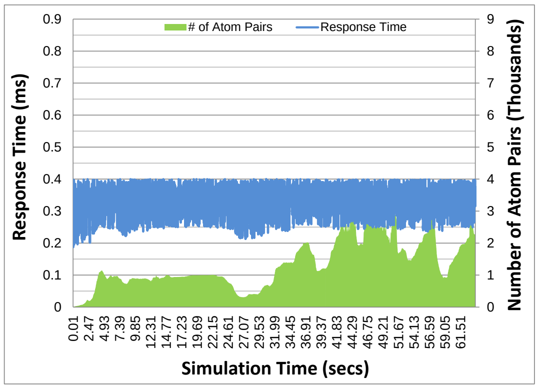
The results show that all interaction forces were calculated in less than one millisecond throughout the simulation period and for varying numbers of atom pairs.

In general, the interaction forces displayed and felt on the haptic device were fairly smooth, without any device-induced instabilities and vibrations. When the simulation involved very large structures however (GroEL-GroES), rapid force fluctuations in magnitude and direction (especially when the molecules were in contact) induced device jittering which could be perceived by the user as unstable force rendering. Force scaling/smoothing methods such as the ones described in Chapter 6 were implemented in order to 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.

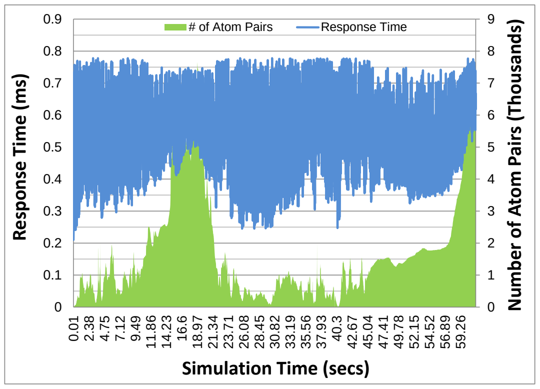
5.6 Implementing a hybrid approach

Even though the grid-based method outperformed the octree-based method consistently in all test cases, both methods were able to attain force responses at haptic refresh rates. Moreover, in all cases the octrees had a smaller memory footprint than the grids, which poses them as a useful alternative when GPU memory is a scarce commodity. With that in mind, Haptimol_RD implements/utilizes both force calculation methods interchangeably and unites them under one hybrid approach. This 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 Haptimol_RD uses the Equation 5.6.1,

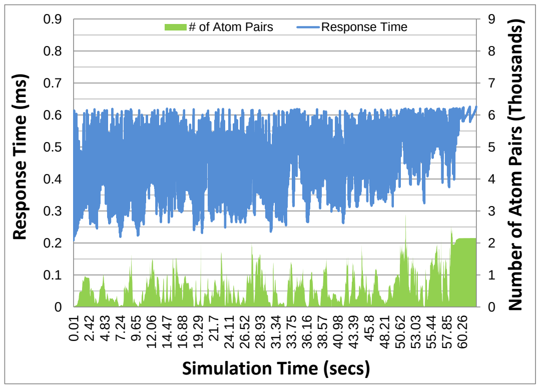
$$m_G = \left\lfloor \frac{\ell_x}{c_g} \right\rfloor \left\lfloor \frac{\ell_y}{c_g} \right\rfloor \left\lfloor \frac{\ell_z}{c_g} \right\rfloor c_b \tag{5.6.1}$$



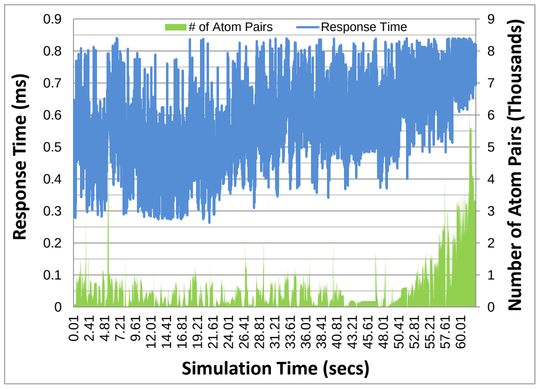
(a)



(b)

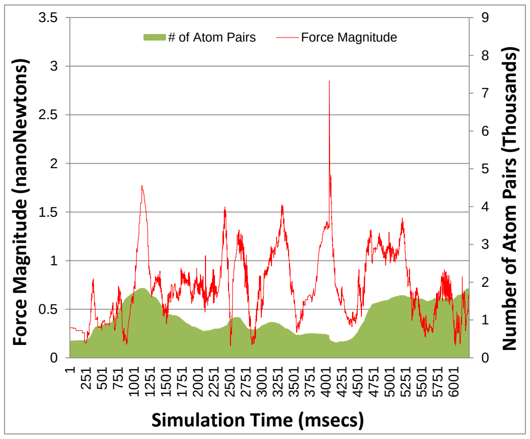


(c)

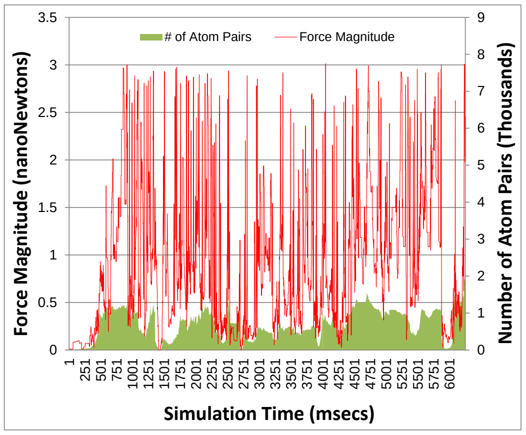


(d)

Figure 5.8: 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 graphs depict the force response times attained, at 10ms intervals, and the respective sets of interatomic interactions accounted for by the grid-based approach during the simulation.



(a)



(b)

Figure 5.9: 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 graphs depict 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.

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 (see Section 5.4.1). When m_G is less than or equal to the GPU's available memory, Haptimol_RD utilizes the grid-based method; otherwise it utilizes the octree-based method. All sizes in Equation 5.6.1 are measured in Ångstrom, c_b is 16 bytes and $c_g = 8.5$Å since it is the recommended cell size for optimal grid-querying performance (see Section 5.5.1). The actual cell size used, however, might change slightly in order to divide the bounding box into an integer number of subdivisions. Because of this hybrid approach, Haptimol_RD can support adaptively a wide range of GPUs.

5.7 Balancing occupancy and execution convergence on the GPU

Both force calculation methods discussed here employ one work-item per each receptor atom in order to attain maximum GPU occupancy. As seen however, these methods are not immune to execution divergence, since their atom-based localized search contains conditional statements that can alter the execution flow of the respective thread within its warp. To address this, and to check whether or not there is a better strategy, two additional strategies were implemented and tested, both of which were designed to achieve optimum execution convergence while maintaining similar/reasonable levels of occupancy. The first strategy uses one work-item per each receptor atom, but in this case the receptor atom performs the distance test to all cells/leaf octants indiscriminately, i.e. fewer conditions to manage but more ligand atoms to check per thread. For the second strategy, two space partition structures had to be built: one for the ligand the other for the receptor. Each receptor-related cell/leaf octant was then assigned to a different work-item and performed the same distance checks to all ligand-related cells/leaf octants, as the first strategy did. Both strategies were tested with the 2YEY-2YEY and 1XI4-1XI4 complexes (see Section 5.5.1) at various cell/octree subdivision sizes/levels and on the same computing platform used for performance testing. None of the strategies however managed to achieve force responses within 2ms. Moreover, both strategies suffered a 1.3-5.2 ms execution overhead (depending on the strategy, querying method, molecular size and grid/octree structure used) right from the beginning, since their atom-cell/octant or cell/octant-cell/octant distance checks were performed at all times, irrespectively of where the molecules were in space, i.e. within cut-off or not. As such, these strategies were not examined any further.

Although this work does not reflect an exhaustive search of all possible GPU-based querying strategies, it does indicate that modern GPUs favour occupancy more than they favour execution convergence. The recent work of Kaluschke et. al [KZD^{+}14], provides additional support to this argument, and justifies the use of a strategy that opts for maximum GPU occupancy rather than minimum execution divergence.

5.8 Testing different GPUs

All experiments thus far tested how querying scales with molecular size on the same GPU, with the question how it scales with different faster GPU architectures still remaining unanswered. Moreover, the relation between the number of processing cores/clock speeds, querying response times, and molecular sizes is unclear, and as such poses an equally interesting question. In an attempt to answer them, additional tests were conducted using the grid-based querying method, two different GPUs, and six different molecules of various sizes. The two GPUs used were the GTX 580, and GTX 980 architectures from NVIDIA. Likewise, the molecules tested were the 1AT1, 2YEY, and 1XI4 proteins (used in benchmarking), the *Myosin II* (1MVW) and *Glutamine Synthetase* (1HTQ) proteins, and an artificial structure named 3_1XI4 which was created out of three 1XI4 molecules in a similar way to the 4_1XI4 molecule. The GPUs varied in the number of computing cores, memory sizes and clock speeds, whereas the molecules varied in the number of atoms (ranged from 20 thousands up to 1 million atoms), and bounding box sizes. Tables 5.5 and 5.6 list technical and structural information for the GPUs and the molecules respectively. Out of these proteins the following six test cases were generated: 1AT1-1AT1, 2YEY-2YEY, 1MVW-1MVW, 1XI4-1XI4, 3_1XI4-3_1XI4, and 1HTQ-1HTQ. Each case was tested on these GPUs using a similar testing procedure (and computing environment) to

Table 5.5: The two GPU architectures used for testing the scalability of the grid-querying method on different GPUs. The table lists per architecture, the number of processing cores offered, the processing clock speed, the global memory size, the memory clock speed, and the memory bandwidth.

