6-DoF Haptic Rendering Using Contact Levels of Detail and Haptic Textures

by
Miguel Angel Otaduy Tristán

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2004

Approved by:

Ming C. Lin, Advisor

Dinesh Manocha, Reader

Jack Snoeyink, Reader

Russell M. Taylor II, Committee Member

Frederick P. Brooks Jr., Committee Member

ABSTRACT
MIGUEL ANGEL OTADUY TRISTÁN: 6-DoF Haptic Rendering Using Contact Levels of Detail and Haptic Textures.
(Under the direction of Ming C. Lin.)

Humans use tactile and force cues to explore the environment around them and to identify and manipulate objects. An ideal human-machine interface for virtual environments should empower the user to feel and orient objects by providing force feedback. The synthesis of force and torque feedback arising from object-object interaction, commonly referred to as six-degree-of-freedom (6-DoF) haptic rendering, can greatly benefit many applications involving dexterous manipulation and complex maneuvering of virtual objects. Examples of such applications include assembly and disassembly operations in rapid prototyping, endoscopic surgical training, and virtual exploration with limited visual feedback. However, existing techniques for 6-DoF haptic rendering are applicable only to relatively simple contact configurations or low-complexity models.

In this dissertation, I propose a novel approach for 6-DoF haptic rendering that combines multiresolution representations, hierarchical collision detection algorithms, and perception-based force models. This approach enables 6-DoF haptic rendering of interaction between two polygonal models of high combinatorial complexity. I introduce *contact levels of detail*, a collision detection algorithm based on multiresolution hierarchies for performing contact queries between complex models at force update rates, by adaptively selecting the appropriate contact resolutions. I also present a new algorithm for 6-DoF haptic rendering of intricate interaction between textured surfaces, based on a perceptually inspired force model and the representation of the objects as low-resolution models with *haptic textures*. Finally, I derive a novel implicit formulation for computing rigid body dynamics with haptic interaction, and I integrate all these techniques together, thereby achieving stable and responsive 6-DoF haptic rendering.

ACKNOWLEDGMENTS

Six years ago I could not imagine that today I would be here, completing a PhD. I would like to thank my advisor, Prof. Ming Lin, for trusting me in the first place, for being understanding, for helping me identify research problems, and for her good advice, not only on my research, but on all professional aspects in general. I would also like to thank especially the advise of Prof. Dinesh Manocha. Profs. Lin and Manocha have made me enjoy these years of hard work with the GAMMA group at the University of North Carolina, by building an inspiring and encouraging research environment.

Thanks to the rest of the members of my committee, Prof. Jack Snoeyink, Prof. Russell M. Taylor II, and Prof. Frederick P. Brooks Jr. They have all contributed during this process by providing feedback on early drafts of my publications and by sharing their experience in the form of invaluable guidance. And thanks to Dr. Roberta Klatzky and Dr. Susan Lederman, who provided important insight into some of the problems.

I want to thank other colleagues who have helped me get this far. Stephen Ehmann and Young Kim, for their teaching in the area of collision detection, and for building the SWIFT++ and DEEP libraries that I have profusely used during these years. Mark Foskey and Bill Baxter, for all the knowledge they have shared with me, and for helping me with early drafts of my publications. Nitin Jain and Avneesh Sud, for collaborating with me in my work. And the rest of the GAMMA group, for all the fruitful research discussions. But I would like to thank Avneesh Sud especially. I have enjoyed tremendously all the research discussions we have shared, at the office, at home, or even at diners over breakfast after long nights of work.

I deeply appreciate the professional opportunity given by Rick Cunningham at Immersion Medical, as it allowed me to see the field of haptics from a different perspective. And I thank Jokin Mujika, Ana Martínez, and Raúl Reyero at Ikerlan, for all what I have learned from them and for encouraging

and guiding me. I also thank all the faculty and staff of the Department of Computer Science at the University of North Carolina at Chapel Hill, for making Sitterson Hall such a wonderful place to work at. And I thank the alumni of the Department of Computer Science at UNC and the Government of the Basque Country for fellowships that have allowed me to focus on my research, and all other agencies that have supported my research: Intel Corporation, the National Science Foundation, the Office of Naval Research, and the U. S. Army Research Office.

Getting this far involved considerable work, and there are many people who have made it easier by letting me be their friend and by supporting me at all times. In the US, I want to thank first Shelita Atkinson, for making it so easy to move into a new country. And my buddies Adrian Ilie, Avneesh Sud, Udita Patel, John Chek, Heather Hanna, Tom Buller, Crista Wetherington, and Larry Land, for all the fun at home, at the basketball court, and with the kids.

Back in Spain, I thank the friendship of Ana Vidarte, Iván Echevarria, Gorka Rodríguez, Iulen Errarte, Ismael Madina, Luis Miguel Alegría, Jesús Vicente, Itziar Casal, Javier Sánchez, and especially Irantzu López de Bergara, who has been my closest confidant. But also all my cousins and other friends, who always filled me with energy when I went home, and then I was able to keep working hard back in Chapel Hill.

My deepest thanks to God, for being next to me everyday, and to my parents, Angel and Puri, and my sister Arantza, for their love and care. Eskerrik asko.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

When we manipulate an object, we often use it to touch other objects around us. Contacts between objects produce forces and torques that we perceive in our muscles and tendons. These forces and torques convey information about the contours and the compliances of the objects. The perception of geometric information through forces and torques is particularly important in situations with limited visual feedback, as it becomes the main aid for successful task completion.

Imagine a simple assembly task, such as inserting a peg in a hole. A girl picks up the peg and moves it towards the hole, trying to insert it there. The peg collides with the edges of the hole, but it does not fit at the first attempt. She feels the contact in her hand as the peg is suddenly stopped by the surface around the hole. She then moves the peg, feeling how it follows the edges of the hole. And, suddenly, as the peg and the hole are aligned, she can slide the peg in.

During this process, the girl actively translates and rotates the peg. The contact between the peg and the surface around the hole produces forces and torques that are transmitted to her muscles and tendons, and this feedback helps her to accomplish the insertion task. The operation she is performing is an example of 6-degree-of-freedom (DoF) object manipulation with force and torque feedback.

Now imagine a virtual replica of this type of interaction. Instead of picking up a peg, the girl grasps the handle of a haptic device. She can translate and rotate the handle, in a way similar to how she manipulates the actual peg. Simultaneously, a virtual peg is moving in a graphic display, following her actions with the handle. When the virtual peg collides with the surface around the virtual hole, she feels the net contact force and torque in her muscles and tendons, as they are transmitted by the handle of the device. As a result, she is able to insert the virtual peg in the hole intuitively, much

like she does with the real peg. The computation and display of contact force and torque in situations such as this virtual assembly task are known as 6-DoF haptic rendering, and they are the scope of this dissertation.

1.1 Why 6-DoF Haptic Rendering?

For a long time, human beings have dreamed of a virtual world where it is possible to interact with synthetic entities as if they were real. To date, the advances in computer graphics allow us to *see* virtual objects and avatars, to *hear* them, and even to *move* them, but we can rarely *touch* them. It has been shown that the ability to touch virtual objects increases the sense of presence in virtual environments [Ins01]. However, the existing haptic interfaces and computational techniques present serious limitations that, for the most part, restrict haptic interaction with virtual worlds to rather simple desktop applications. In my dissertation I present computational techniques that attempt to enhance haptic interaction with complex virtual objects.

The idea of working on the problem of 6-DoF haptic rendering is mostly motivated by its applicability in engineering and medical training tasks. In this section I briefly describe the evolution of the research in haptic rendering, and I discuss practical applications of 6-DoF haptic rendering. But, first, I define the concept of 6-DoF haptic rendering and some terminology related to the sense of touch.

1.1.1 Definitions

The term *haptic* (from the Greek *haptesthai*, meaning "to touch") is the adjective used to describe something relating to or based on the sense of touch. Haptic is to touching as visual is to seeing and as auditory is to hearing [FFM+04].

As described by Klatzky and Lederman [KL03], touch is one of the main avenues of sensation, and it can be divided into cutaneous, kinesthetic, and haptic systems, based on the underlying neural inputs. The cutaneous system employs receptors embedded in the skin, while the kinesthetic system employs receptors located in muscles, tendons, and joints. The haptic sensory system employs both cutaneous and kinesthetic receptors, but it differs in the sense that it is associated with an active procedure. Touch becomes active when the sensory inputs are combined with controlled body motion.

For example, cutaneous touch becomes active when we explore a surface or grasp an object, while kinesthetic touch becomes active when we manipulate an object and touch other objects with it.

Haptic rendering is defined as the process of computing and generating forces in response to user interactions with virtual objects [SBM+95]. Several haptic rendering algorithms consider the paradigm of touching virtual objects with a single contact point. Rendering algorithms that follow this description are called 3-DoF haptic rendering algorithms, because a point in 3D has only three DoFs. However, my dissertation deals with the problem of rendering the forces and torques arising from the interaction of two virtual objects. This problem is called 6-DoF haptic rendering, because the grasped object has six DoFs (position and orientation in 3D), and the haptic feedback comprises 3D force and torque. When we eat with a fork, write with a pen, or open a lock with a key, we are moving an object in 3D, and we feel the interaction with other objects. This is, in essence, 6-DoF object manipulation with force-and-torque feedback. Six-DoF haptic rendering consists of creating simulated reproductions of these types of interactions. Fig. 1.1 shows an example of a user experiencing 6-DoF haptic rendering. When we manipulate an object and touch other objects with it, we perceive cutaneous feedback as the result of grasping, and kinesthetic feedback as the result of contact between objects. 6-DoF haptic rendering focuses on the synthesis of kinesthetic feedback.

Figure 1.1: **Example of 6-DoF Haptic Rendering.** *A person manipulates a virtual jaw using a haptic device (shown on the right of the image), and the interaction between jaws is displayed both visually and haptically.*

1.1.2 From Telerobotics to 6-DoF Haptic Rendering

In 1965, Ivan Sutherland [Sut65] proposed a multimodal display that would incorporate haptic feedback into the interaction with virtual worlds. Before that, haptic feedback had already been used mainly in two applications: flight simulators and master-slave robotic teleoperation. The early teleoperator systems had mechanical linkages between the master and the slave. But, in 1954, Goertz and Thompson [GT54] developed an electrical servomechanism that received feedback signals from sensors mounted on the slave and applied forces to the master, thus producing haptic feedback.

From there, haptic interfaces evolved in multiple directions, but there were two major breakthroughs central to the type of haptic simulation that I cover in my dissertation. The first breakthrough was the idea of substituting the slave robot by a simulated system, in which forces were computed using physically based simulations. The GROPE project at the University of North Carolina at Chapel Hill [BOYBK90], lasting from 1967 to 1990, was the first one to address the synthesis of force feedback from simulated interactions. In particular, the aim of the project was to perform real-time simulation of 3D molecular-docking forces. The second breakthrough was the advent of computer-based Cartesian control for teleoperator systems [BS80], enabling a separation of the kinematic configurations of the master and the slave. Later, Cartesian control was applied to the manipulation of simulated slave robots [KB91].

Those first haptic systems were able to simulate the interaction of simple virtual objects only. Perhaps the first project to target computation of forces in the interaction with objects with rich geometric information was Minsky's *Sandpaper* [MOYS$^+$90]. Minsky et al. developed a planar force feedback system that allowed the exploration of textures. A few years after Minsky's work, Zilles and Salisbury presented an algorithm for 3-DoF haptic rendering of polygonal models [ZS95]. Almost in parallel with Zilles and Salisbury's work, Massie and Salisbury [MS94] designed the PHANToM, a stylus-based haptic interface that was later commercialized and has become one of the most commonly used force-feedback devices. But in the late '90s, research in haptic rendering revived one of the problems that first inspired virtual force feedback: 6-DoF haptic rendering or, in other words, grasping of a virtual object and synthesis of kinesthetic feedback of the interaction between this object and its environment.

Research in the field of haptics in the last 35 years has covered many more areas than what I have summarized here. I suggest [Bur96] for a survey of haptics and [MHS02] for insight into some of the current research topics in the field.

1.1.3 Application of 6-DoF Haptic Rendering

Certain professional activities, such as training for high-risk operations or pre-production prototype testing, can benefit greatly from simulated reproductions. The fidelity of the simulated reproductions depends, among other factors, on the similarity of the behaviors of real and virtual objects. In the real world, solid objects cannot interpenetrate. Contact forces can be interpreted mathematically as constraint forces imposed by penetration constraints. However, unless penetration constraints are explicitly imposed, virtual objects are free to penetrate each other in virtual environments. Indeed, one of the most disconcerting experiences in virtual environments is to pass through virtual objects [IMWB01, SU93]. Virtual environments require the simulation of non-penetrating rigid body dynamics, and this problem has been extensively explored in the robotics and computer graphics literature [Bar92, Mir96].