	GTX 580	**GTX 980**
Cores	512	2048
GPU Clock (MHz)	772	1126
Memory (GB)	1.5	4
Mem. Clock (GB/sec)	2.004	7.0
Mem. Bandwidth (GB/sec)	192.4	224

Table 5.6: Structural information for the six molecules used for testing the scalability of the grid-querying method on different GPUs. 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)**
1AT1	21318	150.51
2YEY	53613	184.50
1MVW	94966	462.23
1XI4	183600	747.22
3_1XI4	550800	2113.67
1HTQ	978720	225.71

the one described in benchmarking; namely each test utilized a common 4×4 ligand repositioning matrix and arbitrary force parameters, allowed molecular overlapping, and recorded 10000 different response times. Again, the times account for all costs before, during and after kernel execution (i.e. prepare kernel on the CPU, submit and queue kernel for execution, execute kernel, transfer results to the CPU and compute the total force), and not only the cost to execute the kernel. In all cases the regular grids were constructed with c_g=5r_C (8.5Å), the cut-off distance applied was 8Å, and the number of interacting atom pairs returned from querying was slightly more than 20 thousands. The results reported per test case and GPU architecture are the averages of those 10000 responses. Figure 5.10 illustrates graphically these averages.

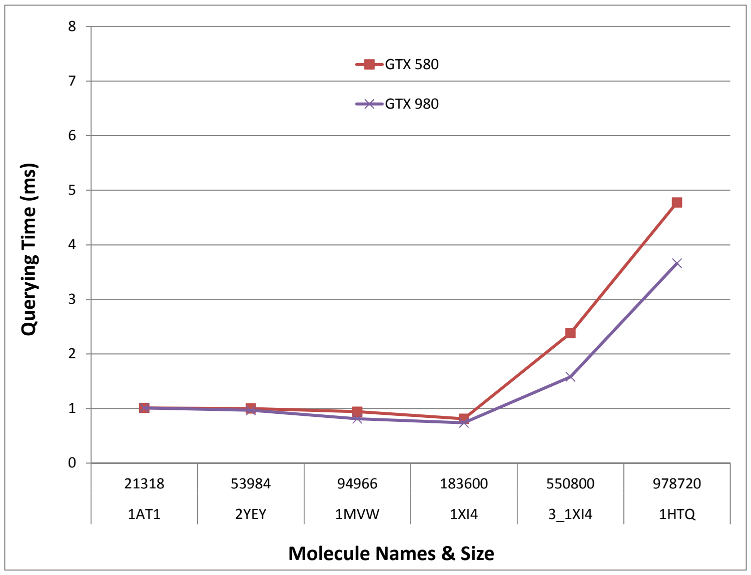


Figure 5.10: Measuring the scalability of the grid-querying method on different GPUs. The graph shows the averages of the 10000 response times recorded per test case and different GPU architecture. The number of interacting atom pairs returned were slightly more than 20K. All regular grids were of cell size $c_g = 5r_c$.

Despite its superior processing power the GTX 980 recorded similar query responses as the GTX 580, for molecular sizes up to 183.6K atoms. A close examination of these findings revealed that although the kernel execution times obtained by the GTX 980 was about 2 times faster than those of GTX 580, the overall response times of the GTX 980 were not significantly better than those of GTX 580, due to the following fixed costs: a) time to prepare the kernel for execution on the CPU, b)time to submit and queue kernel execution on the GPU, c) transfer force subtotals from the CPU to GPU, and d) compute the total force on the CPU. For smaller molecules these costs attributed to around 50% of the total querying cost, and as such averaged out the kernel-execution speed improvements obtained by the GTX 980. Another reason that might have affected negatively the GTX 980's performance, during the docking of molecules up to 183.6K atoms each, is the size of the workgroups used. Namely, the number of workgroups sent for execution in those cases was less than or equal to 718 (i.e. ceiling(183600/256)), meaning that the GTX 980 architecture was under-utilized, i.e. not enough work to utilize all of its processing cores. That might also explain the significant performance improvements (up to 30%) recorded on this GPU for molecules larger than 500K atoms. In those cases the assigned workload was large enough (i.e. $\geq$2149 workgroups) to benefit from the extra processing cores offered by the GTX 980, and at the same time overwhelm the processing capabilities of the GTX 580. This hypothesis was tested by conducting an additional docking simulation on the GTX 980, using the molecule 1XI4 (test case 1XI4-1XI4) and workgroups of size 128. In that case, the average querying times attained by the GTX 980 were about 22% better than the ones reported initially, i.e. they dropped from 0.75ms to 0.61ms. This indicates that the method would perform better if its kernel execution is fine-tuned based on the specifications (and hardware design) of each GPU. However, such fine-tuning defeats the purpose of using OpenCL in order to maximize the portability of Haptimol_RD to different GPU architectures, and as such it was not investigated any further and left outside the scope of this thesis. Overall, the results show that the proposed method would scale adequately with different/more powerful GPUs, as long as, the workload produced utilizes properly the additional processing power.

5.9 Conclusion

This chapter describes implementation details for two real-time, GPU-accelerated force calculation methods pertinent to interactive haptics-assisted docking. Both methods utilize 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 partitioning structures) in order to compute the interaction forces in real time. The methods adapt the CPU-based methods discussed in Chapter 4 on the GPU, and as such compute the forces only for those interacting atom pairs found within a cut-off distance. To minimize pre-computation requirements and achieve high levels of GPU occupancy, the methods construct the respective partitioning structures on the ligand only, and then query this structure for all receptor atoms in parallel. Since there are no spatial constraints on the receptor atoms (i.e. free to move during conformational changes), both approaches can support in principle receptor flexibility (provided conformational change can be computed sufficiently fast), which means that they can compute these forces at haptic refresh rates during receptor deformation.

These methods have been implemented and tested with docking simulations of different molecular shapes and sizes. In all test cases, the grid-based method performed consistently faster than the octree-based method. However, both of them achieved force updates in less than 2ms, which ranged from standard protein-drug to very large protein-protein interaction problems, i.e. with molecules consisting of hundreds of thousands of atoms each. For this reason the grid-based method is set as the default force calculation method in Haptimol_RD (when in GPU mode), and it is switched over automatically to the octree-based method when the available GPU memory cannot accommodate the construction of a regular grid. This forms a scalable, hybrid approach that can support (and adapt its execution to) different GPU architectures and facilitate the haptics-assisted study of very large protein-protein interactions (as proved with additional testing), while improving substantially upon its CPU-based counterpart. As shown however, force stability issues arise when the interacting molecules are very large and in close proximity, due to an erratic force profile both in magnitude and direction. The latter is attributed to the eventual enlargement of the set of atom pairs considered during force calculations. Force scaling and collision response techniques might be able to address this issue sufficiently. The following chapter describes such techniques and closes with a brief description of the software implementation.

Chapter 6

Force Scaling, Stability and Haptimol_RD

6.1 Introduction

The methods described in the previous two chapters compute the interaction forces in $kJ\ mol^{-1}\ nm^{-1}$ (see Section 3.4). It is therefore necessary that these forces are converted to Newtons (i.e. device units), and scaled appropriately prior to rendering them on the haptic device. This scaling is necessary in order to ensure that a good range of forces can be felt by the user through the device. If not done appropriately, the scaling can affect drastically the profile of the forces rendered (i.e. render only the weak, long-range attractive/repulsive interactions and not the strong, short-range repulsive ones, and vice versa), and as such decrease the effectiveness of the simulation. However, force scaling by itself, although crucial, does not ensure the haptic stability of the forces rendered. As seen in Section 5.5.3, the rigid docking of large biomolecules can introduce to the simulation forces that are erratic (fluctuate rapidly) both in magnitude and direction, causing device jittering (especially when the molecules are in contact). The stability of the haptic force feedback therefore is another factor that can affect negatively the user's experience during a docking simulation, if not addressed properly.

This chapter discusses novel intuitive methods that address sufficiently the force scaling and haptic stability issues pertinent to the interactive docking of large rigid biomolecules. An implementation overview of Haptimol_RD is also given here, outlining the application's main functionality. The discussion starts in the following section with the force scaling methods implemented in Haptimol_RD.

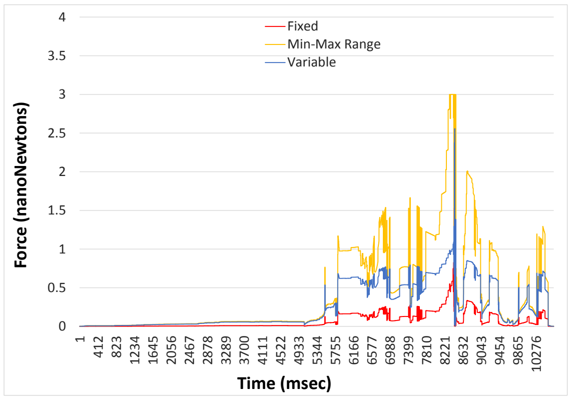
6.2 Force scaling

A haptic device has a finite force-rendering range (0-3.3 Newtons for the Geomagic Touch), whereas the intermolecular forces **f** can take a very large range of values. As such the mapping between the forces acting at the molecular level and the forces rendered to the user at the physical level must take into account these range differences. Failure to do so could hinder drastically the user's perception of the interaction forces, e.g. render perceivably repulsive VDW forces at close range but failing to render perceivably weaker electrostatic forces at longer ranges. To address this issue Haptimol_RD allows the user to select in real time amongst three different scaling methods, each of which is capable of altering the magnitude of the interaction force felt by the user during the simulation. The first method is a fixed scaling method, similar to the one proposed by Wollacott and Merz [WMJ07]. When applied, the user senses on the haptic device strong force magnitudes (approx. 3N) when the molecules are in collision, and weaker forces as the ligand moves away from the receptor. The second method is a new intuitive method introduced in this thesis, which scales the total interaction force by mapping it linearly to a user defined min/max range of force magnitudes. The min-max method enables the user to focus on a specific range of intermolecular forces during the simulation. As such, it can help the user perceive certain force ranges on the haptic device (e.g. weak long range attractive VDW or electrostatics), which would have been otherwise undetectable. When applied, all interaction forces greater than max are mapped to a haptic force of 3N and all forces less than min are capped to a haptic force of 0N. Unlike fixed scaling, the intensity of the interaction forces felt on the haptic device depends only on the range, and not on the relative position of the two molecules, e.g. if in collision, or a distance apart. The third method is the variable gain scaling method proposed by Bolopion et.al [BCRR11]. Their method amplifies small amplitude forces using a series of arctangent functions.