But, impenetrability of real-world solid objects is conveyed both visually and haptically. One may conclude that kinesthetic feedback of virtual object contact is required for providing the necessary contact cues. Researchers in virtual reality, however, have often successfully used additional visual and auditory cues (e.g., colors and sounds) for conveying the existence of contact between virtual objects. Some studies even show that there are situations where kinesthetic feedback is not necessary, because subjects tend to follow the sensory cues of the modality with the higher statistical reliability (i.e., visual over haptic) [EB01]. Even if kinesthetic feedback is not essential for some tasks in virtual environments, one can analyze whether kinesthetic feedback provides other benefits, such as higher task performance or higher sense of presence.

It has been shown that being able to touch physical replicas of virtual objects (a technique known as *passive haptics* [Ins01]) increases the sense of presence in virtual environments. This conclusion can probably be generalized to the case of synthetic cutaneous feedback of the interaction with virtual objects. On the other hand, no studies are known on the impact on the sense of presence resulting from kinesthetic feedback during contact with virtual objects through an intermediate grasped object. Nev-

ertheless, as reported by Brooks et al. [BOYBK90], kinesthetic feedback radically improved situation awareness in virtual 3D molecular docking.

Kinesthetic feedback has proved to enhance task performance in applications such as telerobotic object assembly [HS77], virtual object assembly [UNB+02], and virtual molecular docking [OY90]. In particular, task completion time is shorter with kinesthetic feedback in docking operations but not in prepositioning operations. In order to understand this finding, we must look at how subjects take advantage of kinesthetic feedback in everyday life.

Recall the example of the peg-in-the-hole assembly task described at the beginning of this chapter. When the girl tries to insert the peg, she first brings it near the target. In this prepositioning operation, subjects employ visual feedback to direct the motion, and indeed it is known that proprioceptive feedback alone is not accurate enough [TSEC94, Bur96]. The peg is rarely inserted at the first attempt, because the peg itself occludes the hole. At this point, the girl slides the peg along the edges of the hole, following the contours. Kinesthetic feedback of the motion constraints enables a fast search for the position where the peg aligns perfectly with the hole and can be inserted. This operation can be successfully completed without kinesthetic feedback, as proved in visual-only virtual experiments, but the natural and intuitive way to perform the task is to take advantage of kinesthetic feedback, and this tendency could explain the increase in task performance perceived in virtual-docking experiments [BOYBK90]. Even if visual-only feedback were enough for virtual task completion, it seems that the synthesis of kinesthetic feedback is justified in order to provide more natural and intuitive interaction with virtual objects, and this advantage might have an impact on the sense of presence and situation awareness.

To summarize, 6-DoF haptic rendering is especially useful in particular examples of training for high-risk operations or pre-production prototype testing activities that involve intensive object manipulation and interaction with the environment. Such examples include minimally invasive or endoscopic surgery [EHS+97, HGA+98] and virtual prototyping for assembly and maintainability assessment [MPT99, Che99, And02, WM03]. Force feedback becomes particularly important and useful in situations with limited visual feedback. Visual feedback may not exist at all, because of occlusion, or it may be non-intuitive, as in cases where the only visual access to the region of interest is by a camera whose viewing direction is not aligned with the user's viewing direction, which is a very common

situation in endoscopy.

1.1.4 6-DoF vs. 3-DoF

Much of the existing work in haptic rendering has focused on 3-DoF haptic rendering [ZS95, RKK97, TJC97, GLGT99, HBS99]. Given a virtual object A and the 3D position of a point $\mathbf{p}$ governed by an input device, 3-DoF haptic rendering can be summarized as finding a contact point $\mathbf{p}'$ constrained to the surface of A. The contact force will be computed as a function of $\mathbf{p}$ and $\mathbf{p}'$. In a dynamic setting, and assuming that A is a polyhedron with n triangles, the problem of finding $\mathbf{p}'$ has an $O(n)$ worst-case complexity. Using spatial partitioning strategies and exploiting motion coherence, however, the complexity becomes $O(1)$ in many practical situations [GLGT99].

This reduced complexity has made 3-DoF haptic rendering an attractive solution for many applications with virtual haptic feedback, such as: sculpting and deformation [DQ$^+$99, GEL00, MQW01], painting [JTK$^+$99, GEL00, FOL02], volume visualization [AS96], nanomanipulation [TRC$^+$93], and training for diverse surgical operations [KKH$^+$97, GSM$^+$97]. In each of these applications, the interaction between the subject and the virtual objects is sufficiently captured by a point-surface contact model.

In 6-DoF manipulation and exploration, however, when a subject grasps an object and touches other objects in the environment, the interaction generally cannot be modeled by a point-surface contact. One reason is the existence of multiple contacts that impose multiple simultaneous non-penetration constraints on the grasped object. In a simple 6-DoF manipulation example, such as the peg-in-the-hole example described earlier, the grasped object (i.e., the peg) collides at multiple points with the rest of the scene (i.e., the walls of the hole and the surrounding surface). This contact configuration cannot be modeled as a point-object contact. Another reason is that the grasped object presents six DoFs, 3D translation and rotation, as opposed to the three DoFs of a point. The feasible trajectories of the peg are embedded in a 6-dimensional space with translational and rotational constraints, that cannot be captured with three DoFs.

Note that some cases of object-object interaction have been modeled in practice by ray-surface contact [BHS97]. In particular, several surgical procedures are performed with 4-DoF tools (e.g., laparoscopy), and this constraint has been exploited in training simulators with haptic feedback [ÇTS02].

Nevertheless, these approximations are valid only in a limited number of situations and cannot capture full 6-DoF object manipulation.

So far I have discussed the need for a haptic rendering technique that captures full object-object interaction. But 6-DoF haptic rendering also requires 6-DoF tracking of the user's actions and a 6-DoF actuation system. Fortunately, haptic devices serve as both tracking and actuation systems, and various desktop 6-DoF force-and-torque feedback devices exist [BH95, SP97, LPD+98, Che99, GCH+01, Hay01, GFL04]. Some of them are commercially available.

1.2 The Challenges

Six-DoF haptic rendering is in essence an interactive activity, and its realization is mostly handicapped by two conflicting challenges: high required update rates and high computational cost. In this section I will outline the computational pipeline of 6-DoF haptic rendering, and I will discuss the associated challenges.

1.2.1 6-DoF Haptic Rendering Pipeline

Six-DoF haptic rendering comprises two main tasks. One of them is the computation of the position and orientation of the virtual object grasped by the user. The other one is the computation of contact force and torque that are fed back to the user. The existing methods for 6-DoF haptic rendering can be classified into two large groups based on their overall pipelines.

In *direct rendering* methods [NJC99, GME+00, KOLM03, JW03, JW04], the position and orientation of the haptic device are applied directly to the grasped object. Collision detection is performed between the grasped object and the rest of the virtual objects, and collision response is applied to the grasped object as a function of object separation or penetration depth. The resulting contact force and torque are directly fed back to the user.

In *virtual coupling* methods [CC97, Ber99, MPT99, RK00, WM03], the position and orientation of the haptic device are set as goals for the grasped object, and a virtual viscoelastic coupling [CSB95] produces a force that attracts the grasped object to its goals. Collision detection and response are performed between the grasped object and the rest of the virtual objects. The coupling force and

torque are combined with the collision response in order to compute the position and orientation of the grasped object. The same coupling force and torque are fed back to the user.

In Sec. 2.6, I describe the different existing methods for 6-DoF haptic rendering in more detail, and I discuss their advantages and disadvantages. Also, as explained in more detail in Sec. 2.2, there are two major types of haptic devices, and for each type of device the rendering pipeline presents slight variations. Impedance-type devices read the position and orientation of the handle of the device and control the force and torque applied to the user. Admittance-type devices read the force and torque applied by the user and control the position and orientation of the handle of the device.

1.2.2 Force Update Rate

The ultimate goal of 6-DoF haptic rendering is to provide force and torque feedback to the user. This goal is achieved by controlling the handle of the haptic device, which is in fact the end-effector of a robotic manipulator. When the user holds the handle, he or she experiences kinesthetic feedback. The entire haptic rendering system is regarded as a mechanical impedance that sets a transformation between the position and velocity of the handle of the device and the applied force.

The quality of haptic rendering can be measured in terms of the dynamic range of impedances that can be simulated in a stable manner [CB94]. When the user moves the haptic device in free space, the perceived impedance should be very low (i.e., small force), and when the grasped virtual object touches other rigid objects, the perceived impedance should be high (i.e., high stiffness and/or damping of the constraint). The quality of haptic rendering can also be measured in terms of the responsiveness of the simulation [BOYBK90, Ber99]. In free-space motion the grasped virtual object should respond quickly to the motion of the user. Similarly, when the grasped object collides with a virtual wall, the user should stop quickly, in response to the motion constraint.

With impedance-type devices, virtual walls are implemented as large stiffness values in the simulation. In haptic rendering, the user is part of a closed-loop sampled dynamic system [CS94], along with the device and the virtual environment, and the existence of sampling and latency phenomena can induce unstable behavior under large stiffness values. System instability is directly perceived by the user in the form of disturbing oscillations. A key factor for achieving a high dynamic range of impedances (i.e., stiff virtual walls) while ensuring stable rendering is the computation of feedback

forces at a high update rate [CS94, CB94]. Brooks et al. [BOYBK90] reported that, in the rendering of textured surfaces, users were able to perceive performance differences at force update rates between 500Hz and 1kHz. This observation suggests desired force update rates for 6-DoF haptic rendering in the order of 1kHz.

A more detailed description of the stability issues involved in the synthesis of force feedback, and a description of related work, are given in Sec. 2.2. Although here I have focused on impedance-type haptic devices, similar conclusions can be drawn for admittance-type devices (See [AH98a] and Sec. 2.2).

1.2.3 Contact Determination

The computation of non-penetrating rigid-body dynamics of the grasped object and, ultimately, synthesis of haptic feedback require a model of collision response. Forces between the virtual objects must be computed from contact information. Determining whether two virtual objects collide (i.e., intersect) is not enough, and additional information, such as penetration distance, contact points, contact normals, and so forth, need to be computed. Contact determination describes the operation of obtaining the contact information necessary for collision response [Bar92].

Collision response can be computed as a function of object separation, with worst-case cost $O(mn)$, or penetration depth, with a complexity bound of $\Omega(m^3n^3)$. But collision response can also be applied at multiple *contacts* simultaneously. Given two objects A and B with m and n triangles respectively, contacts can be defined as pairs of intersecting triangles or pairs of triangles inside a distance tolerance. The number of pairs of intersecting triangles is $O(mn)$ in worst-case pathological cases, and the number of pairs of triangles inside a tolerance can be $O(mn)$ in practical cases. In Chapter 2, I will discuss in more detail existing techniques for determining the contact information.

The cost of contact determination depends largely on factors such as the convexity of the interacting objects or the contact configuration. There is no direct connection between the polygonal complexity of the objects and the cost of contact determination but, as a reference, existing exact collision detection methods can barely execute contact queries for 6-DoF haptic rendering between pairs of objects with $1,000$ triangles in complex contact scenarios [KOLM03] at force update rates of 1kHz.

Contact determination becomes particularly expensive in the interaction between textured surfaces. Studies have been done on the highest texture resolution that can be perceived through cutaneous touch, but there are no clear results regarding the highest resolution that can be perceived kinesthetically through an intermediate object. It is known that, in the latter case, texture-induced roughness perception is encoded in vibratory motion [KL02]. Psychophysics researchers report that 1mm textures are clearly perceivable, and perceived roughness appears to be even greater with finer textures [LKHG00]. Based on Shannon's sampling theorem, a 10cm $\times$ 10cm plate with a sinusoidal texture of 1mm in orthogonal directions is barely correctly sampled with $40,000$ vertices. This measure gives an idea of the triangulation density required for capturing texture information of complex textured objects. Note that the triangulation density may grow by orders of magnitude if the textures are not sinusoidal and/or if information about normals and curvatures is also needed. The conclusions from this analysis are that highly textured 3D objects of a size comparable to the human hand may require millions of triangles in order to be properly described, and that exact contact determination between two highly textured objects cannot be executed at interactive rates.

1.2.4 Stable and Responsive Interaction with Complex Objects

As discussed throughout this section, stable and responsive 6-DoF haptic rendering requires high force update rates, preferably in the order of 1kHz, and this requirement conflicts with the inherent cost of contact determination between complex models. Previous methods for 6-DoF haptic rendering describe the virtual objects using either parametric surfaces [NJC99], polygonal models [GME$^+$00, KOLM03, JW03, JW04], or combinations of point-sampled and voxelized models [MPT99, WM03]. In Sec. 2.6, I discuss these methods in more detail but, to summarize, they all employ fixed representations of the virtual objects. With fixed representations, the sampling resolution can be selected based on the interactivity of contact determination in complex contact configurations, or based on the smallest perceivable geometric features of the objects. Contact determination between finely sampled complex objects cannot be executed at high force update rates in complex contact configurations. Consequently, fixed representations impose serious limitations for stable and responsive 6-DoF haptic rendering of complex objects.

1.3 Thesis

My thesis is:

> *Efficient multiresolution data structures and collision detection algorithms, coupled with perceptually inspired force models and simplification techniques, can enable stable and responsive 6-degree-of-freedom haptic rendering of complex polygonal models.*

To support this thesis, I present an approach to 6-DoF haptic rendering of complex polygonal models that combines novel algorithms for contact determination, collision response, and synthesis of force and torque feedback.

I have developed an efficient data structure for multiresolution collision detection that combines properties of multiresolution representations and data structures for hierarchical culling. I have also designed a multiresolution collision detection algorithm that uses this novel data structure and selects the appropriate object resolution at every contact. Furthermore, I have designed an efficient algorithm for refining contact information that accounts for surface texture detail.

The creation of the multiresolution representation and the selection of the appropriate contact resolution take advantage of perceptual observations made by psychophysics researchers. I have designed a novel force model that also takes advantage of perceptual factors highlighted in psychophysics studies.

I compute locally linear approximations of the contact forces in order to ensure fast update of the simulation of rigid body dynamics. And I propose an implicit solution to the simulation of rigid body dynamics that, coupled with a fast update of feedback force and torque, enables stable yet responsive interaction. Before describing the main results of my work in more detail, I list general assumptions about the nature of the virtual objects and the haptic devices.

1.3.1 Main Assumptions

Throughout the dissertation, I make several general assumptions:

1. All the objects in the virtual environment, except for the one grasped by the user, are static. Following this assumption, one needs to compute the dynamics of one object only, considerably

reducing the cost of rigid body dynamics simulation and contact determination.

2. The virtual objects are assumed to be rigid. If the objects suffer no deformations, efficient data structures for contact determination can be precomputed.

3. The objects are represented by triangle meshes.

4. The haptic device is of impedance type. This implies that the haptic rendering algorithm receives user positions as inputs and outputs force and torque values.

5. A driver that controls the haptic device is provided. This driver is responsible for measuring the position, orientation, and velocity of the haptic device, as well as for controlling the actuators.

In most of the existing applications of 6-DoF haptic rendering, the environment is, to a large extent, rigid and static. The techniques that I present in this dissertation are valid for applications such as virtual prototyping and surgical training on hard tissue.

As I describe in Sec. 2.2, Adams and Hannaford [AH98a] presented a unified framework for stable haptic rendering with impedance- and admittance-type devices. By using Adams and Hannaford's framework, the techniques presented here can also be applied to admittance-type devices.

1.3.2 Overview of the Rendering Pipeline

As described in Sec. 1.2.2, a high force update rate is the key factor for obtaining stable and responsive 6-DoF haptic rendering. Therefore, my approach to 6-DoF haptic rendering of complex models focuses on the acceleration of contact determination and collision response. In order to achieve stable and responsive haptic rendering, efficient contact determination and collision response techniques must be integrated with an appropriate rendering pipeline. I follow the pipeline of *virtual coupling* methods, described in Sec. 1.2.1, and I also adapt the concept of *intermediate representation*, successfully employed in 3-DoF haptic rendering [AKO95, MRF+96]. The overall haptic rendering pipeline is depicted in Fig. 1.2. Conceptually, I divide the global problem of 6-DoF haptic rendering into three main subproblems:

1. Contact determination.

2. Collision response.

3. Simulation of rigid body dynamics and synthesis of force feedback.

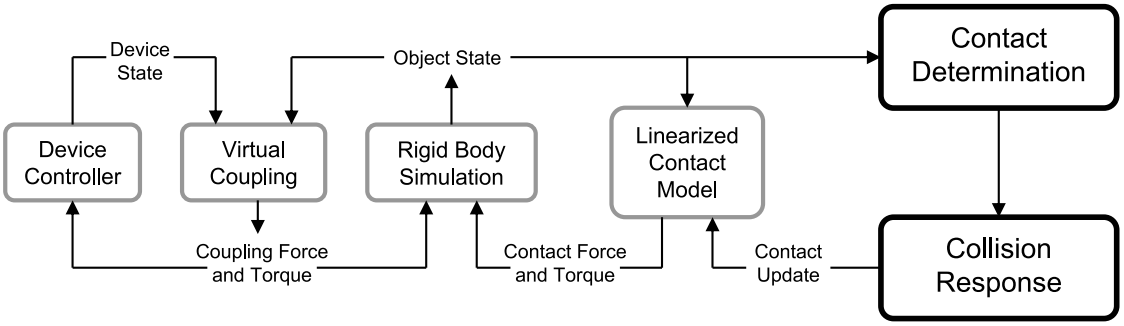

Figure 1.2: **Overall Rendering Pipeline.** *6-DoF haptic rendering pipeline, highlighting the modules of contact determination and collision response.*

One key element of my approach is to decouple the problems of contact determination and collision response from the simulation of rigid body dynamics and synthesis of kinesthetic feedback, adapting the concept of intermediate representation [AKO95, MRF+96]. As a result of contact determination and collision response, I compute a linearized model of the contact forces. This linearized model is used for the simulation of rigid body dynamics and the computation of feedback forces.

Another key element is the computation of contact forces in two steps. The first step identifies contacts between the grasped virtual object and the rest of the environment. Each contact is described at an adaptively selected resolution. The second step refines the contact information at each contact and computes per-contact force and torque.

1.3.3 Main Results

I now discuss the results for 6-DoF haptic rendering that I present in this dissertation. I classify them according to their connections with the subproblems of contact determination, collision response, and simulation of rigid body dynamics with force feedback.

Contact Determination

As explained in Sec. 1.2.3, exact contact determination between geometrically complex surfaces becomes a challenging problem at the high update rates required by haptic rendering. The cost of collision detection grows when large areas of the objects are in contact, but perceptual findings indicate that the perceptibility of surface features decreases with larger contact areas [KL95, OC99, OC01]. This observation motivates the use of multiresolution collision detection, selecting low-resolution representations when the contact area is large, and thus drastically reducing the computational cost. In this dissertation, I present the following key results for contact determination:

- *Contact levels of detail* (CLODs), a novel data structure that combines static levels of detail (LODs) and bounding volume hierarchies (BVHs) into one dual hierarchical representation.
- A multiresolution collision detection algorithm based on CLODs that selects the appropriate object resolution at each contact independently.
- The application of CLODs to contact determination in 6-DoF haptic rendering of complex polygonal models.

I present a general definition of CLODs, independent of the types of bounding volumes and simplification operations employed for creating LODs. I also present a framework for constructing a hierarchy of CLODs, by combining mesh simplification and clustering operations. Along with the general definition, I present an implementation of CLODs using convex hulls. Convex hulls offer attractive properties for fast execution of proximity queries. I have designed a novel atomic simplification operation, *filtered edge collapse*, that combines mesh decimation and filtering, but also accounts for constraints imposed by the selection of convex hulls as bounding volumes. Filtered edge collapses are prioritized based on perceptual observations, thus contributing to a *sensation preserving simplification* process.

The combination of LODs and BVHs in one dual hierarchical representation facilitates the execution of multiresolution collision detection. In this way, descending on the BVH has the additional effect of refining the LODs. I incorporate a selective refinement test to the execution of collision detection using BVHs. The selective refinement test automatically determines the appropriate object

resolution at each contact location independently, and it is founded on perceptually inspired error metrics. I have designed haptic, velocity-dependent, and view-dependent error metrics. All metrics take into account the relationship among local resolution, contact area, and surface deviation. I have defined a metric of mesh resolution that can be compared consistently across different objects, and this consistency facilitates the prioritization of the sensation preserving simplification process, the prioritization of the splitting of bounding volumes in hierarchical collision detection, and the formulation of a consistent error metric that captures the contact error between two objects. I have integrated CLODs with penalty-based collision response methods and with simulation methods that require the location of the time of collision.

I have successfully tested CLODs in 6-DoF haptic rendering of models with more than one hundred thousand triangles. Contact determination using CLODs runs at frequencies higher than 300Hz in complex contact configurations, with little error in the contact information. CLODs enable up to a 2-order-of-magnitude speed-up compared to existing exact collision detection algorithms. CLODs are a general approach to multiresolution collision detection, and their application is not restricted to 6-DoF haptic rendering. For instance, I have used CLODs for collision detection in impulse-based rigid body simulation, achieving roughly 1-order-of-magnitude performance gain compared to exact collision detection methods with objects with tens of thousands of triangles.

Collision Response

Once the contacts between the grasped virtual object and the rest of the environment are identified, it remains to compute per-contact force and torque. CLODs determine the appropriate contact resolution based on observations related to feature identification. However, in the interaction between textured surfaces, forces can also arise due to the interaction between sub-feature geometry. Texture-mapping techniques have been successfully used for the synthesis of feedback forces in 3-DoF haptic rendering [Min95, HBS99], but no previous techniques were able to synthesize force and torque arising from the interaction between two textured surfaces. In this dissertation I present the following key results for collision response:

- The first 6-DoF haptic texture rendering algorithm. It represents objects as low-resolution ge-

ometric representations along with *haptic textures*, texture images that encode fine geometric detail.

- An image-based algorithm for computing directional penetration depth.
- A physically based and perceptually inspired force model for capturing contact force and torque between textured surfaces.

The 6-DoF haptic texture rendering algorithm computes contact information between interacting objects in two steps. First, CLODs are used for performing contact determination, obtaining contact locations and penetration directions between low-resolution representations of the models. Second, a novel image-based algorithm is used for refining the directional penetration depth at each contact, incorporating the geometric detail stored in haptic textures. The use of haptic textures reduces the impact of the high polygonal complexity of the input models on the cost of contact determination while maintaining the effects of surface texture on force display. The image-based algorithm for computing directional penetration depth maps very naturally to an implementation on graphics processors, and I have exploited the parallelism of graphics processors in order to obtain high performance.

I have designed a force model that captures the interaction of textured surfaces in the contact region, in both translational and rotational degrees of freedom. The design of the force model considers factors highlighted in psychophysics studies on perception of roughness [KL02]. Specifically, an analysis of the perceptual observations has led to force and torque equations based on the gradient of penetration depth. I have compared the simulated forces produced by the force model with the results of the studies on roughness perception. Perceptual factors highlighted in psychophysics studies influence simulated texture forces in a way that matches qualitatively their influence on actual roughness perception.

I have successfully applied the 6-DoF haptic texture rendering algorithm to textured models whose full-resolution representations consist of several hundred thousand triangles. The force update rate lies between 100Hz and 200Hz in complex contact situations but reaches 500Hz in less complex cases. I have performed tests on conveyance of roughness effects in translational as well as rotational motion. In both cases the results are satisfactory, showing that the force model and the two-step algorithm for computing contact information successfully capture the interaction between textured surfaces.

Simulation of Rigid Body Dynamics and Synthesis of Force Feedback

As mentioned earlier, I have followed the approach of virtual coupling methods (see Secs. 1.2.1 and 2.6 for more details) for designing the global 6-DoF haptic rendering pipeline. As I discuss in Sec. 2.6, previous virtual coupling methods suffer some limitations. Some methods have only been applied to simple models, others provide limited responsiveness and stability, and others cannot synthesize dynamic force effects. The direct integration of the contact determination and collision response techniques that I have developed with previous virtual coupling methods would suffer from similar problems, especially because the force update rate would drop in complex contact configurations. In this dissertation I present the following key results for the simulation of rigid body dynamics and the synthesis of force feedback:

- Implicit integration of rigid body dynamics simulation with haptic manipulation.
- A linearized contact model that decouples contact determination and collision response from the simulation of rigid body dynamics.
- The application of the complete rendering pipeline in 6-DoF haptic rendering of complex polygonal models, achieving stable and responsive interaction.

I propose the simulation of the rigid body dynamics of the virtual object grasped by the user using implicit integration. Implicit integration offers advantageous stability properties [CSB95, BW98]. It is stable for a set of parameters larger than explicit integration, enabling higher coupling stiffness, higher contact stiffness, smaller object interpenetration, and more responsive motion. Specifically, I simulate the rigid body dynamics of the grasped object following a semi-implicit backward Euler discretization of the Newton-Euler equations of rigid body motion, accounting for user interaction through virtual coupling, penalty-based contact forces, and texture force and torque. The formulation involves the linearization in the state-space of a rigid body (positions and velocities) of viscoelastic virtual coupling force and torque, penalty-based force and torque, and texture force and torque

A linearized contact model that decouples the computation of contact forces from the simulation of rigid body dynamics enables a multirate architecture of the rendering pipeline. The simulation of rigid body dynamics and the synthesis of force feedback run in a fast thread, while the execution

of contact determination and collision response run in a separate asynchronous thread. In this way, contact determination is not a bottleneck for the update of output force and torque, with beneficial consequences on stability and responsiveness. The formulation of the linearized contact model employs the same linearization as the formulation of implicit integration, therefore it involves almost no additional cost. I propose a contact clustering technique, based on the K-means clustering algorithm [JMF99], that clusters the contacts output by the contact determination module and computes a set of representative contacts for collision response.

I have integrated multiresolution collision detection using CLODs in the rendering pipeline, producing stable and responsive 6-DoF haptic rendering of models with tens of thousands of triangles. The interaction between complex models remains highly responsive and stable at force update rates of 1kHz, even with update rates of contact determination as low as 100Hz.