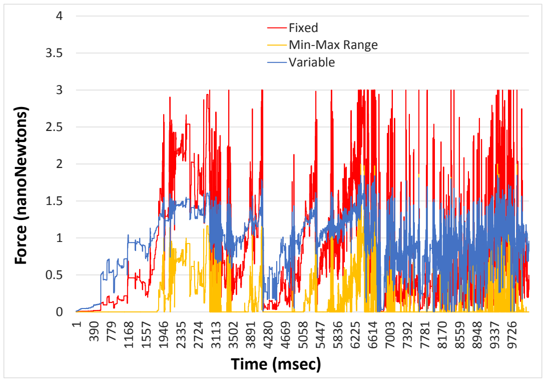
For the first two methods, Haptimol_RD converts **f** from $kJ\ mol^{-1}\ nm^{-1}$ to nanoNewtons (nN). Specifically, **f** is converted initially from $kJ\ mol^{-1}\ nm^{-1}$ to Newtons, via a division by 6.02329×10^{11} (since 1N is equivalent to 6.02329×10^{11} kJ $mol^{-1}\ nm^{-1}$), and the result is then scaled by 10^9. In fixed scaling mode, Haptimol_RD renders the result on the haptic device only if the resultant interaction force is less than or equal to 3nN (i.e. 1nN maps to a 1N haptic force). Otherwise it caps **f** at 3nN and renders the result. In min-max scaling mode, **f** is mapped to a user-defined scaling range using Equation 6.2.1, and the result is rendered on the haptic device. Equation 6.2.1 returns the force rendered on the device $\mathbf{f_h}$ which is given by,

$$\mathbf{f_h} = \begin{cases} \mathbf{0}, & \text{if } f \leq f^{min} \\ (f_h^{max})\,\hat{\mathbf{f}}, & \text{if } f \geq f^{max} \\ \left(\frac{(f - f^{min}) f_h^{max}}{f^{max} - f^{min}}\right)\hat{\mathbf{f}}, & \text{if } f^{min} \leq f \leq f^{max} \end{cases} \tag{6.2.1}$$

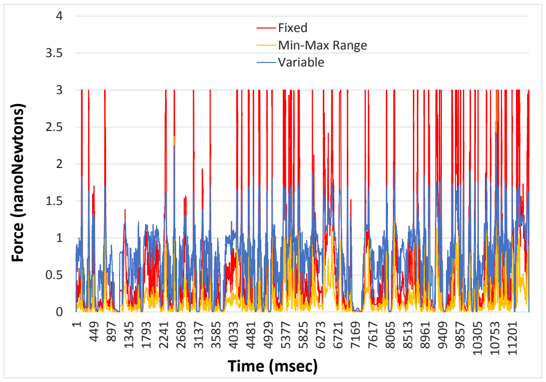
where $\hat{\mathbf{f}}$ is the unit vector in the direction of **f**, f^{max} and f^{min} are the upper and lower limits of the user defined range of interaction force magnitudes in nanoNewtons, f is the magnitude of **f**, and f_h^{max} is the magnitude of the maximum force exerted by the haptic device (3N in this case). The variable gain method does not require **f** to be



(a)



(b)



(c)

Figure 6.1: Graphing the interaction forces obtained after scaling, during three different rigid docking simulations of protein *GroES* and the receptor protein *GroEL*. Each simulation lasted for approximately 10 seconds and the force was scaled using the fixed, min-max range and variable gain methods. (a) The min-max range was set equal 0-0.5 nanoNewtons in order to scale up (focus the study on) the long-range interactions when the molecules are farther apart. (b) The min-max range was set equal 1-6 nanoNewtons in order to scale down the magnitude of the short-range, repulsive VDW interactions, and study structural complementarity close to the docking site. (c) The min-max range was set equal 0-10 nanoNewtons in order to smooth out the rapid force fluctuations rendered on the haptic device during the simulation.

converted from $kJ\ mol^{-1}\ nm^{-1}$ to Newtons.

In all scaling methods, the maximum force rendered on the haptic device is limited to 3N, i.e. $\mathbf{f_h} \leq$3N. Moreover, the user cannot perceive haptic forces less than or equal to 0.26N because they are masked by the back-drive friction of the haptic device.

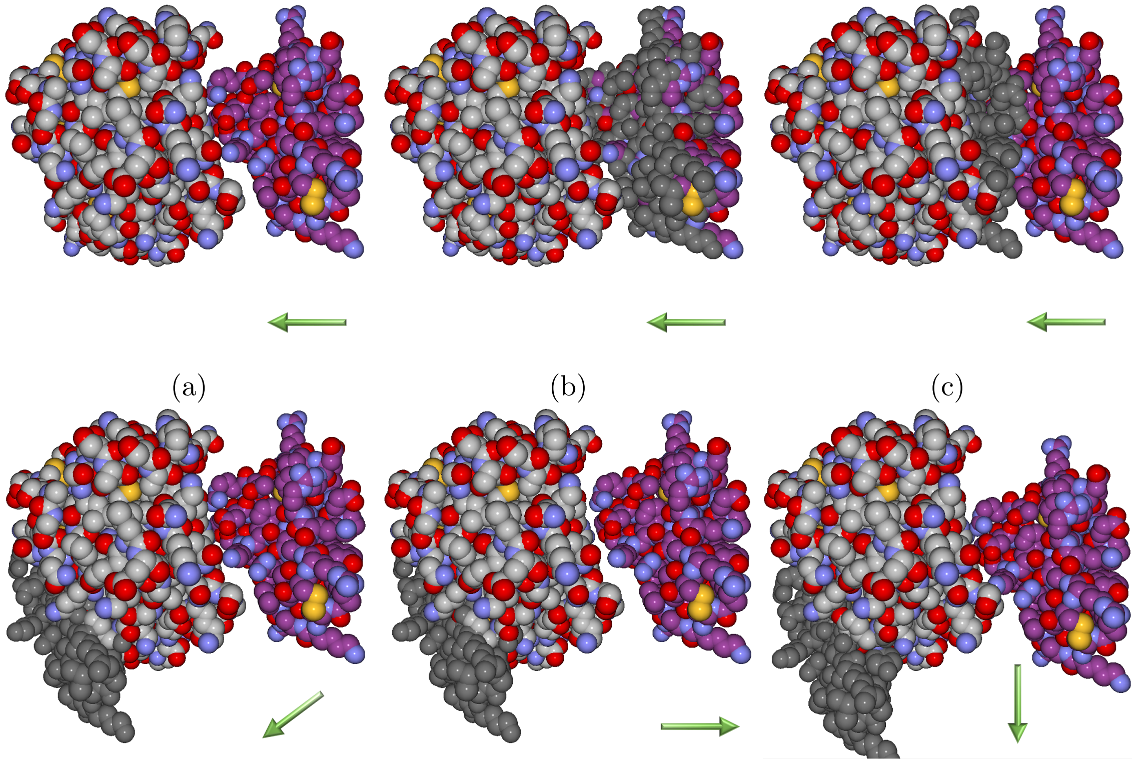
6.3 Haptic Stability and Multi-point Collision Response

In an interactive haptics-assisted docking application, attaining haptic-refresh rates and an appropriate scaling factor is crucial but does not guarantee force stability. Especially for those applications that model the VDW interaction forces using LJ force instabilities arise when the two molecules are in very close proximity with each other. When this occurs, device vibrations and jittering prohibit the user from perceiving the actual interaction forces. Atomic interpenetration exacerbates this issue substantially by making the interaction forces extremely erratic in both magnitude and direction, and causing the haptic stylus to move uncontrollably. The main reason for this instability is the rapid change in magnitude and force direction (e.g. from attractive to repulsive) of the VDW interactions. Haptimol_RD addresses atomic interpenetration by implementing an intuitive force-based multi-point collision response method which prevents extreme receptor/ligand atom overlapping. The method allows the application to update the interaction force and the ligand position only if the computed force satisfies certain criteria. Otherwise, the application keeps the ligand at its last valid position, and renders on the haptic device the last valid force (Figure 6.2). Unlike other approaches [LL04, BJ08], the method does not rely on penetration depths and uniform grids/distance maps in order to resolve ligand movement during collision, and as such it is free of any spatial constraints. Instead, ligand movement is resolved using the concept of relative movement as discussed in Section 3.6. Algorithm 10 outlines this method.

Algorithm 10 Force-based Collision Response

```
Require: $p_{curr}$, current HIP position
Require: $p_{last}$, last HIP position
Ensure: $f_{curr}$, the force rendered on the haptic device
1: $m_{tmp}^{Cursor} \leftarrow$ GetTmpRelativeCursorMovement($p_{curr}$, $p_{last}$)
2: $T_{tmp} \leftarrow$ GetTmpTransformationMatrix($m_{tmp}^{Cursor}$)
3: $p_{tmp}^{L} \leftarrow$ UpdateTmpLigandPosition($T_{tmp}$)
4: $f_{tmp} \leftarrow$ ComputeTmpForce($p_{tmp}^{L}$)
5: // check/set the force/position updating flags
6: if $f_{last} > 3nN$ then
7:     $flag_f \leftarrow$ true
8: else
9:     $flag_f \leftarrow$ false
10: end if
11: if $f_{tmp} \leq 3nN$ OR ($flag_f$ is true AND $f_{tmp} \leq f_{last}$) then
12:     update $\leftarrow$ true
13: else
14:     update $\leftarrow$ false
15: end if
16: if update is false then
17:     set all positions back to their last valid values
18:     $f_{curr} \leftarrow f_{last}$
19: else
20:     // set all temporary positions as current
21:     $p_{last} \leftarrow p_{curr}$
22:     $m_{curr}^{Cursor} \leftarrow m_{tmp}^{Cursor}$
23:     $T_{curr} \leftarrow T_{tmp}$
24:     $p_{curr}^{L} \leftarrow p_{tmp}^{L}$
25:     $f_{curr} \leftarrow f_{tmp}$)
26: end if
```

The main idea of Algorithm 10 is that ligand-position updates should occur only if the resultant forces are valid ($\leq$3nN), or converge to 3nN when invalid. Since the VDW repulsive interactions dwarf all other interactions when the atoms overlap (especially when multiple atoms are overlapping), the method guarantees that atom



(a) (b) (c) (d) (e) (f)