1.3.4 Organization

The rest of the dissertation is organized as follows. Chapter 2 describes and discusses related work. Chapter 3 presents *contact levels of detail* for multiresolution collision detection. Chapter 4 introduces a 6-DoF haptic texture rendering algorithm based on *haptic textures*. Chapter 5 presents a stable and responsive 6-DoF haptic rendering pipeline, based on implicit integration of rigid body dynamics simulation. Finally, Chapter 6 summarizes the main conclusions and discusses future research directions.

Chapter 2

Related Work

The development of techniques for stable and responsive 6-degree-of-freedom haptic rendering of complex polygonal models is founded largely on a good understanding of four main subjects: the psychophysics of haptic perception, the application of control theory to haptic rendering, collision detection algorithms, and rigid body simulation. The first four sections of this chapter cover relevant work on these four subjects. The remainder of the chapter discusses previous methods for haptic texture rendering and 6-DoF haptic rendering, which constitute the overall research problems tackled in this dissertation.

2.1 Psychophysics of Haptics

In the design of contact determination algorithms for haptic rendering, it is crucial to understand the psychophysics of touch and to account for perceptual factors. The structure and behavior of human touch have been studied extensively in the field of psychology. The topics analyzed by researchers include characterization of sensory phenomena as well as cognitive and memory processes.

Haptic perception of physical properties includes a first step of stimulus registration and communication to the thalamus, followed by a second step of higher-level processing. Perceptual measures can be originated by individual mechanoreceptors but also by the integration of inputs from populations of different sensory units [KL03]. Klatzky and Lederman [KL03] discuss object and surface properties that are perceived through the sense of touch (e.g., texture, hardness, and weight) and divide them between geometric and material properties. They also analyze active exploratory procedures (e.g.,

lateral motion, pressure, or unsupported holding) typically conducted by subjects in order to capture information about the different properties.

Knowing the exploratory procedure(s) associated with a particular object or surface property, researchers have studied the influence of various parameters on the accuracy and magnitude of sensory outputs. Perceptual studies on tactile feature detection and identification, as well as studies on texture or roughness perception are of particular interest for my dissertation. In this section I summarize existing research on perception of surface features and perception of roughness, and then I discuss issues associated with the interaction of visual and haptic modalities.

2.1.1 Perception of Surface Features

Klatzky and Lederman describe two different exploratory procedures followed by subjects in order to capture shape attributes and identify features and objects. In *haptic glance* [KL95], subjects extract information from a brief haptic exposure of the object surface. Then they perform higher-level processing for determining the identity of the object or other attributes. In *contour following* [KL03], subjects create a spatiotemporal map of surface attributes, such as curvature, that serves as the pattern for feature identification. Contact determination algorithms attempt to describe the geometric interaction between virtual objects. Fast contact determination algorithms for 6-DoF haptic rendering must minimize the computational cost while preserving important geometric attributes. The instantaneous nature of haptic glance [KL95] makes it strongly dependent on purely geometric attributes, unlike the temporal dependency of contour following.

Klatzky and Lederman [KL95] conducted experiments in which subjects were instructed to identify objects from brief cutaneous exposures (i.e., haptic glances). Subjects had an advance hypothesis of the nature of the object. The purpose of the study was to discover how, and how well, subjects identify objects from brief contact. According to Klatzky and Lederman, during haptic glance a subject has access to three pieces of information: roughness, compliance, and local features. Roughness and compliance are material properties that can be extracted from lower-level processing, while local features can lead to object identification by feature matching during higher-level processing. In the experiments, highest identification accuracy was achieved with small objects, whose *shapes* fit on a fingertip. Klatzky and Lederman concluded that large contact area helped in the identification of

textures or patterns, although it was better to have a stimulus of a size comparable to or just slightly smaller than that of the contact area for the identification of geometric surface features.

The experiments conducted by Klatzky and Lederman posit an interesting relation between feature size and contact area during cutaneous perception. For the purpose of designing a contact determination algorithm, however, I am interested in kinesthetic perception arising from the interaction of two objects. Okamura and Cutkosky [OC99, OC01] analyzed feature detection in robotic exploration, which can be regarded as a case of object-object interaction. They characterized geometric surface features based on the ratios of their curvatures to the radii of the robotic fingertips acquiring the surface data. They observed that a larger fingertip, which provides a larger contact area, can miss small geometric features. To summarize, the studies by Klatzky and Lederman [KL95] and Okamura and Cutkosky [OC99, OC01] lead to the following observation, which drives the design of contact determination algorithms for 6-DoF haptic rendering (See Sec. 3.1.1):

> *Human haptic perception of the existence of a geometric surface feature depends on the ratio between the contact area and the size of the feature, not the absolute size of the feature itself.*

2.1.2 Perception of Texture and Roughness

Klatzky and Lederman [KL03] describe a textured surface as a surface with protuberant elements arising from a relatively homogeneous substrate. Interaction with a textured surface results in perception of roughness. Existing research on the psychophysics of texture perception indicates a clear dichotomy of exploratory procedures: (a) perception of texture with the bare skin, and (b) perception through an intermediate (rigid) object, a probe.

Most of the research efforts have been directed towards the characterization of cutaneous perception of textures. Katz [Kat89] suggested that roughness is perceived through a combination of spatial and vibratory codes during direct interaction with the skin. More recent evidence demonstrates that static pressure distribution plays a dominant role in perception of coarse textures (features larger than 1mm) [Led74, CJ92], but motion-induced vibration is necessary for perceiving fine textures [LS91, HR00].

Six-DoF haptic rendering tackles the display of texture- and/or roughness-induced forces arising from the interaction between two objects. As pointed out by Klatzky and Lederman [KL02], in object-object interaction roughness is encoded in vibratory motion transmitted to the subject. In the last few years, Klatzky and Lederman have directed experiments that analyze the influence of several factors on roughness perception through a rigid probe. Klatzky et al. [KLH$^+$03] distinguished three types of factors that may affect the perceived magnitude of roughness: interobject physical interaction, skin- and limb-induced filtering prior to cutaneous and kinesthetic perception, and higher-level factors such as efferent commands. The design of contact determination and collision response algorithms for haptic texture rendering is mostly concerned with factors related to the physical interaction between objects: object geometry [LKHG00, KLH$^+$03], applied force [LKHG00], and exploratory speed [LKHR99, KLH$^+$03]. In Sec. 4.2.1, I describe how the influence of these factors is addressed in the design of a 6-DoF haptic texture rendering algorithm.

The experiments conducted by Klatzky and Lederman to characterize roughness perception [KL02] used a common setup: subjects explored a textured plate with a probe with a spherical tip, and then they reported a subjective measure of roughness. Plates of jittered raised dots were used, and the mean frequency of dot distribution was one of the variables in the experiments. The resulting data was analyzed by plotting subjective roughness values vs. dot interspacing in logarithmic graphs, as shown in Figs. 4.8, 4.9, and 4.10.

Klatzky and Lederman [KL99] compared graphs of roughness vs. texture spacing (a) with finger exploration and (b) with a rigid probe. They concluded that, in the range of their data, roughness functions were best fit by linear approximations in finger exploration and by quadratic approximations in probe-based exploration. In other words, when perceived through a rigid spherical probe, roughness initially increases as texture spacing increases, but, after reaching a maximum roughness value, it decreases again. Based on this finding, the influence of other factors on roughness perception can be characterized by the maximum value of roughness and the value of texture spacing at which this maximum takes place.

Lederman et al. [LKHG00] demonstrated that the diameter of the spherical probe plays a crucial role in the maximum value of perceived roughness and the location of the maximum. The roughness peak is higher for smaller probes, and it occurs at smaller texture spacing values (See Fig. 4.8).

Lederman et al. [LKHG00] also studied the influence of the applied normal force during exploration. Roughness is higher for larger force, but the influence on the location of the peak is negligible (See Fig. 4.9).

The effect of exploratory speed was studied by Lederman et al. [LKHR99]. They found that the peak of roughness occurs at larger texture spacing for higher speed (See Fig. 4.10). Also, with higher speed, textured plates feel smoother at small texture spacing, and rougher at large spacing values. The studies reflected that speed has a stronger effect in passive interaction than in active interaction.

2.1.3 Haptic and Visual Cross-modal Interaction

Haptic rendering is often presented along with visual display. Therefore, it is important to understand the issues involved in cross-modal interaction. Klatzky and Lederman [KL03] discuss aspects of visual and haptic cross-modal integration from two perspectives: attention and dominance.

Spence et al. [SPD00] have studied how visual and tactile cues can influence a subject's attention. Their conclusions are that visual and tactile cues are treated together in a single attentional mechanism, and wrong attention cues can affect perception negatively.

Sensory dominance is usually studied by analyzing perceptual discrepancies in situations where cross-modal integration yields a unitary perceptual response. One example of relevance for this dissertation is the detection of object collision. During object manipulation, humans determine whether two objects are in contact based on a combination of visual and haptic cues. Early studies of sensory dominance seemed to point to a strong dominance of visual cues over haptic cues [RV64], but in the last decades psychologists agree that sensory inputs are weighted based on their statistical reliability or relative appropriateness, measured in terms of accuracy, precision, and cue availability [HCGB99, EB01, KL03].

The design of contact determination algorithms can also benefit from existing studies on the visual perception of collisions in computer animations. O'Sullivan and her colleagues [ORC99, OD01, ODGK03] have investigated different factors affecting visual collision perception, including eccentricity, separation, distractors, causality, and accuracy of simulation results. Basing their work on a model of human visual perception validated by psychophysical experiments, they demonstrated the feasibility of using these factors for scheduling interruptible collision detection among large num-

bers of visually homogeneous objects. In this dissertation I present contact determination techniques that are not restricted to haptic rendering and can also be applied to rigid body simulation. Consequently, these techniques can benefit from the observations related to visual perception of collisions, as presented in Sec. 3.4.3.

2.2 Stability and Control Theory Applied to Haptic Rendering

In haptic rendering, the human user is part of the dynamic system, along with the haptic device and the computer implementation of the virtual environment. The complete human-in-the-loop system can be regarded as a sampled-data system [CS94], with a continuous component (the user and the device) and a discrete one (the implementation of the virtual environment and the device controller). Stability becomes a crucial feature, because instabilities in the system can produce oscillations that distort the perception of the virtual environment, or uncontrolled motion of the device that can even hurt the user. In Sec. 1.2.2, I have briefly discussed the importance of stability for haptic rendering, and I have introduced the effect of the force update rate on stability. In this section I review and discuss in more detail existing work in control theory related to stability analysis of haptic rendering.

2.2.1 Mechanical Impedance Control

The concept of mechanical impedance extends the notion of electrical impedance and refers to the quotient between force and velocity. Hogan [Hog85] introduced the idea of impedance control for contact tasks in manipulation. Earlier techniques controlled contact force, robot velocity, or both, but Hogan suggested controlling directly the mechanical impedance, which governs the dynamic properties of the system. When the end effector of a robot touches a rigid surface, it suffers a drastic change of mechanical impedance, from low impedance in free space, to high impedance during contact. This phenomenon imposes serious difficulties on earlier control techniques, inducing instabilities.

The function of a haptic device is to display the feedback force of a virtual world to a human user. Haptic devices present control challenges very similar to those of manipulators for contact tasks. As introduced in Sec. 1.2.1, there are two major ways of controlling a haptic device: impedance control and admittance control. In impedance control, the user moves the device, and the controller produces

a force dependent on the interaction in the virtual world. In admittance control, the user applies a force to the device, and the controller moves the device according to the virtual interaction.

In both impedance and admittance control, high control gains can induce instabilities. In impedance control, instabilities may arise in the simulation of stiff virtual surfaces. The device must react with large changes in force to small changes in the position. Conversely, in admittance control, rendering a stiff virtual surface is not a challenging problem, because it is implemented as a low controller gain. In admittance control, however, instabilities may arise during free-space motion in the virtual world, because the device must move at high velocities under small applied forces, or when the device rests on a stiff physical surface. Impedance and admittance control can therefore be regarded as complementary control techniques, best suited for opposite applications. The contact determination and force computation algorithms designed in this dissertation are independent of the control strategy. On the other hand, the force rendering technique presented in Chapter 5 assumes impedance control of the haptic device, but it can be adapted to admittance control, following the unifying framework presented by Adams and Hannaford [AH98a].

2.2.2 Stable Rendering of Virtual Walls

Since the introduction of impedance control by Hogan [Hog85], the analysis of the stability of haptic devices and haptic rendering algorithms has focused on the problem of rendering stiff virtual walls. This was known to be a complex problem at early stages of research in haptic rendering [Kil76], but impedance control simplified the analysis, because a virtual wall can be modeled easily using stiffness and viscosity parameters.

Ouh-Young [OY90] created a discrete model of the Argonne ARM and the human arm and analyzed the influence of force update rate on the stability and responsiveness of the system. Minsky, Brooks, et al. [MOYS$^+$90, BOYBK90] observed that update rates as high as 500Hz or 1kHz might be necessary in order to achieve stability.

Colgate and Brown [CB94] coined the term Z-Width for describing the range of mechanical impedances that a haptic device can render while guaranteeing stability. They concluded that physical dissipation is essential for achieving stability and that the maximum achievable virtual stiffness is proportional to the update rate. They also analyzed the influence of position sensors and quantization,

and concluded that sensor resolution must be maximized and the velocity signal must be filtered.

Almost in parallel, Salcudean and Vlaar [SV94] studied haptic rendering of virtual walls, and techniques for improving the fidelity of the rendering. They compared a continuous model of a virtual wall with a discrete model that accounts for differentiation of the position signal. The continuous model is unconditionally stable, but this is not true for the discrete model. Moreover, in the discrete model fast damping of contact oscillations is possible only with rather low contact stiffness and, as indicated by Colgate and Brown too [CB94], this value of stiffness is proportional to the update rate. Salcudean and Vlaar proposed the addition of braking pulses, proportional to collision velocity, for improving the perception of virtual walls.

2.2.3 Passivity and Virtual Coupling

A subsystem is *passive* if it does not add energy to the global system. Passivity is a powerful tool for analyzing stability of coupled systems, because the coupled system obtained from two passive subsystems is always stable. Colgate and his colleagues were the first to apply passivity criteria to the analysis of stability in haptic rendering of virtual walls [CGSS93]. Passivity-based analysis has enabled separate study of the behavior of the human subsystem, the haptic device, and the virtual environment in force-feedback systems.

Human Sensing and Control Bandwidths

Hogan discovered that the human neuromuscular system exhibits externally simple, springlike behavior [Hog86]. This finding implies that the human arm holding a haptic device can be regarded as a passive subsystem, and the stability analysis can focus on the haptic device and the virtual environment.

Note that human limbs are not passive in all conditions, but the bandwidth at which a subject can perform active motions is very low compared to the frequencies at which stability problems may arise. Some authors [Shi92, Bur96] report that the bandwidth at which humans can perform controlled actions with the hand or fingers is between 5 and 10Hz. On the other hand, sensing bandwidth can be as high as 20 to 30Hz for proprioception, 400Hz for tactile sensing, and 5 to 10kHz for roughness perception.

Passivity of Virtual Walls

Colgate and Schenkel [CS94] observed that the oscillations perceived by a haptic user during system instability are a result of active behavior of the force-feedback system. This active behavior is a consequence of time delay and loss of information inherent in sampled-data systems, as suggested by others before [BOYBK90]. Colgate and Schenkel formulated passivity conditions in haptic rendering of a virtual wall. For that analysis, they modeled the virtual wall as a viscoelastic unilateral constraint, and they accounted for the continuous dynamics of the haptic device, sampling of the position signal, discrete differentiation for obtaining velocity, and a zero-order hold of the output force. They reached a sufficient condition for passivity that relates the stiffness K and damping B of the virtual wall, the inherent damping b of the device, and the sampling period T:

$$b > \frac{KT}{2} + B. \tag{2.1}$$

Stability of Non-linear Virtual Environments

After deriving stability conditions for rendering virtual walls modeled as unilateral linear constraints, Colgate and his colleagues considered more complex environments [CSB95]. A general virtual environment is non-linear, and it presents multiple and variable constraints. Their approach enforces a discrete-time passive implementation of the virtual environment and sets a multidimensional viscoelastic *virtual coupling* between the virtual environment and the haptic display. In this way, the stability of the system is guaranteed as long as the virtual coupling is itself passive, and this condition can be analyzed using the same techniques as those used for virtual walls [CS94]. As a result of Colgate's virtual coupling [CSB95], the complexity of the problem was shifted towards designing a passive solution of virtual world dynamics. As noted by Colgate et al. [CSB95], one possible way to enforce passivity in rigid body dynamics simulation is to use implicit integration with penalty methods.

Adams and Hannaford [AH98a] provided a framework for analyzing stability with admittance-type and impedance-type haptic devices. They derived stability conditions for coupled systems based on network theory. They also extended the concept of virtual coupling to admittance-type devices.

Miller et al. [MCF90] extended Colgate's passivity analysis techniques, relaxing the requirement of passive virtual environments but enforcing *cyclo-passivity* of the complete system. Hannaford and his colleagues [HRK02] investigated the use of adaptive controllers instead of the traditional fixed-value virtual couplings. They designed passivity observers and passivity controllers for dissipating the excess of energy generated by the virtual environment.

2.2.4 Multirate Approximation Techniques

Multirate approximation techniques, though simple, have been successful in improving the stability and responsiveness of haptic rendering systems. The idea is to perform a full update of the virtual environment at a low frequency (limited by computational resources and the complexity of the system) and to use a simplified approximation for performing high-frequency updates of force feedback.

Adachi [AKO95] proposed an *intermediate representation* for haptic display of complex polygonal objects. In a slow collision detection thread, he computed a plane that served as a unilateral constraint in the force-feedback thread. This technique was later adapted by Mark et al. [MRF+96], who interpolated the intermediate representation between updates. This approach enables higher stiffness values than approaches that compute the feedback force values at the rate imposed by collision detection. More recently, a similar multirate approach has been followed by many authors for haptic interaction with deformable models [AH98b, ÇT00, DAK04]. Ellis et al. [ESJ97] produce higher-quality rendering by upsampling directly the output force values.

2.3 Collision Detection

Collision detection has received much attention in robotics, computational geometry, and computer graphics. Some researchers have investigated the problem of interference detection as a mechanism for indicating whether object configurations are valid or not. Others have tackled the problems of computing separation or penetration distances, with the objective of applying collision response in simulated environments. The existing work on collision detection can be classified based on the types of models handled: 2-manifold polyhedral models, polygon soups, curved surfaces, etc. In this section I focus on collision detection between polyhedral models. The vast majority of the algorithms used

in practice proceed in two steps: first they cull large portions of the objects that are not in close proximity, using spatial partitioning, hierarchical techniques, or visibility-based properties, and then they perform primitive-level tests.

In this section, I first describe the problems of interference detection and computation of separation distance between polyhedra, with an emphasis on algorithms specialized for convex polyhedra. Then I survey algorithms for the computation of penetration depth, the use of hierarchical techniques, and multiresolution collision detection. I conclude the section by covering briefly the use of graphics processors for collision detection and the topic of continuous collision detection. For more information on collision detection, please refer to surveys on the topic [LG98, KHM+98, LM04].

2.3.1 Proximity Queries Between Convex Polyhedra

The property of convexity has been exploited in algorithms with sublinear cost for detecting interference or computing the closest distance between two polyhedra. Detecting whether two convex polyhedra intersect can be posed as a linear programming problem, searching for the coefficients of a separating plane. Well-known linear programming algorithms [Sei90] can run in expected linear time due to the low dimensionality of the problem.

The separation distance between two polyhedra A and B is equal to the distance from the origin to the Minkowski sum of A and $-B$ [CC86]. This property was exploited by Gilbert et al. [GJK88] in order to design a convex optimization algorithm (known as GJK) for computing the separation distance between convex polyhedra, with linear-time performance in practice. Cameron [Cam97] modified the GJK algorithm to exploit motion coherence in the initialization of the convex optimization at every frame for dynamic problems, achieving nearly constant running-time in practice.

Lin and Canny [LC91, Lin93] designed an algorithm for computing separation distance by tracking the closest features between convex polyhedra. Their algorithm "walks" on the surfaces of the polyhedra until it finds two features that lie on each other's Voronoi region. Exploiting motion coherence and geometric locality, *Voronoi marching* runs in nearly constant time per frame. Mirtich [Mir98b] later improved the robustness of this algorithm.

Given polyhedra A and B with m and n polygons respectively, Dobkin and Kirkpatrick [DK90] proposed an algorithm for interference detection with $O(\log m \log n)$ time complexity that uses hi-

erarchical representations of the polyhedra. Others have also exploited the use of hierarchical convex representations along with temporal coherence in order to accelerate queries in dynamic scenes. Guibas et al. [GHZ99] employ the inner hierarchies suggested by Dobkin and Kirkpatrick, but they perform faster multilevel walking. Ehmann and Lin [EL00] employ a modified version of Dobkin and Kirkpatrick's outer hierarchies, computed using simplification techniques, along with a multilevel implementation of Lin and Canny's Voronoi marching [LC91].

2.3.2 Penetration Depth

The penetration depth between two intersecting polyhedra A and B is defined as the minimum translational distance required for separating them. For intersecting polyhedra, the origin is contained in the Minkowski sum of A and $-B$, and the penetration depth is equal to the minimum distance from the origin to the surface of the Minkowski sum. The computation of penetration depth can be $\Omega(m^3n^3)$ for general polyhedra [DHKS93].

Many researchers have restricted the computation of penetration depth to convex polyhedra. In computational geometry, Dobkin et al. [DHKS93] presented an algorithm for computing directional penetration depth, while Agarwal et al. [AGHP$^+$00] introduced a randomized algorithm for computing the penetration depth between convex polyhedra. Cameron [Cam97] extended the GJK algorithm [GJK88] to compute bounds of the penetration depth, and van den Bergen [van01] furthered his work. Kim et al. [KLM02] presented an algorithm that computes a locally optimal solution of the penetration depth by walking on the surface of the Minkowski sum.

The fastest algorithms for computation of penetration depth between arbitrary polyhedra take advantage of discretization. Fisher and Lin [FL01] estimate penetration depth using distance fields computed with fast marching level-sets. Hoff et al. [HZLM01] presented an image-based algorithm implemented on graphics hardware. On the other hand, Kim et al. [KOLM02] presented an algorithm that decomposes the polyhedra into convex patches, computes the Minkowski sums of pairwise patches, and then uses an image-based technique in order to find the minimum distance from the origin to the surface of the Minkowski sums.

2.3.3 Hierarchical Collision Detection

The algorithms for collision detection between convex polyhedra are not directly applicable to non-convex polyhedra or models described as polygon soups. Brute force checking of all triangle pairs, however, is usually unnecessary. Collision detection between general models achieves large speed-ups by using hierarchical culling or spatial partitioning techniques that restrict the primitive-level tests. Over the last decade, bounding volume hierarchies (BVH) have proved successful in the acceleration of collision detection for dynamic scenes of rigid bodies. For an extensive description and analysis of the use of BVHs for collision detection, please refer to Gottschalk's PhD dissertation [Got00].

Assuming that an object is described by a set of triangles T, a BVH is a tree of BVs, where each BV C_i bounds a cluster of triangles $T_i \in T$. The clusters bounded by the children of C_i constitute a partition of T_i. The effectiveness of a BVH is conditioned by ensuring that the branching factor of the tree is $O(1)$ and that the size of the leaf clusters is also $O(1)$. Often, the leaf BVs bound only one triangle. A BVH may be created in a top-down manner, by successive partitioning of clusters, or in a bottom-up manner, by using merging operations.

In order to perform interference detection using BVHs, two objects are queried by recursively traversing their BVHs in tandem. Each recursive step tests whether a pair of BVs a and b, one from each hierarchy, overlap. If a and b do not overlap, the recursion branch is terminated. Otherwise, if they overlap, the algorithm is applied recursively to their children. If a and b are both leaf nodes, the triangles within them are tested directly. This process can be generalized to other types of proximity queries as well.

One determining factor in the design of a BVH is the selection of the type of BV. Often there is a trade-off among the tightness of the BV (and therefore the culling efficiency), the cost of the collision test between two BVs, and the dynamic update of the BV (relevant for deformable models). Some of the common BVs, sorted approximately according to increasing query time, are: spheres [Qui94, Hub94], axis-aligned bounding boxes (AABB) [BKSS90], oriented bounding boxes (OBB) [GLM96], k-discrete-orientation polytopes (k-DOP) [KHM$^+$98], convex hulls [EL01], and swept sphere volumes (SSV) [LGLM00]. BVHs of rigid bodies can be computed as a preprocessing step, but deformable models require a bottom-up update of the BVs after each deformation. Recently, James and Pai [JP04]

have presented the BD-tree, a variant of the sphere-tree data structure [Qui94] that can be updated in a fast top-down manner if the deformations are described by a small number of parameters. As will be explained in Sec. 3.3.1, I have opted for using BVHs of convex hulls for collision detection.

2.3.4 Multiresolution Collision Detection

Multiresolution analysis of a function decomposes the function into a basic low-resolution representation and a set of detail terms at increasing resolutions. Wavelets provide a mathematical framework for defining multiresolution analysis [SDS96].

Multiresolution representations of triangles meshes have drawn important attention in computer graphics. They have been defined in two major ways: following the mathematical framework of wavelets and subdivision surfaces [LDW97, EDD+95] or following level-of-detail (LOD) simplification techniques (please refer to [LRC+02] for a survey on the topic). LOD techniques present the advantage of being applicable to arbitrary meshes, but they lack a well-defined metric of resolution. They construct the multiresolution representations starting from full-resolution meshes and applying sequences of local simplification operations. LOD techniques can be divided into those that produce a discrete set of representations (static LODs), and those that produce continuously adaptive representations (dynamic LODs). Multiresolution or LOD techniques have been used in applications such as view-dependent rendering [Hop97, LE97], interactive editing of meshes [ZSS97], or real-time deformations [DDCB01]. The idea behind multiresolution techniques is to select the resolution or LOD of the representation in an adaptive manner based on perceptual parameters, availability of computational resources, and so forth.