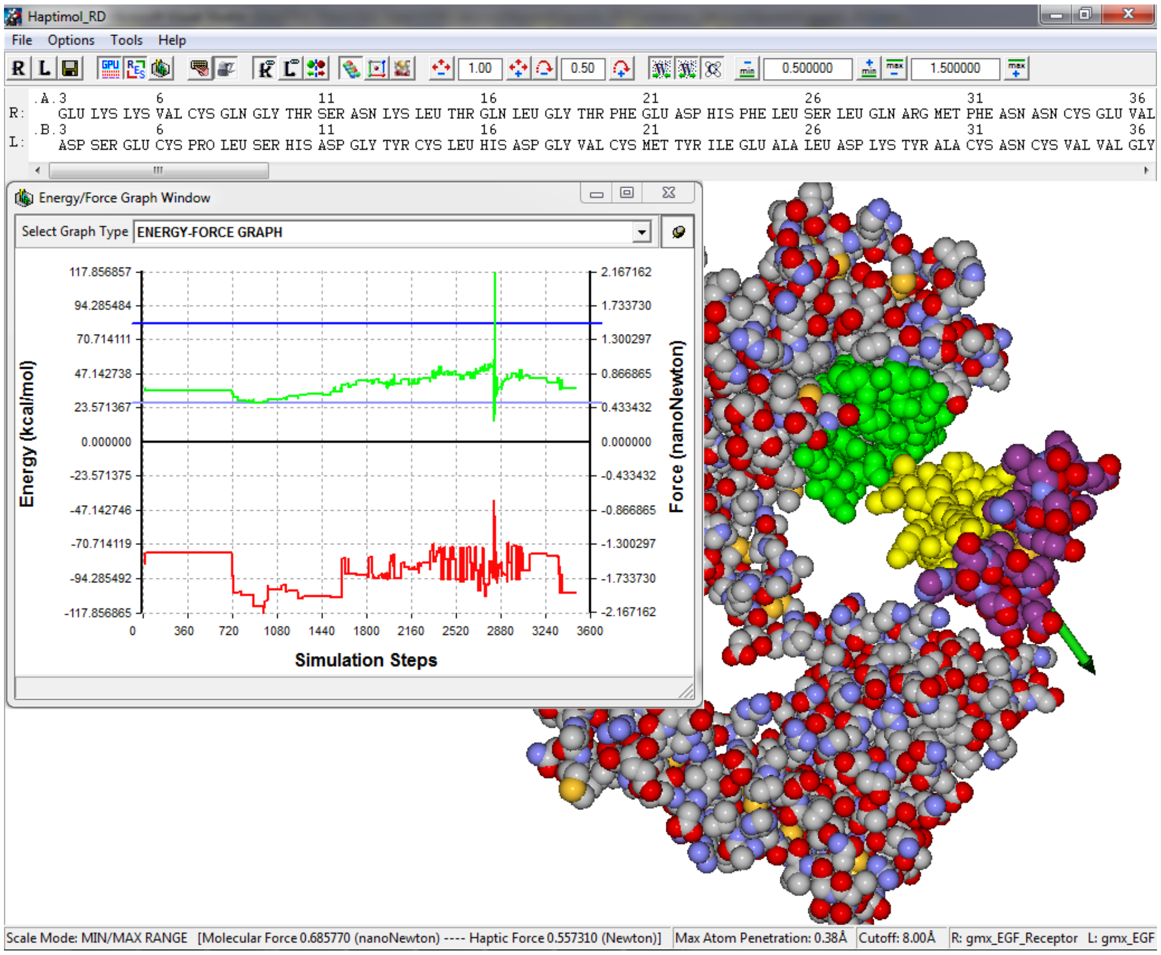
Figure 6.2: Applying the multipoint, force-based collision response method during a docking simulation. The ligand in grey colour is centred at the HIP position, and the ligand in purple colour (the actual ligand) shows the position of the virtual cursor (see Section 3.6). The green arrow at the bottom of each picture depicts the relative displacement of the HIP. During collision the HIP can be displaced without constraints, unlike the virtual cursor which must remain at its last valid (i.e. collision free) position. Collision occurs when the interaction force is greater than 3nN. a) The ligand moves towards the negative x axis without causing a collision. b,c) The molecules are in collision while the user keeps pushing the HIP (grey molecule) down the negative x axis. d) The user moves the HIP diagonally (along the negative x,y axes) while the molecules are still in collision. e) Relative HIP movement towards the positive x axis, results in a collision free movement for the virtual cursor and so the purple ligand molecule moves in the positive x direction. f) Again relative HIP movement towards the negative y axis, results in a collision free movement for the virtual cursor and so the purple ligand also moves in the same direction.

interpenetration is kept at acceptable levels. In addition, the method enables the user to experience the sliding of the ligand over the receptor, as the former moves over the latter during a multipoint collision event. In that case the user will sense the interaction force whilst being able to slide the ligand over the surface of the receptor. This sliding effect is important in haptics-assisted docking because it enables the user to explore structural complementarity between the molecules, like a 3D jigsaw puzzle. The force applied for collision response purposes is the same as the fixed-scaling force, with both VDW and electrostatic interactions accounted for. The 3nN was derived empirically and is the threshold that allowed Haptimol_RD to constrain, in all test cases, atom penetration at depths no deeper than 0.6A. A linear smoothing function [HS11] is also applied as a means to further reduce device jittering during penetration (necessary when the docking involves large proteins). Unlike the most popular approach in the field (proposed by Lee and Lyons [LL04] and improved by Wallcot and Merz [WMJ07]), this method does not have to alter the force profile of the docking simulation (i.e. add spring-based forces) in order to achieve haptic stability. Moreover it can attain a sliding effect for interacting molecules of very large sizes (colliding potentially at multiple/distributed points), without constraining the ligand to a probe sphere or to a drug-like molecule.

6.4 Implementation

Haptimol_RD (Figure 6.3) is developed using the Visual C++ programming language, the Windows Standard Development Kit (win SDK), and the OpenGL and OpenCL libraries. The win SDK and the OpenGL library were used for developing the Graphical User Interface, and for rendering/visualizing the 3D molecular structures respectively. The OpenCL library was used for programming the GPU to compute the interaction forces and it was chosen in order to maximize the portability of Haptimol_RD to different GPU architectures. Finally, the interface with the Geomagic Touch haptic device was implemented using the Open Haptics toolkit from Geomagic.

Haptimol_RD provides two modes of molecular visualization, as seen in Chapter 3. The first mode renders the molecule using a space-filling model, whereas the second uses a C_α backbone model. The user is allowed to select the two modes interchangeably at runtime. Haptic rendering of the interaction force is provided in either mode, with the force being computed, however, based on all interatomic interactions within the cut-off distance (regardless of the mode). A 3D force arrow, when enabled, allows the user to identify the direction and magnitude of the interaction force (green=weak, red=strong) at any point and time during the simulation. In addition to the visuo-haptic feedback, the application offers additional visual cues that can provide further assistance to the user during the docking simulation. These cues include a real-time graph of the interaction energy/force values and a residue colouring feature. The graph window provides a visual representation of the interaction energy/forces as the simulation progresses. This information can be used qualitatively by the user to identify potential local energy minima/barriers, and score/evaluate the respective conformations. With the residue colouring feature the user can colour-code different parts of the molecule (e.g. potential active sites), and use these codes to identify these parts readily during the docking simulation (reducing thus the search space). Residue selection and colouring is implemented in a manner similar to PyMol [Del02], and can be applied to both receptor and ligand molecules (Figure 6.3). A file-save feature allows the user to store the coordinates of potential docking conformations in different PDB files. Using this feature the user can export various docking poses, throughout the simulation, which can then import in other applications for further processing.



```
  .A.3          6              11             16             21             26             31             36
R:  GLU LYS LYS VAL CYS GLN GLY THR SER ASN LYS LEU THR GLN LEU GLY THR PHE GLU ASP HIS PHE LEU SER LEU GLN ARG MET PHE ASN ASN CYS GLU VAL
  .B.3          6              11             16             21             26             31             36
L:  ASP SER GLU CYS PRO LEU SER HIS ASP GLY TYR CYS LEU HIS ASP GLY VAL CYS MET TYR ILE GLU ALA LEU ASP LYS TYR ALA CYS ASN CYS VAL VAL GLY
```

Scale Mode: MIN/MAX RANGE [Molecular Force 0.685770 (nanoNewton) ---- Haptic Force 0.557310 (Newton)] | Max Atom Penetration: 0.38Å | Cutoff: 8.00Å | R: gmx_EGF_Receptor L: gmx_EGF

Figure 6.3: The Graphical User Interface of Haptimol_RD. The Epidermal Growth Factor (EGF) interacts with its receptor (EGFr), i.e. PDB code 1NQL. The interaction energy (red line) and force (green line) are displayed in real-time in the Energy/Force Graph Window. The dark and light blue lines within the same window depict the user-defined max and min limits of the force scaling range, respectively. The user can adjust this range during the simulation in real-time, and as such affect the profile of the forces rendered on the haptic device. In this case, the force is repulsive as visualized by the green force arrow. Using the residue selection/colouring control (the scrollable area above the Energy/Force Graph Window) the active sites of EGFr and EGF are coloured in green and yellow, respectively. The user utilizes this information in order to focus the haptic simulation in this region only, and thus reduce the search space of docking conformations substantially.

In addition to a haptics-assisted navigation, Haptimol_RD allows the user to conduct the docking simulation using a keyboard and a mouse. This mode offers the same level of control over the simulation as the haptics mode (including visual cues) except force feedback. This allows users who do not have access to a haptic device to still utilize the application in their studies. The lack of force feedback however, would have a negative impact on the usefulness of the application.

6.5 Conclusion

As stated in Chapter 2, computing the force at haptic refresh rates is a necessary condition but not the only one in haptics assisted docking. The other equally important condition is the stability of the forces rendered on the haptic device, especially when the two molecules interact at close proximity. Force scaling and collision response methods can help a haptic-based application attain stable force rendering by constraining the range of forces felt on the device, and by prohibiting atom interpenetration (and the display of the erratic forces obtained therein), respectively. The design and implementation of such methods is the main focus of this chapter. Three force scaling methods are examined. All methods scale the interaction force to nanoNewtons, using either a fixed scaler, a min-max range of force magnitudes, or a combination of arctangent functions. The resultant force in nanoNewtons is then mapped to a range of 0-3N of haptic force and rendered on the device. An intuitive, force-based collision response method is also described here capable of handling molecular collision at multiple contact points, and stabilizing the forces rendered on the haptic device when such collisions occur. These methods in combination with the 3D molecular visualization, haptic navigation and force calculation methods discussed earlier form Haptimol_RD, the first interactive haptics-assisted docking application that can accommodate the rigid docking of very large biomolecules, and the study of the underlying binding interactions.

Chapter 7

Conclusions

7.1 Discussion and Conclusions

Haptics can benefit the field of molecular docking by enabling the user intervene with the binding process using one's experience, knowledge and intuition. Nonetheless, this technology has not been adopted notably by the docking community, despite a forty-year research effort, due mainly to the strict force-update rate requirements of the device (i.e. within 1000Hz, but it can be relaxed down to 500Hz as noted in Otaduy and Lin [OL05a]), limitations on computing power, hardware costs, the lack of freely available haptics-assisted docking software, and constraints on the size of the molecules supported, e.g. small proteins and drugs. This thesis addressed these issues in an attempt to lift these barriers, and promote the use of haptics within the community. The result is a low-cost, free-to-download, user-friendly, interactive application called Haptimol_RD, that can facilitate the haptics-assisted, rigid docking of very large biomolecules, e.g. large proteins. Table 2.1 shows how Haptimol_RD sits in comparison to the state of the art in haptics-assisted docking. The thesis discusses the design and implementation of Haptimol_RD, and proposes novel methods pertinent to haptic navigation, force calculation, force scaling, and multipoint collision response. It also introduces future research directions and extensions to this work.

The discussion begins in Chapter 2 with a review on molecular docking, and on the advancements made in the field of haptics-assisted docking. It then continues in Chapter 3 with the design and development of Haptimol_RD's molecular visualization and haptic navigation routines. These are core functions of any interactive haptics-assisted docking system (and the first ones implemented in Haptimol_RD), since they enable the user to search for, identify and score visually the geometric complementarity between molecules, and cruise haptically the virtual world. Three graphics rendering techniques were examined here for molecular visualization. The method producing the best results both visually and performance-wise utilized an impostor-based ray tracing technique. The method preloaded and ray traced molecular structure directly on the GPU using GLSL, and achieved real-time rendering rates for molecules comprising close to two hundred thousand atoms each. The method however, by design, was prone to execution conflicts with other methods utilizing the same GPU unit. This became evident when the GPU-based force calculation method discussed in Chapter 5 was implemented in and tested on a single GPU configuration. The performance penalties inflicted on the force calculation method due to these conflicts were substantial, reaching up to 3ms. A dual GPU configuration would address this issue effectively, but it would have increased the overall cost of a standard laptop/desktop system. To keep this cost as low as possible it was decided to implement the second-best performing rendering technique, in Haptimol_RD, as the default one and leave the ray-tracing method for future use when a dual GPU configuration becomes commonplace. This technique renders the molecular structures using OpenGL and GLSL commands; namely, it describes a single atomic structure on the GPU, and then uses OpenGL and GLSL to populate, transform and render this structure into a molecule. The method achieves real-time rendering rates for molecules comprising up to forty thousand atoms each. Moreover it can accommodate molecular deformation easily since the atomic coordinates are fed to the graphics card at every rendering frame (and can be modified as such), unlike the ray tracing approach which uploads these coordinates on the GPU at startup. To navigate efficiently these large structures within their virtual environment, a fully decoupled version of the navigation cube concept proposed by Stocks et. al. [SHL09] was developed. The proposed navigation method introduces the concepts of a VHW (Virtual Haptic Workspace) and virtual cursor, and, unlike the original approach, utilizes two navigation cubes (i.e. one for the device workspace the second for the VHW) instead of one. The first navigation cube (mapped to the device workspace) translates HIP displacement into a position/rate control movement [SB08] for the virtual cursor, whereas the second cube (mapped to the VHW) translates virtual cursor displacement into a position/rate control movement for the virtual object (i.e. ligand) and VHW. Using this double layer of movement management, the method decouples HIP and virtual cursor movement, and attains unconstrained object/VHW movement within the virtual world (unlike the virtual coupling approach [BJ08] which constrains HIP and cursor movement using a spring). The multipoint collision response method described in Chapter 6 relies on this decoupling in order to handle molecular collision during a docking simulation.

The molecular visualization and haptic navigation routines formed the basis for the next two main pieces of work, relating to the calculation of the interaction forces. As seen in Chapter 2, a major issue in haptics-assisted docking is the 1-2ms force-update constraint, required for smooth and stable force-feedback. Current interactive approaches achieve such refresh rates by utilizing precomputed force grids and linear interpolation to accelerate the respective computations. However, these methods are limited to the docking of molecules comprising up to a couple of thousand of atoms each. Moreover as noted before, precomputed-grids are memory hungry, induce rough force transitions at cell boundaries, and by design cannot model molecular flexibility. To address these issues two ways for calculating the force were explored. The first approach computed the force by accounting for all interatomic interactions between the receptor and ligand molecules (i.e. the *Brute Force* approach), whereas the second approach utilized a set-reduction technique to reduce the number of interatomic interactions accounted for in Equation 2.2.8 and accelerate significantly the respective force computations. Performance tests on the Brute Force approach (on CPU and GPU) however, indicated that this approach is unsuitable for the interactive docking of large molecules due to its high execution complexity (see Sections 4.2 and Sections 5.3); leaving thus the set-reduction technique as the only alternative. Using a cut-off distance, the set-reduction technique relies on spatial partitioning structures (i.e. regular grids and octrees) and proximity querying algorithms (on those structures) in order to identify, at haptic refresh rates, the set of interacting atom-pairs within the cut-off. Force calculations are then executed using this set. Based on this concept, two novel force calculation approaches (the first optimized for the CPU the second for the GPU) were developed that can facilitate the haptics-assisted docking of large biomolecules.

The CPU-based force calculation approach described in Chapter 4 was the first approach investigated and implemented in Haptimol_RD. Two proximity querying algorithms were examined during this investigation, the first one utilized regular grids, the second octrees. Performance measurements taken on both querying methods showed that the octree-based algorithm outperformed the grid-based algorithm consistently. Using octree-based querying, a force calculation method was developed that overcomes the computational limitations of previous CPU-based approaches (i.e. utilizing pre-computed force grids), and can be applied to the haptic-assisted docking of rigid and flexible structures (providing conformational change can be computed sufficiently fast). When applied to rigid docking, the approach can facilitate larger molecular structures than the ones reported in previous studies (i.e. more than a threefold increase), and it is therefore not constrained to small proteins. Since tree construction times are irrelevant in rigid docking, the approach can maintain haptic force refresh rates on the CPU for molecular pairs comprising 7K atoms each. In flexible docking, however, the method would have to construct the trees at each haptic frame, and as such, haptic refresh rates could be achieved only for molecular pairs of up to one 1.7k each. Preliminary work on the GPU indicated that high-end GPU technology could benefit this approach substantially, both in the size of molecules supported and the force response times attained. This belief stemmed from the observation that the GPU-based implementation of the brute force approach outperformed the CPU-based implementation by almost sixfold.

To investigate this systematically, the grid-based and octree-based proximity querying methods were transferred and tested on the GPU. Chapter 5 details this work, and presents a hybrid GPU-accelerated force calculation approach that can facilitate effectively the interactive haptics-assisted study of very large biomolecules, i.e. the docking of molecules comprising hundreds of thousands of atoms each. 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 partitioning structures. The selection of the space partitioning structure used (i.e. a regular grid or an octree) is made at runtime, based on the available GPU memory. However, GPU memory does not seem to be an issue for the grid-based method, since 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 512MB memory), or (b) the GPU performs at the same time other memory hungry tasks (e.g. ray tracing, texture mapping), the octree-based method would represent an effective alternative. When compared to the CPU-based force calculation method, the GPU-based approach was up to 90 times faster. Moreover, 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. Again, providing the new positions of the receptor atoms are calculated sufficiently quickly, receptor flexibility could be modelled. For ligand flexibility however, the approach would have to construct the respective partitioning structure on the CPU and then transfer it on the GPU, at sub-millisecond times, as the ligand deforms. Because of this, sub-millisecond grid and octree construction times can be achieved by this approach for ligand molecules comprising up to 3.5k and 1.7k atoms respectively.

Given the size of molecules supported by Haptimol_RD, several issues arose related to force scaling and stability. Chapter 6 describes the methods developed during this thesis in order to address these issues, outlines the implementation of Haptimol_RD, and discusses the limitations of this docking application using real docking examples. Three force scaling techniques were examined and implemented within Haptimol_RD. The first scales the interactive force using a fixed scaling factor, the second using a user defined min/max range of force magnitudes, and the third using a series of arctangent functions. To address force stability (especially at close proximity), atom interpenetration was addressed by implementing an intuitive force-based multi-point collision response method which disallows extreme receptor/ligand atom overlapping. The method allows the application to update the interaction force and the ligand position only if the computed force satisfies certain criteria. With the force scaling and multipoint collision response methods implemented, Haptimol_RD can attain sufficient force stability during docking simulations of large molecules.

In conclusion, this thesis describes the implementation of an interactive haptics-assisted docking system, capable of docking very large rigid biomolecules. The system computes in real-time the electrostatic and VDW forces in docking, using cut-off-based proximity querying algorithms optimized for CPU/GPU-based execution. These methods overcome the issues of pre-computed force grids (section 2.2.3 discusses these issues), and allow the system to achieve force updates, at haptic refresh rates, for examples of interest in protein-protein and protein-drug docking. In its current implementation Haptimol_RD can facilitate the docking of molecules comprising up to forty thousand atoms each (even though it attains haptic force updates for molecules comprising up to 200k atoms each), due to constraints imposed by the application's molecular visualization routine (when run on one GPU) and the cost of acquiring a second GPU unit. Within a system with a dual GPU configuration however, Haptimol_RD can easily increase these sizes by fivefold. Because of the cut-off distance, it is expected that any inaccuracy in the force outcome will arise from the longer range electrostatic interactions rather than from the VDW. To test this, a docking experiment was conducted with BPTI on the receptor trypsin, and all atom pairs accounted for during the force calculation (brute force). However, no perceptible difference could be found on the forces rendered on the haptic device, even though in MD simulations variation of the cut-off distance can have significant effects on the outcome. 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. Even though the focus was on rigid docking, the methods discussed here can be applied without any modification to docking problems that model molecular flexibility. This was one of the design choices made during this thesis, with the size of molecules supported being the second one.