Multiresolution collision detection refers to the execution of approximate collision detection queries using adaptive object representations. However, little work exists in this respect. Hubbard [Hub94] introduced the idea of using sphere-trees [Qui94] for multiresolution collision detection, refining the BVHs in a breadth-first manner until the time allocated for collision detection expires. In a sphere-tree each level of the BVH can be regarded as an implicit approximation of the given mesh, by defining the surface as a union of spheres. Unlike LOD techniques, in which simplification operations minimize surface deviation, sphere-trees add extraneous “bumpiness” to the surface, and this characteristic can hurt collision response.

O'Sullivan and Dingliana [OD01] have incorporated perceptual parameters into the refinement of sphere-trees. They insert pairs of spheres that test positive for collision in a priority queue sorted according to perceptual metrics (e.g., local relative velocity, distance to the viewer, etc.). In this way the adaptive refinement focuses on areas of the objects where errors are most noticeable.

The use of multiresolution representations for haptic rendering has also been investigated by several researchers. Pai and Reissel [PR97] investigated the use of multiresolution image curves for 2D haptic interaction. El-Sana and Varshney [ESV00] applied LOD techniques to 3-DoF haptic rendering. They created a multiresolution representation of the haptically rendered object as a preprocessing step and, at runtime, they represented the object at high resolution near the probe point and at low resolution further away. Their approach does not extend naturally to the interaction between two objects, since multiple disjoint contacts can occur simultaneously at widely varying locations without much spatial coherence.

2.3.5 Other Techniques for Collision Detection

I briefly cover two additional topics with potential applicability in haptic rendering: the use of graphics processors for collision detection, and continuous collision detection.

Use of Graphics Processors for Collision Detection

The processing capability of GPUs is growing at a rate higher than Moore's law [GRLM03], and this circumstance has generated an increasing use of GPUs for general-purpose computation, including collision detection. Rasterization hardware enables high performance of image-based collision detection algorithms. Hoff et al. [HZLM01] presented an algorithm for estimating penetration depth between deformable polygons using distance fields computed on graphics hardware. Others have formulated collision detection queries as visibility problems. Lombardo et al. [LCN99] intersected a complex object against a simpler one using the view frustum and clipping planes, and they detected intersecting triangles by exploiting OpenGL capabilities. More recently, Govindaraju et al. [GRLM03] have designed an algorithm that performs series of visibility queries and achieves fast culling of non-intersecting primitives in N-body problems with nonrigid motion.

Continuous Collision Detection

Continuous collision detection refers to a temporal formulation of the collision detection problem. The collision query attempts to find intersecting triangles and the time of intersection. Redon et al. [RKC02] proposed an algorithm that assumes an arbitrary interframe rigid motion and incorporates the temporal dimension in OBB-trees using interval arithmetic. Continuous collision detection offers potential applicability to haptic rendering because it may enable constraint-based simulations without expensive backtracking operations used for computing the time of first collision.

2.4 Rigid Body Simulation

Computation of the motion of a rigid body consists of solving a set of ordinary differential equations (ODEs). The most common way to describe the motion of a rigid body is by means of the Newton-Euler equations, which define the time derivatives of the linear momentum, **P**, and angular momentum, **L**, as a function of external force **F** and torque **T**:

$$\begin{aligned} \mathbf{F}(t) &= \dot{\mathbf{P}}(t) = m\,\ddot{\mathbf{x}}(t), \\ \mathbf{T}(t) &= \dot{\mathbf{L}}(t) = \omega(t) \times (M\omega(t)) + M\dot{\omega}(t). \end{aligned} \tag{2.2}$$

As shown in the equations, momentum derivatives can be expressed in terms of the linear acceleration of the center of mass $\ddot{\mathbf{x}}$, the angular velocity ω, the mass of the body m, and the mass matrix M. The complexity of rigid body simulation lies in the computation of force and torque resulting from contacts between bodies. Research in the field of rigid body simulation has revolved around different methods for computing contact forces and the resulting accelerations and velocities, ranging from approximate methods that consider each contact independently (such as penalty-based methods) to analytic methods that account concurrently for all non-penetration constraints. Important efforts have been devoted to capturing friction forces as well.

In this section I briefly describe the main methods for solving the motion of colliding rigid bodies, focusing on their applicability to haptic rendering. For further information, please refer to Baraff's

or Mirtich's dissertations [Bar92, Mir96], SIGGRAPH course notes on the topic [BW01], or recent work by Stewart and Trinkle [ST00]. In the last few years, especially in the field of computer graphics, attention has been drawn towards the problem of simulating the interaction of many rigid bodies [Mir00, MS01, GBF03]. For haptic rendering, however, one is mostly concerned with the dynamics of the object grasped by the user; therefore the interaction of many rigid objects is not discussed here.

2.4.1 Penalty-Based Methods

When two objects touch or collide, collision response must be applied to prevent object interpenetration. One method for implementing collision response is the insertion of stiff springs at the points of contact [MW88]. This method is inspired by the fact that, when objects collide, small deformations take place at the region of contact, and these deformations can be modeled with springs, even if the objects are geometrically rigid.

Given two intersecting objects A and B, penalty-based collision response requires the definition of a contact point $\mathbf{p}$, a contact normal $\mathbf{n}$ and a penetration depth δ. The penalty-based spring force and torque applied to object A are defined as follows:

$$\begin{aligned} \mathbf{F}_A &= -f(\delta)\mathbf{n}, \\ \mathbf{T}_A &= (\mathbf{p}-\mathbf{c}_A)\times\mathbf{F}_A, \end{aligned} \tag{2.3}$$

where $\mathbf{c}_A$ is the center of mass of A. Opposite force and torque are applied to object B. The function f could be a linear function defined by a constant stiffness k or a more complicated non-linear function. It could also contain a viscous term, dependent on the derivative of the penetration depth.

The basic formulation of penalty methods can be modified slightly in order to introduce repulsive forces between objects, by inserting contact springs when the objects come closer than a distance tolerance d. In this way, object interpenetration occurs less frequently. The addition of a tolerance has two major advantages: the possibility of using penalty methods in applications that do not allow object interpenetration, and a reduction of the cost of collision detection. As noted in Sec. 2.3, computation

of penetration depth is notably more costly than computation of separation distance.

Penalty-based methods offer several attractive properties: the force model is local to each contact and computationally simple, object interpenetration is inherently allowed, and contact determination needs to be performed only once per simulation frame. This last property makes penalty-based methods best suited for interactive applications with fixed time steps, such as haptic rendering [MPT99, KOLM03, JW03] and games [Wu00, Lar01]. But penalty-based methods also have some disadvantages. There is no direct control over physical parameters, such as the coefficient of restitution. Non-penetration constraints are enforced by means of very high contact stiffness, and this circumstance leads to instability problems if numerical integration is executed using fast, explicit methods. The solution of penalty-based simulation using implicit integration, however, enhances stability in the presence of high contact stiffness [Wu00, Lar01].

Friction effects can be incorporated into penalty-based methods by means of localized force models that consider each contact point independently. Most local friction methods propose different force models for static or dynamic situations [Kar85, HA00]. Static friction is modeled by fixing adhesion points on the surfaces of the colliding objects and setting tangential springs between the contact points and the adhesion points. If the elastic friction force becomes larger than a threshold determined by the normal force and the friction coefficient, the system switches to dynamic mode. In the dynamic mode, the adhesion point follows the contact point. The system returns to static mode if the velocity falls under a certain threshold.

So far, I have analyzed contact determination and collision response as two separate problems, but the output of the contact determination step has a strong influence on the smoothness of collision response and, as a result, on the stability of numerical integration. As pointed out by Larsen [Lar01], when a new contact point is added, the associated spring must be unstretched. In other words, the penetration depth value must be zero initially and must grow smoothly. Fig. 2.1 shows a sequence of object configurations in which a small change in rotation produces a discontinuity in the point of contact. This situation translates into torque discontinuities, which can induce the object to oscillate. The existence of geometry-driven discontinuities is an inherent problem of penalty-based simulations with fixed time steps. Some authors [HS04] have proposed sampling the intersection volume to avoid geometric discontinuities in the application of penalty-based methods to rigid body simulation and

haptic rendering, but this approach is applicable only to very simple objects.

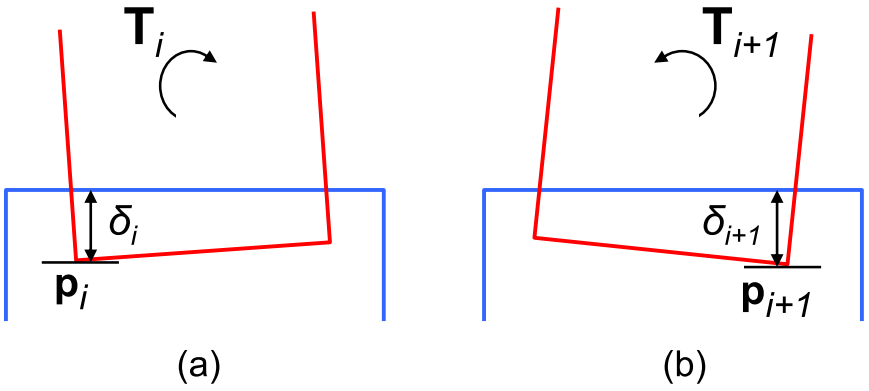

Figure 2.1: **Torque Discontinuity:** *(a) penetration depth and torque at time t_i, with contact point $\mathbf{p}_i$; (b) penetration depth and torque at time t_{i+1}, after the contact moves to contact point $\mathbf{p}_{i+1}$.*

2.4.2 Constraint-Based Simulation

Constraint-based methods for the simulation of rigid body dynamics handle all concurrent contacts in a single computational problem and attempt to find contact forces that produce physically and geometrically valid motions. Specifically, they integrate the Newton-Euler equations of motion (see Eq. 2.2), subject to geometric constraints that prevent object interpenetration. The numerical integration of Newton-Euler equations must be interrupted before objects interpenetrate. At a collision event, object velocities and accelerations must be altered, so that non-penetration constraints are not violated and numerical integration can be restarted. One must first compute contact impulses that produce constraint-valid velocities. Then, one must compute contact forces that produce valid accelerations.

The relative normal accelerations **a** at the points of contact can be expressed as linear combinations of the contact forces **F** (with constant matrix A and vector **b**). Moreover, one can impose non-penetration constraints on the accelerations and non-attraction constraints on the forces:

$$\mathbf{a} = A\mathbf{F} + \mathbf{b},$$
$$\mathbf{a} \geq 0, \quad \mathbf{F} \geq 0. \tag{2.4}$$

Baraff [Bar89] pioneered the application of constraint-based approaches to rigid body simulation in computer graphics. He posed constrained rigid body dynamics simulation as a quadratic programming problem on the contact forces, and he proposed a fast, heuristic-based solution for the frictionless case. He defined a quadratic cost function based on the fact that contact forces occur only at contact points that are not moving apart:

$$\min\left(\mathbf{F}^T\mathbf{a}\right) = \min\left(\mathbf{F}^T A\mathbf{F} + \mathbf{F}^T\mathbf{b}\right). \tag{2.5}$$

The quadratic cost function suggested by Baraff indicates that either the normal acceleration or the contact force should be 0 at a resting contact. As indicated by Cottle et al. [CpS92], this condition can be formulated as a linear complementarity problem (LCP). Baraff [Bar91, Bar92] added dynamic friction to the formulation of the problem and suggested approaches for static friction, as well as a solution following an algorithm by Lemke [Lem65] with expected polynomial cost in the number of constraints. Earlier, Lötstedt had studied the problem of rigid body dynamics with friction in the formulation of the LCP [Löt84]. Later, Baraff himself [Bar94] adapted an algorithm by Cottle and Dantzig [CD68] for solving frictionless LCPs to the friction case, and achieved linear-time performance in practice.

Stewart and Trinkle [ST96] presented an implicit LCP formulation of constraint-based problems. Unlike previous algorithms, which enforced the constraints only at the beginning of each time step, their algorithm solves for contact impulses that also enforce the constraints at the end of the time step. This formulation eliminates the need to locate collision events, but it increases the number of constraints to be handled, and it is unclear how it behaves with complex objects.

Stewart and Trinkle [ST96] mention the existence of geometry-driven discontinuities, similar to the ones appearing with penalty methods, in their implicit formulation of the LCP. After numerical integration of object position and velocities, new non-penetration constraints are computed. If numerical integration is not interrupted at collision events, the newly computed non-penetration constraints may not hold. Constraint violation may produce unrealistically high contact impulses and object velocities in the next time step. This phenomenon is equivalent to the effect of prestretched penalty-based springs described by Larsen [Lar01]. Stewart and Trinkle suggest solving a non-linear

complementarity problem, with additional cost involved.

If numerical integration is interrupted at collision events, the effects of geometry-driven discontinuities can be alleviated by capturing all the contact points that bound the contact region. Baraff [Bar89] considers polygonal contact regions between polyhedral models and defines contact constraints at the vertices that bound the polygonal regions. Similarly, Mirtich [Mir98a] describes polygonal contact areas as combinations of edge-edge and vertex-face contacts.