Haptics-assisted docking enables intuitive and interactive exploration of binding poses which may give an advantage over automated approaches. As this work has demonstrated, the inherent execution parallelism of GPUs can benefit haptics-assisted docking systems by allowing them to accommodate larger structures than before, and as such improve the applicability and usefulness of such systems. 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.

7.2 Future Work

The results presented in this thesis suggest interesting research directions to the field, including several extensions to this work. These are discussed in the following five paragraphs.

7.2.1 Molecular Flexibility

The GPU-based force calculation approach discussed here does not impose any pre-computational requirements (i.e. a spatial partitioning structure) on the receptor, and as such can facilitate force calculations for deformable receptors without any additional modifications. One approach already taken in haptics for modelling protein flexibility is to use an elastic network model [SLH11]. Using this model and the GPU-based force method, receptor flexibility can be achieved as follows: a) identify the set of interacting atom pairs, b) use the network model to deform the receptor based on this set of interatomic interactions, and c) compute the total interaction force after the receptor deformation. Modelling of both ligand and receptor flexibility can also be done on the GPU using MD simulation, given that the spatial decomposition of the ligand is performed within micro seconds (see next paragraph). Using the MD trajectories, the method of linear response [IUSK05] can be applied to calculate conformational change, on both receptor and ligand, due to interaction forces at a cut-off distance. A feature of protein dynamics (i.e. atom motions occur in a very reduced dimensional space) can also be used to reduce the number of calculations within the linear response method with minimal and quantifiable sacrifice in accuracy.

7.2.2 Grid/Octree Construction on the GPU

Existing octree [Kar12] and regular grid [GPBG11] subdivision methods suggest that sub-millisecond construction times can be achieved for very large molecules, if the construction occurs directly on the GPU. Investigating this possibility is the next logical step, since it will lift the size constraints currently imposed on the ligand (because of the CPU-based construction), and thus enable Haptimol_RD to support ligand flexibility for molecules comprising more than 3.5K atoms. The construction of the spatial partitioning structures on the GPU might also allow the exploration of real-time ray tracing techniques capable of enhancing the 3D perception of the molecules through shadows and illumination. Given that the molecules will not change shape significantly between haptic frames the possibility of exploiting coherence during deformation and proximity querying is another interesting question that necessitates further investigation.

7.2.3 Torques

During a docking simulation, torques affect ligand rotation around its center of mass and ideally they should be accounted for. The force calculation methods discussed here could be modified to compute the torques, with negligible computational overhand. 6DoF rendering techniques [Tri04, BJ08] could then be investigated in order to render these results on a 6DoF haptic device, and 6-DoF multipoint collision response methods could be developed in order to account for torques during molecular collision. Lower cost and more commonly used 3DOF haptic devices do not render torques and therefore ways to depict them graphically could also be explored.

7.2.4 Real time rendering of the Surface model

The graphical representation in Haptimol_RD shows either the VDW surface or the backbone of the molecule. However, none of these depictions is ideal for showing clearly cavities and depressions which are often associated with binding sites. These features are shown by the solvent excluded surface (SES), which is a surface traced by a water molecule, i.e. the surface model. Real-time rendering of the SESs of the flexible molecules may show the appearance of complementary binding features providing a strong visual cue of a possible binding pose. Although the molecular surface is drawn in molecular graphics software such as Pymol, it is slow and the challenge is to develop methods for rendering the SES within 30 ms for large deformable biomolecules. One direction worth investigating, is the rendering of the

SES only for areas near to the viewer or in close proximity to the other molecule, instead of for the entire structure. A parallel implementation of the occlusion-culling technique proposed by Hao et. al. [HVS04] could be a step towards that direction, enabling the real-time rendering of the SES for large structures. This might also allow Haptimol_RD to visualize on a single GPU larger molecules than the ones currently supported, due to the relatively small number of triangles rendered.

Bibliography

[ACL96] Norman L Allinger, Kuohsiang Chen, and Jenn-Huei Lii. An improved force field (mm4) for saturated hydrocarbons. *Journal of Computational Chemistry*, 17(5-6):642–668, 1996.

[ADPH11] Apostolos Axenopoulos, Petros Daras, Georgios Papadopoulos, and Elias Houstis. A shape descriptor for fast complementarity matching in molecular docking. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(6):1441–1457, 2011.

[AGB13] Athanasios Anthopoulos, Ian Grimstead, and Andrea Brancale. Gpu-accelerated molecular mechanics computations. *J. Comput. Chem.*, 34(26):2249–2260, 2013.

[AGO08] Adriano D Andricopulo, Rafael VC Guido, and Glaucius Oliva. Virtual screening and its integration with modern drug design technologies. *Current medicinal chemistry*, 15(1):37–46, 2008.

[AJL$^+$08] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. Molecular biology of the cell. *New York: Garland Science*, 1, 2008.

[And03] Amy C Anderson. The process of structure-based drug design. *Chemistry & biology*, 10(9):787–797, 2003.

[APGB14] Athanasios Anthopoulos, Gaia Pasqualetto, Ian Grimstead, and Andrea Brancale. Haptic-driven, interactive drug design: implementing a gpu-based approach to evaluate the induced fit effect. *Faraday Discuss.*, 169:323–342, 2014.

[AT01] Ruben Abagyan and Maxim Totrov. High-throughput docking for lead generation. *Current Opinion in Chemical Biology*, 5(4):375–382, 2001.

[BAT11] Petter Bivall, Shaaron Ainsworth, and Lena AE Tibell. Do haptic representations help complex molecular learning? *Science Education*, 95(4):700–719, 2011.

[BBO$^+$83] Bernard R Brooks, Robert E Bruccoleri, Barry D Olafson, S Swaminathan, Martin Karplus, et al. Charmm: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.*, 4(2):187–217, 1983.

[BBZW04] Stefan Birmanns, Maik Boltes, Hwrwig Zilken, and Willy Wriggers. Adaptive visuo-haptic rendering for hybrid modeling of macromolecular assemblies. In *Proceedings Mechatronics and Robotics*, volume 4, pages 1351–1356, 2004.

[BCRR11] Aude Bolopion, Barthelemy Cagneau, Stephane Redon, and Stéphane Régnier. Variable gain haptic coupling for molecular simulation. In *World Haptics Conference (WHC), 2011 IEEE*, pages 469–474. IEEE, 2011.

[BH05] Jerome Baudry and Paul J Hergenrother. Structure-based design and in silico virtual screening of combinatorial libraries. a combined chemical-computational project. *Journal of chemical education*, 82(6):890, 2005.

[Biv10] Petter Bivall. *Touching the Essence of Life: Haptic Virtual Proteins for Learning.* PhD thesis, Linköping University, Linköping, Sweden, 2010.

[BJ08] Jernej Barbic and Doug L James. Six-dof haptic rendering of contact between geometrically complex reduced deformable models. *Haptics, IEEE Transactions on*, 1(1):39–52, 2008.

[BJOYBJK90] Frederick P Brooks Jr, Ming Ouh-Young, James J Batter, and P Jerome Kilpatrick. Project grope - haptic displays for scientific visualization. In *ACM SIGGraph computer graphics*, volume 24, pages 177–185. ACM, 1990.

[BK03] Natasja Brooijmans and Irwin D Kuntz. Molecular recognition and docking algorithms. *Annual review of biophysics and biomolecular structure*, 32(1):335–373, 2003.

[Bon64] A_ Bondi. van der waals volumes and radii. *The Journal of physical chemistry*, 68(3):441–451, 1964.

[BS02] Cagatay Basdogan and Mandayam A Srinivasan. Haptic rendering in virtual environments. *Handbook of virtual environments*, pages 117–134, 2002.

[BSA01] O Burchan Bayazit, Guang Song, and Nancy M Amato. Ligand binding with obprm and user input. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 954–959. IEEE, 2001.

[BW03] Stefan Birmanns and Willy Wriggers. Interactive fitting augmented by force-feedback and virtual reality. *Journal of structural biology*, 144(1):123–131, 2003.

[BWF$^+$00] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, TN Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic Acids Res.*, 28(1):235–242, 2000.

[CH88] Homer H Chen and Thomas S Huang. A survey of construction and manipulation of octrees. *Computer Vision, Graphics, and Image Processing*, 43(3):409–431, 1988.

[chi] chimera. chimera: User's guide.

[Con83a] Michael L Connolly. Analytical molecular surface calculation. *Journal of Applied Crystallography*, 16(5):548–558, 1983.

[Con83b] Michael L Connolly. Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 221(4612):709–713, 1983.

[CPN10] Peter Csermely, Robin Palotai, and Ruth Nussinov. Induced fit, conformational selection and independent dynamic segments: an extended view of binding events. *Trends in biochemical sciences*, 35(10):539–546, 2010.

[CWL12] Christopher R Corbeil, Christopher I Williams, and Paul Labute. Variability in docking success rates due to dataset preparation. *Journal of computer-aided molecular design*, 26(6):775–786, 2012.

[DAMR07] B Daunay, A Abbaci, A Micaelli, and S Regnier. The wave variables, a solution for stable haptic feedback in molecular docking simulations. In *Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications*, pages 67–73. Springer, 2007.

[DDKA06] Christian Duriez, Frederic Dubois, Abderrahmane Kheddar, and Claude Andriot. Realistic haptic rendering of interacting deformable objects in virtual environments. *Visualization and Computer Graphics, IEEE Transactions on*, 12(1):36–47, 2006.

[Del02] W. L. Delano. The PyMOL Molecular Graphics System, 2002.

[DFTRJ97] Wolfgang Damm, Antonio Frontera, Julian Tirado-Rives, and William L Jorgensen. Opls all-atom force field for carbohydrates. *Journal of Computational Chemistry*, 18(16):1955–1970, 1997.

[DJMH05] R Andrew Davies, Nigel W John, John N MacDonald, and Keith H Hughes. Visualization of molecular quantum dynamics: a molecular visualization tool with integrated web3d and haptics. In *Proceedings of the tenth international conference on 3D Web technology*, pages 143–150. ACM, 2005.

[DMR07a] B. Daunay, A. Micaelli, and S. Régnier. 6 dof haptic feedback for molecular docking using wave variables. In *Actes de ICRA'07 IEEE International Conference on Robotics and Automation*, pages 840–845, Rome, Italie, April 2007.

[DMR07b] Bruno Daunay, Alain Micaelli, and Stéphane Régnier. Energy-field reconstruction for haptic-based molecular docking using energy minimization processes. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2704–2709. IEEE, 2007.

[DR09] Bruno Daunay and Stephane Régnier. Stable six degrees of freedom haptic feedback for flexible ligand–protein docking. *Computer-Aided Design*, 41(12):886–895, 2009.

[Eas13] Robert Easdon. Ambient occlusion and shadows for molecular graphics. Master's thesis, University of East Anglia, Norwich, UK, 11 2013.

[EKK04] Miriam Eisenstein and Ephraim Katchalski-Katzir. On proteins, grids, correlations, and docking. *C. R. Biol.*, 327(5):409–420, May 2004.

[far14] Computing power revolution and new algorithms: Gp-gpus, clouds and more: general discussion. *Faraday Discuss.*, 169:379–401, 2014.

[FDGB08] Nicolas Férey, Olivier Delalande, Gilles Grasseau, and Marc Baaden. A vr framework for interacting with molecular simulations. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 91–94. ACM, 2008.

[FHJL88] Thomas E Ferrin, Conrad C Huang, Laurie E Jarvis, and Robert Langridge. The midas display system. *Journal of Molecular Graphics*, 6(1):13–27, 1988.

[FNM$^+$09] Nicolas Férey, Julien Nelson, Christine Martin, Lorenzo Picinali, Guillaume Bouyer, A Tek, Patrick Bourdot, Jean-Marie Burkhardt, Brian FG Katz, Mehdi Ammi, et al. Multisensory vr interaction for protein-docking in the corsaire project. *Virtual Reality*, 13(4):273–293, 2009.

[FP93] T-P Fang and Les A Piegl. Delaunay triangulation using a uniform grid. *Computer Graphics and Applications, IEEE*, 13(3):36–47, 1993.

[GME$^+$00] Arthur Gregory, Ajith Mascarenhas, Stephen Ehmann, Ming Lin, and Dinesh Manocha. Six degree-of-freedom haptic display of polygonal models. In *Proceedings of the conference on Visualization'00*, pages 139–146. IEEE Computer Society Press, 2000.

[GPBG11] Kirill Garanzha, Simon Premože, Alexander Bely, and Vladimir Galaktionov. Grid-based sah bvh construction on a gpu. *The Visual Computer*, 27(6-8):697–706, 2011.

[HDS96] William Humphrey, Andrew Dalke, and Klaus Schulten. Vmd: visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33–38, 1996.

[HKR12] Lennart Heinzerling, Robert Klein, and Matthias Rarey. Fast force field-based optimization of protein–ligand complexes with graphics processor. *J. Comput. Chem.*, 33(32):2554–2565, 2012.

[HN96] Thomas A Halgren and Robert B Nachbar. Merck molecular force field. iv. conformational energies and geometries for mmff94. *Journal of Computational Chemistry*, 17(5-6):587–615, 1996.

[HS10] Xiyuan Hou and Olga Sourina. Haptic rendering algorithm for biomolecular docking with torque force. In *Cyberworlds (CW), 2010 International Conference on*, pages 25–31. IEEE, 2010.

[HS11] Xiyuan Hou and Olga Sourina. Six degree-of-freedom haptic rendering for biomolecular docking. In *Transactions on computational science XII*, pages 98–117. Springer, 2011.

[HVS04] Xuejun Hao, Amitabh Varshney, and Sergei Sukharev. Real-time visualization of large time-varying molecules. In *Proceedings of the High-Performance Computing Symposium*, volume 4. Citeseer, 2004.

[IHL14] Georgios Iakovou, Steven J Hayward, and Stephen D Laycock. Fd169: A real-time proximity querying algorithm for haptic-based molecular docking. *Faraday Discussions*, 2014.

[IHL15] Georgios Iakovou, Steven Hayward, and Stephen D Laycock. Adaptive gpu-accelerated force calculation for interactive rigid molecular docking using haptics. *Journal of Molecular Graphics and Modelling*, 61:1–12, 2015.

[ILH16] Georgios Iakovou, Stephen Laycock, and Steven Hayward. Determination of locked interfaces in biomolecular complexes using Haptimol_RD. *Biophysics and Physicobiology*, 2016.

[IUSK05] Mitsunori Ikeguchi, Jiro Ueno, Miwa Sato, and Akinori Kidera. Protein structural change upon ligand binding: linear response theory. *Physical review letters*, 94(7):078102, 2005.

[JBT04] Doug L James, Jernej Barbič, and Christopher D Twigg. Squashing cubes: Automating deformable model construction for graphics. In *ACM SIGGRAPH 2004 Sketches*, page 38. ACM, 2004.

[Jmo] Jmol. Jmol: File formats/coordinates.

[JMTR96] William L Jorgensen, David S Maxwell, and Julian Tirado-Rives. Development and testing of the opls all-atom force field on conformational energetics and properties of organic liquids. *Journal of the American Chemical Society*, 118(45):11225–11236, 1996.

[KAR$^+$94] Ronald Knegtel, J Antoon, C Rullmann, Rolf Boelens, and Robert Kaptein. Monty: a monte carlo approach to protein-dna recognition. *Journal of molecular biology*, 235(1):318–324, 1994.

[Kar12] Tero Karras. Maximizing parallelism in the construction of bvhs, octrees, and k-d trees. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics conference on High-Performance Graphics*, pages 33–37. Eurographics Association, 2012.

[KBO$^+$82] Irwin D Kuntz, Jeffrey M Blaney, Stuart J Oatley, Robert Langridge, and Thomas E Ferrin. A geometric approach to macromolecule-ligand interactions. *Journal of molecular biology*, 161(2):269–288, 1982.

[KDFB04] Douglas B Kitchen, Hélène Decornez, John R Furr, and Jürgen Bajorath. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nature reviews Drug discovery*, 3(11):935–949, 2004.

[Kre01] Aleš Krenek. Haptic rendering of molecular conformations. In *Proceedings of Eurohaptics*, pages 142–145, 2001.

[KSL05] EM Krovat, T Steindl, and T Langer. Recent advances in docking and scoring. *Current Computer-Aided Drug Design*, 1(1):93–102, 2005.

[KZD$^+$14] Max Kaluschke, Uwe Zimmermann, Marinus Danzer, Gabriel Zachmann, and Rene Weller. Massively-parallel proximity queries for point clouds. In *Workshop on Virtual Reality Interaction and Physical Simulation*, pages 19–28. The Eurographics Association, 2014.

[LA91] Jenn-Huei Lii and Norman L Allinger. The mm3 force field for amides, polypeptides and proteins. *Journal of computational chemistry*, 12(2):186–199, 1991.

[LL04] Yong-Gu Lee and Kevin W Lyons. Smoothing haptic interaction using molecular force calculations. *Computer-Aided Design*, 36(1):75–90, 2004.

[LMM10] Christian Lauterbach, Qi Mo, and Dinesh Manocha. gproximity: Hierarchical gpu-based operations for collision and distance queries. In *Computer Graphics Forum*, volume 29, pages 419–428. Wiley Online Library, 2010.

[LR71] Byungkook Lee and Frederic M Richards. The interpretation of protein structures: estimation of static accessibility. *Journal of molecular biology*, 55(3):379–IN4, 1971.

[LR96] Thomas Lengauer and Matthias Rarey. Computational methods for biomolecular docking. *Current opinion in structural biology*, 6(3):402–406, 1996.

[LSP06] Andrew R Leach, Brian K Shoichet, and Catherine E Peishoff. Prediction of protein-ligand interactions. docking and scoring: successes and gaps. *J. Med. Chem.*, 49(20):5851–5855, 2006.

[LYL05] Susana K Lai-Yuen and Yuan-Shin Lee. Computer-aided molecular design (camd) with force-torque feedback. In *Computer Aided Design and Computer Graphics, 2005. Ninth International Conference on*, pages 199–204. IEEE, 2005.

[LYL06a] Susana K Lai-Yuen and Yuan-Shin Lee. Energy-field optimization and haptic-based molecular docking and assembly search system for computer-aided molecular design (camd). In *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2006 14th Symposium on*, pages 233–240. IEEE, 2006.

[LYL06b] Susana K Lai-Yuen and Yuan-Shin Lee. Interactive computer-aided design for molecular docking and assembly. *Computer-Aided Design and Applications*, 3(6):701–709, 2006.

[Mar02] Eric Martz. Protein explorer: easy yet powerful macromolecular visualization. *Trends in Biochemical Sciences*, 27(2):107–109, 2002.

[MCET05] Ross Maciejewski, Seungmoon Choi, David S Ebert, and Hong Z Tan. Multi-modal perceptualization of volumetric data and its application to molecular docking. In *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint*, pages 511–514. IEEE, 2005.

[MEL$^+$08] N Moitessier, P Englebienne, D Lee, J Lawandi, Corbeil, and CR. Towards the development of universal, fast and highly accurate docking/scoring methods: a long way to go. *Br. J. Pharmacol.*, 153(S1):S7–S26, 2008.

[MFC$^+$14] Anne-Elisabeth Molza, Nicolas Férey, Mirjam Czjzek, Elisabeth Le Rumeur, Jean-François Hubert, Alex Tek, Benoist Laurent, Marc Baaden, and Olivier Delalande. Innovative interactive flexible docking method for multi-scale reconstruction elucidates dystrophin molecular assembly. *Faraday Discuss.*, 169:45–62, 2014.

[MHL$^+$09] Garrett M Morris, Ruth Huey, William Lindstrom, Michel F Sanner, Richard K Belew, David S Goodsell, and Arthur J Olson. Autodock4 and autodocktools4: Automated docking with selective receptor flexibility. *Journal of computational chemistry*, 30(16):2785–2791, 2009.

[MPT05] William A McNeely, Kevin D Puterbaugh, and James J Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *ACM SIGGRAPH 2005 Courses*, page 42. ACM, 2005.

[MR01] Edward T Maggio and Kal Ramnarayan. Recent developments in computational proteomics. *Drug discovery today*, 6(19):996–1004, 2001.

[MWS96] Michael Meyer, Peter Wilson, and Dietmar Schomburg. Hydrogen bonding and molecular surface shape complementarity as a basis for protein docking. *Journal of molecular biology*, 264(1):199–210, 1996.

[Neu97] Arnold Neumaier. Molecular modeling of proteins and mathematical prediction of protein structure. *SIAM review*, 39(3):407–460, 1997.