2.4.3 Impulse-Based Dynamics

Mirtich [MC95, Mir96] presented a method for handling collisions in rigid body dynamics simulation based solely on the application of impulses to the objects. In situations of resting, sliding, or rolling contact, constraint forces are replaced by trains of impulses. Mirtich defined a collision matrix that relates contact impulse to the change in relative velocity at the contact. His algorithm decomposes the collision event into two separate processes: compression and restitution. Each process is parameterized separately, and numerical integration is performed in order to compute the velocities after the collision. The parameterization of the collision event enables the addition of a friction model to instantaneous collisions.

The time-stepping engine of impulse-based dynamics is analogous to the one in constraint-based dynamics: numerical integration must be interrupted before interpenetration occurs, and valid velocities must be computed. One of the problems of impulse-based dynamics emerges during inelastic collisions from the fact that accelerations are not recomputed. The energy loss induced by a train of inelastic collisions reduces the time between collisions and increases the cost of simulation per frame. In order to handle this problem, Mirtich suggested the addition of unrealistic, but visually imperceptible, energy to the system when the microcollisions become too frequent.

As has been pointed out by Mirtich, impulse-based approaches are best suited for simulations that are collision-intensive, with multiple, different impacts occurring frequently. This dissertation focuses on haptic rendering for manipulation and exploration tasks, where contact often can be described as resting or sliding contact, not the best scenario for impulse-based dynamics.

2.5 Haptic Texture Rendering

Although haptic rendering of textures was one of the first tackled problems [MOYS+90], prior to the work presented in this dissertation it has been limited to the interaction between a probe point and a textured surface. I begin this section with a description of Minsky's pioneering algorithm for rendering textures on the plane [Min95]. Then I discuss rendering of textures on 3D surfaces, covering basic 3-DoF haptic rendering, height-field-based methods, and probabilistic methods.

2.5.1 Rendering Textures on the Plane

Minsky [Min95] developed the *Sandpaper* system for 2-DoF haptic rendering of textures on a planar surface. Her system was built around a force model for computing 2D forces from texture height field information. Following energy-based arguments, her force model synthesizes a force $\mathbf{F}$ in 2D based on the gradient of the texture height field h at the location of the probe:

$$\mathbf{F} = -k\nabla h. \tag{2.6}$$

Minsky also analyzed qualitatively and quantitatively roughness perception and the believability of the proposed force model. One of the main conclusions of her work is to establish her initial hypothesis, that texture information can be conveyed by displaying forces tangential to the contact surface. This hypothesis was later exploited for rendering textured 3D surfaces [HBS99].

2.5.2 3-Degree-of-Freedom Haptic Rendering

As described in Sec. 1.1.4, 3-DoF haptic rendering methods compute feedback force as a function of the separation between the probe point controlled with the haptic device and a contact point constrained to the surface of the haptically rendered object. Early 3-DoF haptic rendering methods set the contact point as the point on the surface of the object closest to the probe point. As has been addressed by Zilles and Salisbury [ZS95], these methods lead to force discontinuities and possible "pop-through" problems, in which the contact point jumps between opposing sides of the object. Instead, Zilles and Salisbury proposed the *god-object* method, which defines the computation of the contact point as a constrained optimization problem. The contact point is located at a minimum distance from the probe

point, but its interframe trajectory is constrained by the surface. Zilles and Salisbury solve the position of the contact point using Lagrange multipliers, once they define the set of active constraints.

Ruspini et al. [RKK97] followed a similar approach. They modeled the contact point as a sphere of small radius and solved the optimization problem in the configuration space. Ruspini and his colleagues also added other effects, such as force shading for rounding of corners (by modifying the normals of constraint planes), or friction (by adding dynamic behavior to the contact point).

2.5.3 Methods Based on Height Fields

High-resolution surface geometry can be represented by a parameterized coarse mesh along with texture images storing detailed height field or displacement field information, similarly to the common approach of texture mapping in computer graphics [Cat74]. Constraint-based 3-DoF haptic rendering methods determine a unique contact point on the surface of the rendered object. Usually, the mesh representation used for determining the contact point is rather coarse and does not capture high-frequency texture. Nevertheless, the parametric coordinates of the contact point can be used for accessing surface texture information from texture images.

Ho et al. [HBS99] introduced a technique similar to bump mapping [Bli78] that alters the surface normal based on the gradient of the texture height field. A combination of the original and refined normals is used for computing the direction of the feedback force.

Techniques for haptic texture rendering based on a single contact point can capture geometric properties of only one object and are not suitable for simulating full interaction between two surfaces. The geometric interaction between two surfaces is not limited to, and cannot be described by, a pair of contact points. Moreover, the local kinematics of the contact between two surfaces include rotational degrees of freedom, which are not captured by point-based methods.

Ho et al. [HBS99] indicate that a high height field gradient can induce system instability. Along a similar direction, Choi and Tan [CT03b, CT03a] have studied the influence of collision detection and penetration depth computation on 3-DoF haptic texture rendering. Discontinuities in the output of collision detection are perceived by the user, a phenomenon that they describe as *aliveness*. This phenomenon is a possible problem in 6-DoF haptic rendering too.

2.5.4 Probabilistic Methods

Some researchers have exploited statistical properties of surfaces for computing texture-induced forces that are added to the classic 3-DoF contact forces. Siira and Pai [SP96] synthesized texture forces according to a Gaussian distribution for generating a sensation of roughness. In order to improve stability, they did not apply texture forces during static contact. Later, Pai et al. [PvdDJ+01] presented a technique for rendering roughness effects by dynamically modifying the coefficient of friction of a surface. The roughness-related portion of the friction coefficient was computed according to an autoregressive process driven by noise.

Probabilistic methods have proved to be successful for rendering high-frequency roughness effects in point-surface contact. It is also possible, although this approach has yet to be explored, that they could be combined with geometric techniques for synthesizing high-frequency effects in 6-DoF haptic rendering.

2.6 6-Degree-of-Freedom Haptic Rendering

The problem of 6-DoF haptic rendering has been studied by several researchers. As introduced in Sec. 1.2.1, the existing methods for 6-DoF haptic rendering can be classified into two large groups based on their overall pipelines: *direct rendering* methods and *virtual coupling* methods. Each group of methods presents some advantages and disadvantages. Direct rendering methods are purely geometric, and there is no need to simulate the rigid body dynamics of the grasped object. However, penetration values may be quite large and visually perceptible, and system instability can arise if the force update rate drops below the range of stable values. Virtual coupling methods enable reduced interpenetration, higher stability, and higher control of the displayed stiffness. However, virtual coupling [CSB95] may introduce noticeable filtering, both tactile and visual, and it requires the simulation of rigid body dynamics.

The different 6-DoF haptic rendering methods propose a large variety of options for solving the specific problems of collision detection, collision response, and simulation of rigid body dynamics. In the presence of infinite computational resources, an ideal approach to the problem of 6-DoF haptic rendering would be to compute the position of the grasped object using constraint-based rigid body

dynamics simulation [Bar92] and to implement force feedback through virtual coupling. This approach has indeed been followed by some, but it imposes serious limitations on the complexity of the objects and contact configurations that can be handled interactively. I now discuss briefly the different existing methods for 6-DoF haptic rendering, focusing on those that have been applied to moderately complex objects and scenarios.

2.6.1 Direct Haptic Rendering Approaches

Gregory et al. [GME+00] presented a 6-DoF haptic rendering system that combined collision detection based on convex decomposition of polygonal models [EL01], predictive estimation of penetration depth, and force and torque interpolation. They were able to handle interactively dynamic scenes with several convex objects, as well as pairs of non-convex objects with a few hundred triangles and rather restricted motion. Kim et al. [KOLM03] exploited convex decomposition for collision detection and incorporated fast, incremental, localized computation of per-contact penetration depth [KLM02]. In order to improve stability and eliminate the influence of triangulation on the description of the contact manifold, they introduced a contact clustering technique. Their system was able to handle pairs of models with nearly one hundred convex pieces each interactively.

Earlier, Nelson et al. [NJC99] introduced a technique for haptic interaction between pairs of parametric surfaces. Their technique tracks contact points that realize locally maximum penetration depth during surface interpenetration. Tracking contact points, instead of recomputing them for every frame, ensures smooth penetration values, which are used for penalty-based force feedback. The contact points are solved in parametric space, and they are defined as those pairs of points for which their difference vector is collinear with surface normals.

Johnson and Willemsen [JW03] suggested a technique for polygonal models that defines contact points as those that satisfy a local minimum-distance criterion, according to Nelson's definition [NJC99]. Johnson and Willemsen exploit this definition in a fast collision culling algorithm, using spatialized normal cone hierarchies [JC01]. The performance of their technique depends on the convexity and triangulation of the models, which affect the number of contact points. Recently, Johnson and Willemsen [JW04] have incorporated an approximate but fast, incremental contact-point-tracking algorithm that is combined with slower exact collision updates from their previous technique [JW03].

This algorithm handles models with thousands of triangles at interactive rates, but the forces may suffer discontinuities if the exact update is too slow.

2.6.2 Virtual Coupling with Object Voxelization

In 1999, McNeely et al. [MPT99] presented a system for 6-DoF haptic rendering that employs a discrete collision detection approach and virtual coupling. The system is intended for assembly and maintenance planning applications and assumes that only one of the objects in the scene is dynamic. The surfaces of the scene objects are voxelized, and the grasped object is point-sampled. The collision detection module checks for inclusion of the sample points in the scene voxels, and then a local force model is applied. Hierarchical culling of sample points is possible, but ultimately the computational cost depends on the number of contact points. This system has been integrated in a commercial product, VPS, distributed by Boeing.

McNeely and his colleagues introduced additional features in order to alleviate some of the limitations. Surface objects are voxelized only on the surface, therefore deep penetrations, which can occur if objects collide at high velocities, cannot be handled. They propose pre-contact braking forces, similar to the braking impulses suggested by Salcudean [SV94], for reducing the contact velocity of the grasped object and thereby preventing deep penetrations. The existence of multiple contact points produces high stiffness values that can destabilize the simulation of rigid body dynamics. They propose averaging the effects of the different contact points before contact forces are applied to the grasped object, for limiting the stiffness and thereby ensuring stable simulation. The locality of the force model induces force discontinuities when contact points traverse voxel boundaries. They point out that force discontinuities are somewhat filtered by the virtual coupling. Renz et al. [RPP+01] modified McNeely's local force model to ensure continuity of the surface across voxel boundaries, but incurring more expensive force computation.

Using the same voxelization and point-sampling approach for collision detection, Wan and McNeely [WM03] have proposed a novel solution for computing the position of the grasped object. The early approach by McNeely et al. [MPT99] computed object dynamics by explicit integration of Newton-Euler equations. Instead, Wan and McNeely [WM03] presented a purely geometric solution that eliminates the instability problems that can arise due to high contact stiffness. Their algorithm

formulates linear approximations of the coupling and contact force and torque in the space of translations and rotations of the grasped object. The state of the object is computed at every frame by solving for the position of quasi-static equilibrium. Deep penetrations are avoided by formulating the coupling force as a non-linear spring.

2.6.3 Rigid Body Dynamics with Haptic Feedback

Chang and Colgate [CC97] proposed a solution to 6-DoF haptic rendering by combining virtual coupling [CSB95] and rigid body simulation based on impulse dynamics [Mir96]. They found that impulses alone were not efficient in resting contact situations, and in those cases they suggested a combination of impulses and penalty forces. Recently, Constantinescu et al. [CSC04] have reached a similar conclusion. As has been addressed by Constantinescu, combining impulses and penalty forces requires a state machine in order to determine the state of the object, but it is not clear how to extend this solution to scenes with many contacts. Both Chang and Constantinescu have tested their implementations only on simple benchmarks.

One of the reasons for the simplicity of Chang and Constantinescu's benchmarks is the cost of collision detection for the simulation of rigid body dynamics. As has been discussed in Sec. 2.4, impulse- [Mir96] or constraint-based [Bar92] methods must interrupt the integration before object interpenetration, and this leads to many collision queries per frame. Some researchers have integrated haptic interaction with constraint-based rigid body simulations [Ber99, RK00] in scenes with simple geometry.

As indicated in Sec. 2.4.1, non-penetration constraints can be relaxed using penalty-based methods. McNeely et al. [MPT99] employed penalty methods for rigid body simulation but, as explained earlier, they observed numerical instabilities due to high stiffness values, and large interpenetrations under high impact velocities. Those problems can be tackled with high-stiffness penalty contact forces along with implicit integration, an approach used in interactive rigid body simulations [Wu00, Lar01]. Implicit integration requires the evaluation of the Jacobian of the Newton-Euler equations and the solution of a linear system of equations [BW98]. As will be presented in Chapter 5, implicit integration can be performed at force update rates under the assumption that only the grasped object is dynamic.

Chapter 3

Contact Levels of Detail

Collision detection is the first step in displaying force and torque between two 3D virtual objects in 6-DoF haptic rendering. Many practical techniques and theoretical advances for collision detection have been developed (see surveys by Lin and Gottschalk [LG98], Klosowski et al. [KHM+98] and Lin and Manocha [LM04]). Yet, despite the huge body of literature, the existing collision detection algorithms cannot offer the desired performance for haptic rendering of moderately complex contact configurations.