[NMT02] Hiroshi Nagata, Hiroshi Mizushima, and Hiroshi Tanaka. Concept and prototype of protein–ligand docking simulator with force feedback technology. *Bioinformatics*, 18(1):140–146, 2002.

[NVI] NVIDIA. Cuda toolkit 3.1.

[NVI10] NVIDIA. *NVIDIA OpenCL Best Practices Guide.* NVIDIA, https://developer.nvidia.com/cuda-toolkit-32-downloads, 2010.

[OL05a] Miguel A Otaduy and Ming C Lin. Introduction to haptic rendering. In *ACM SIGGRAPH 2005 Courses*, page 3. ACM, 2005.

[OL05b] Miguel A Otaduy and Ming C Lin. Sensation preserving simplification for haptic rendering. In *ACM SIGGRAPH 2005 Courses*, page 72. ACM, 2005.

[OM12] Khronos Opencl and Aaftab Munshi. The opencl specification version: 1.2 document revision: 19, 2012.

[OY90] Ming Ouh-Young. *Force Display in Molecular Docking.* PhD thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 2 1990. TR90-004.

[PCT+07] Petter Bivall Persson, Matthew D Cooper, Lena AE Tibell, Shaaron Ainsworth, Anders Ynnerman, and B-H Jonsson. Designing and evaluating a haptic system for biomolecular education. In *Virtual Reality Conference, 2007. VR'07. IEEE*, pages 171–178. IEEE, 2007.

[PGH+04] Eric F. Pettersen, Thomas D. Goddard, Conrad C. Huang, Gregory S. Couch, Daniel M. Greenblatt, Elaine C. Meng, and Thomas E. Ferrin. Ucsf chimera - a visualization system for exploratory research and analysis. *J. Comput. Chem.*, 25(13):1605–1612, 2004.

[PLFL85] N Pattabiraman, M Levitt, TE Ferrin, and R Langridge. Computer graphics in real-time docking with energy calculation and minimization. *J. Comput. Chem.*, 6(5):432–436, 1985.

[PSVV07] Ioannis Ch Paschalidis, Yang Shen, Pirooz Vakili, and Sandor Vajda. Sdu: A semidefinite programming-based underestimation method for stochastic global optimization in protein docking. *Automatic Control, IEEE Transactions on*, 52(4):664–676, 2007.

[RAM$^+$12] Antonio Ricci, Athanasios Anthopoulos, Alberto Massarotti, Ian Grimstead, and Andrea Brancale. Haptic-driven applications to molecular modeling: state-of-the-art and perspectives. *Future Medicinal Chemistry*, 4(10):1219–1228, 2012.

[Rit08] David W Ritchie. Recent progress and future directions in protein-protein docking. *Current protein & peptide science*, 9(1):1, 2008.

[RJ99] Robert C Rizzo and William L Jorgensen. Opls all-atom model for amines: resolution of the amine hydration problem. *J. Am. Chem. Soc*, 121(20):4827–4836, 1999.

[SB06] Erk Subasi and Cagatay Basdogan. A new approach to molecular docking in virtual environments with haptic feedback. In *Proceedings of EuroHaptics Conference*, pages 141–145, 2006.

[SB08] Erk Subasi and Cagatay Basdogan. A new haptic interaction and visualization approach for rigid molecular docking in virtual environments. *Presence: Teleoperators and Virtual Environments*, 17(1):73–90, 2008.

[SCB04] Kenneth Salisbury, Francois Conti, and Federico Barbagli. Haptic rendering: Introductory concepts. *IEEE Computer Graphics and Applications*, 24(2):24–32, 2004.

[SDINW05] Dina Schneidman-Duhovny, Yuval Inbar, Ruth Nussinov, and Haim J. Wolfson. PatchDock and SymmDock: servers for rigid and symmetric docking. *Nucleic Acids Res.*, 33(Web Server issue):W363–W367, July 2005.

[SEC$^+$11] Nathan Schmid, Andreas P Eichenberger, Alexandra Choutko, Sereina Riniker, Moritz Winger, Alan E Mark, and Wilfred F van Gunsteren. Definition and testing of the gromos force-field versions 54a7 and 54b7. *Eur. Biophys. J.*, 40(7):843–856, 2011.

[SG78] Graham M Smith and Peter Gund. Computer-generated space-filling molecular models. *Journal of Chemical Information and Computer Sciences*, 18(4):207–210, 1978.

[SGJ98] Michael JE Sternberg, Henry A Gabb, and Richard M Jackson. Predictive docking of proteinprotein and proteindna complexes. *Current opinion in structural biology*, 8(2):250–256, 1998.

[SGS01] John E Stone, Justin Gullingsrud, and Klaus Schulten. A system for interactive molecular dynamics simulation. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 191–194. ACM, 2001.

[SH09] Bharat Sukhwani and Martin C Herbordt. Gpu acceleration of a production molecular docking code. In *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, pages 19–27. ACM, 2009.

[SH10] Olga Sourina and Xiyuan Hou. Visual haptic-enabled biomolecular docking system design. *Scientific Visualization*, 2(1):49–58, 2010.

[SHL09] Matthew Stocks, Steven Hayward, and Stephen Laycock. Interacting with the biomolecular solvent accessible surface via a haptic feedback device. *BMC Struct. Biol.*, 9(1):69–75, 2009.

[SHO04] Christian M Sauer, Whitney A Hastings, and Allison M Okamura. Virtual environment for exploring atomic bonding. In *Proceedings of Eurohaptics*, pages 5–7. Citeseer, 2004.

[SHUS10] John E Stone, David J Hardy, Ivan S Ufimtsev, and Klaus Schulten. Gpu-accelerated molecular modeling coming of age. *Journal of Molecular Graphics and Modelling*, 29(2):116–125, 2010.

[SKB92] Brian K Shoichet, Irwin D Kuntz, and Dale L Bodian. Molecular docking using shape descriptors. *Journal of Computational Chemistry*, 13(3):380–397, 1992.

[SLH11] Matthew B Stocks, Stephen D Laycock, and S Hayward. Applying forces to elastic network models of large biomolecules using a haptic feedback device. *J. Comput.-Aided Mol. Des.*, 25(3):203–211, 2011.

[SMW95] R. Sayle and J. Milner-White. RasMol: Biomolecular graphics for all. *Trends Biochem. Sci.*, 20:374–376, 1995.

[SPF$^+$07] John E Stone, James C Phillips, Peter L Freddolino, David J Hardy, Leonardo G Trabuco, and Klaus Schulten. Accelerating molecular modeling applications with graphics processors. *J. Comput. Chem.*, 28(16):2618–2640, 2007.

[SS02] Graham R Smith and Michael JE Sternberg. Prediction of protein–protein interactions by docking methods. *Current opinion in structural biology*, 12(1):28–35, 2002.

[Sto10] Matthew Benedict Stocks. *Interacting with biomolecules using haptic feedback*. PhD thesis, University of East Anglia, Norwich, Norfolk, UK, 2010.

[STW09] Olga Sourina, Jaume Torres, and Jing Wang. Visual haptic-based biomolecular docking and its applications in e-learning. In *Transactions on Edutainment II*, pages 105–118. Springer, 2009.

[Sub06] Erk Subasi. Rigidmolecular docking in virtual environments with haptic feedback. Master's thesis, Koc University, Istanbul, Turkey, 6 2006.

[Sut65] Ivan E Sutherland. The ultimate display. *Multimedia: From Wagner to virtual reality*, 1965.

[SVA04] Alexander W Schuttelkopf and Daan MF Van Aalten. Prodrg: a tool for high-throughput crystallography of protein-ligand complexes. *Acta Crystallogr., Sect. D: Biol. Crystallogr.*, 60(8):1355–1363, 2004.

[SWS$^+$03] Ganesh Sankaranarayanan, Suzanne Weghorst, Michel Sanner, Alexandre Gillet, and Arthur Olson. Role of haptics in teaching structural molecular biology. In *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings. 11th Symposium on*, pages 363–366. IEEE, 2003.

[TPJK01] Miguel L Teodoro, George N Phillips Jr, and Lydia E Kavraki. Molecular docking: A problem with thousands of degrees of freedom. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 960–965. IEEE, 2001.

[Tri04] Miguel Angel Otaduy Tristán. *6-dof haptic rendering using contact levels of detail and haptic textures*. PhD thesis, University of North Carolina at Chapel Hill, 2004.

[vdSLHtGdt13] 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.

[VEM09] Bruno O Villoutreix, Richard Eudes, and Maria A Miteva. Structure-based virtual ligand screening: recent success stories. *Combinatorial chemistry & high throughput screening*, 12(10):1000–1016, 2009.

[vO11] Jeremiah van Oosten. Cuda memory model, 2011.

[WCAK$^+$04] Willy Wriggers, Pablo Chacón, Julio A Kovacs, Florence Tama, and Stefan Birmanns. Topology representing neural networks reconcile biomolecular shape, structure, and dynamics. *Neurocomputing*, 56:365–379, 2004.

[WCM97] David R Westhead, David E Clark, and Christopher W Murray. A comparison of heuristic search algorithms for molecular docking. *Journal of Computer-Aided Molecular Design*, 11(3):209–228, 1997.

[WKC$^+$84] Scott J Weiner, Peter A Kollman, David A Case, U Chandra Singh, Caterina Ghio, Guliano Alagona, Salvatore Profeta, and Paul Weiner. A new force field for molecular mechanical simulation of nucleic acids and proteins. *J. Am. Chem. Soc*, 106(3):765–784, 1984.

[WMJ07] Andrew M Wollacott and Kenneth M Merz Jr. Haptic applications for molecular structure manipulation. *J. Mol. Graphics Modell.*, 25(6):801–805, 2007.

[WvD09] Thomas R Weikl and Carola von Deuster. Selected-fit versus induced-fit protein binding: Kinetic differences and mutational analysis. *Proteins: Structure, Function, and Bioinformatics*, 75(1):104–110, 2009.

[YAR11] Elizabeth Yuriev, Mark Agostino, and Paul A. Ramsland. Challenges and advances in computational docking: 2009 in review. *J. Mol. Recognit.*, 24(2):149–164, March 2011.

[ZGAB09] Nicola Zonta, Ian J Grimstead, Nick J Avis, and Andrea Brancale. Accessible haptic technology for drug design applications. *J. Mol. Model.*, 15(2):193–196, 2009.