Model simplification has been an active research area for the past decade. Applications of mesh simplification algorithms to the problem of collision detection can potentially accelerate collision queries. However, to date only relatively simple algorithms for convex polytopes have been proposed. A simple approach would be to generate a series of simplified representations, also known as levels of detail (LODs), and use them directly for collision detection. But, collision queries require auxiliary data structures, such as bounding volume hierarchies (BVHs) or spatial partitioning, in order to achieve good runtime performance.

In this chapter I introduce *contact levels of detail* (CLODs), a multiresolution collision detection algorithm that integrates BVHs and LODs in one single *dual hierarchy*. I describe a general data structure that combines static LODs and BVHs. Descending on the BVH has the additional effect of refining LODs, in order to perform multiresolution collision detection and select the appropriate object resolution at each contact location, as shown in Fig. 3.1. Findings from tactual perception and spatial recognition [KL95, OC99, OC01] demonstrate that large contact areas reduce the perceptibility of fine surface features.

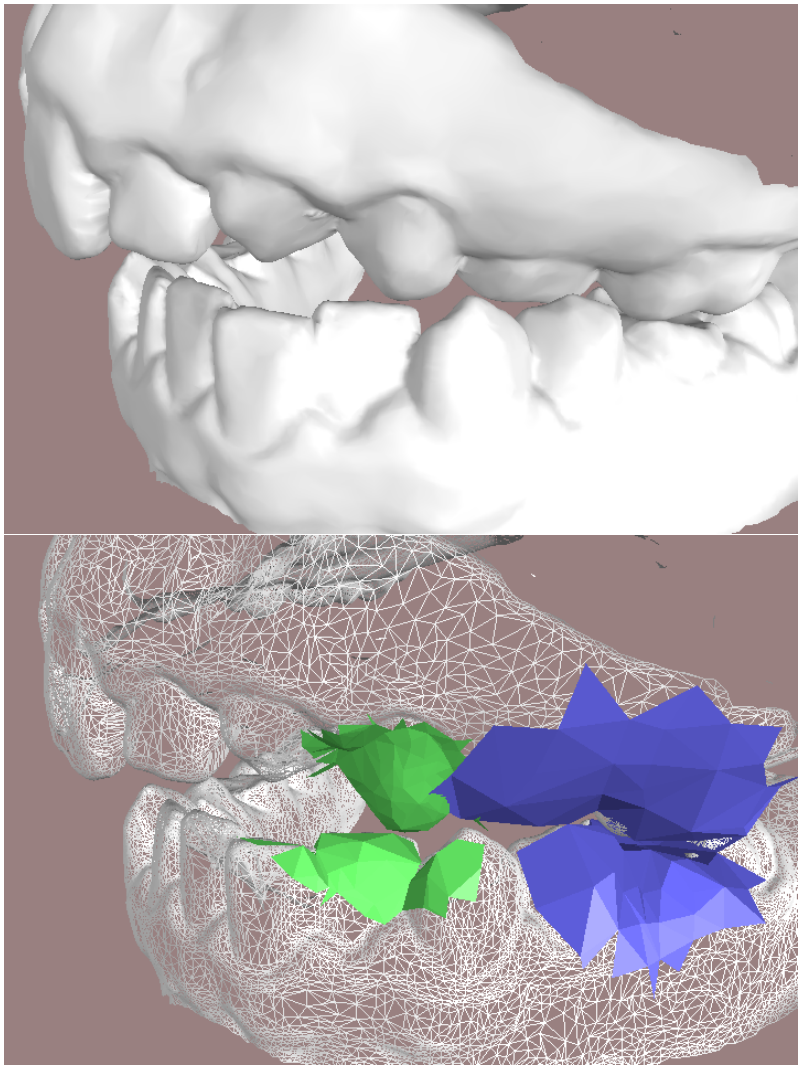

Figure 3.1: **Multiresolution Collision Detection Using CLODs.** *Top: Moving jaws in contact, rendered at their highest resolution; Bottom: The appropriate object resolution (shown in blue and green) is adaptively selected at each contact location, while the finest resolution is displayed in wireframe.*

The practical implementation of CLODs involves many design decisions. One of them is the type of bounding volume (BV) for the BVH. I have selected convex hulls as the BVs for my implementation of CLODs because of their qualities for providing rich contact information between polygonal models. The construction of the CLOD hierarchy with convex hulls follows a *sensation preserving simplification* process. In this process, atomic simplification and filtering operations are combined with merging of convex BVs. In particular, I have designed an atomic operation *filtered edge collapse*

that performs mesh decimation and filtering subject to convexity constraints.

At runtime, multiresolution collision detection using CLODs proceeds by traversing BVHs. A pair of BVs is first tested for collision and, if collision occurs, a selective refinement test is applied. I have designed various error metrics for selective refinement: a haptic metric based on the relationship between contact area and resolution, a view-dependent metric, and a velocity-dependent metric. All error metrics account for surface deviation between coarse CLODs and the full-resolution objects.

I have applied CLODs to both 6-DoF haptic rendering and rigid body simulation, and I have carried out experiments to test the performance in each case. In 6-DoF haptic rendering of challenging contact scenarios using CLODs I have observed up to 2-orders-of-magnitude performance improvement over exact collision detection methods with little degradation in the contact forces.

This chapter compiles work and results previously published in [OL03b] and [OL03a]. The chapter is organized as follows. Section 3.1 presents the motivation and design goals of a multiresolution collision detection algorithm for haptic rendering. In Section 3.2, I introduce the novel data structure of CLODs, and in Section 3.3, I present a particular implementation based on convex hulls, followed by the description of sensation preserving simplification. In Section 3.4, I explain how contact levels of detail are used in runtime collision detection. In Section 3.5, I present the experiments and results, and I conclude the chapter in Section 3.6 with a summary and discussion of limitations.

3.1 Foundations and Objectives of Contact Levels of Detail

In this section, I first connect important findings from studies on tactual perception to the design of CLODs. Then, I describe the requirements for haptic rendering and the design goals.

3.1.1 Haptic Perception of Surface Detail

In Sec. 2.1.1, I summarize perceptual studies on tactile feature identification that lead to the conclusion that human haptic perception of the existence of a geometric surface feature depends on the ratio between the contact area and the size of the feature, not the absolute size of the feature itself. The *size of a feature* is broadly defined here as width $\times$ length $\times$ height. The width and length of a feature can be intuitively considered as the “inverse of resolution” (formally defined in Sec. 3.3) of a polygonal

model. That is, higher resolution around a local area implies that the width and length of the geometric surface features in that neighborhood are smaller, and vice versa. The concept of “height” is extended to describe the amount of surface deviation between polygonal representations of a model at different resolutions.

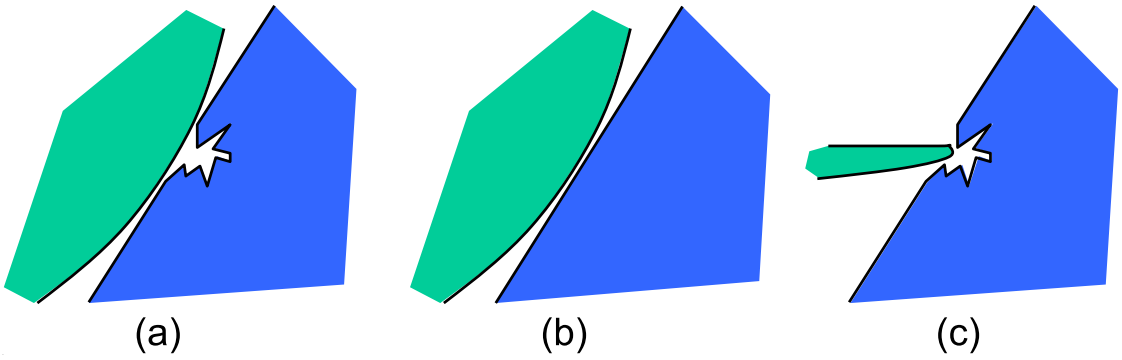

Figure 3.2: **Contact area and resolution:** *(a) high-resolution model with large contact area; (b) low-resolution model with large contact area; (c) high-resolution model with small contact area.*

Figure 3.2 illustrates the observation that relates contact area and perceptibility of features. The contact between two objects typically occurs along a certain contact area. With polygonal models, the contact area may be described by multiple contact points. The number of contact points grows if the objects are described at a higher resolution. Increasing the resolution beyond a sufficiently large value, however, may have little effect on the forces computed between the objects, because these forces are computed as a sum of contact forces arising from a net of contact points. One can argue that, intuitively, a larger contact area allows the objects to be described at a coarser resolution.

The conclusions drawn from perceptual studies set the basis for error metrics in haptic rendering. The minimum acceptable resolution to represent an object will be governed by the relationship between surface deviation and contact area. The haptic error metrics proposed in this dissertation differ notably from visual error metrics in the mesh simplification literature [Hop97, LE97] and from metrics of visual collision perception [OD01]. In visual rendering, the resolution required to represent an object is based on a combination of surface deviation (or Hausdorff distance) and the viewing distance to the object. I will show how haptic error metrics drive the offline construction of CLODs in Sec. 3.3, and runtime contact queries in Sec. 3.4.

3.1.2 Hierarchical Collision Detection

The running time of any collision detection algorithm depends on both the input and output sizes of the problem. Given two polyhedra, characterized by their combinatorial complexity of m and n polygons, the collision detection problem can have an output size as large as $O(mn)$. Please refer to Sec. 2.3 for a summary of techniques for collision detection.

Bounding volume hierarchies (BVHs) are commonly used for accelerating collision detection between general geometric objects. As described in Sec. 2.3.3, a collision query between two objects is performed by recursively traversing their BVHs in tandem. The test between the two BVHs can be described by the *bounding volume test tree* (BVTT) [LGLM00], a tree structure that holds in each node the result of the query between two BVs. In situations with temporal coherence, collision tests can be accelerated by *generalized front tracking* (GFT)[EL01]. GFT caches the front of the BVTT where the result of the queries switches from true to false for initializing the collision query in the next time step. The overall cost of a collision test is proportional to the number of nodes in the front of the BVTT.

When large areas of the two objects are in close proximity, a larger portion of the BVTT front is close to the leaves, and it consists of a larger number of nodes. The size of the front also depends on the resolutions with which the objects are modeled; higher resolutions imply a deeper BVTT. To summarize, the cost of a collision query depends on two key factors: the size of the contact area and the resolutions of the models. As observed in Sec. 3.1.1, however, a larger contact area allows the objects to be described at a coarser resolution, therefore reducing the cost of collision queries. The foundation of CLODs is to exploit the relationship between contact area and object resolution to achieve nearly constant cost in collision queries. The concept of CLODs consists of creating multiresolution representations of the objects and selecting the appropriate level of detail (i.e., resolution) for each object at each contact location independently.

3.1.3 Design Requirements and Desiderata

Efficient multiresolution collision detection depends on two main objectives:

1. Create **accurate multiresolution representations**.

2. Embed the multiresolution representations in **effective bounding volume hierarchies**.

Multiresolution representations are often created by decimating the given polyhedral models. Difficulties arise when trying to embed these representations in BVHs. Considering each LOD of the given object as one whole model, each LOD would require a distinct BVH for collision detection. This requirement would result in inefficient collision queries, because the front of the BVTT would have to be updated for the BVH of each LOD. Instead, I introduce a procedure to create one unique dual hierarchical representation, denoted as *contact levels of detail*, that serves as both a multiresolution representation and a BVH.

On the one hand, this novel dual hierarchy constitutes a multiresolution representation built according to haptic error metrics. This feature enables reporting results of contact queries accurate up to some haptic tolerance value. On the other hand, the dual hierarchy constitutes a BVH that enables effective collision detection. Thanks to the dual nature of the data structure, using CLODs in haptic rendering helps to speed up contact queries while maintaining haptic error tolerances.

In Sec. 3.2, I describe the generic data structure employed in CLODs. The implementation of CLODs, however, requires the selection of one particular type of bounding volume. As discussed later in Sec. 3.3, I have opted for convex hulls as the bounding volumes.

To summarize, my goal in the design of CLODs has been to create **dual multiresolution hierarchies** that:

1. **Minimize perceptible surface deviation.** I achieve this goal by filtering the detail at appropriate resolutions and by using a novel sensation preserving refinement test for collision detection;

2. **Reduce the polygonal complexity of low-resolution representations.** This objective is achieved by incorporating mesh decimation into the creation of the hierarchy;

3. **Are themselves BVHs of convex hulls.** I perform a surface convex decomposition of the given triangular mesh and maintain it across the hierarchy. The convex surface decomposition places both local and global convexity constraints on the mesh decimation process.

The data structure for CLODs imposes no constraints on the input models, but the implementation of CLODs based on convex hulls requires the input models to be represented as oriented 2-manifold

triangular meshes (with or without boundaries).

3.2 Data Structure

Prior to describing the data structure for CLODs, I introduce some notation to be used throughout the chapter. Then I describe the data structure, I discuss its interpretation as a multiresolution representation, and I overview the process of building generic CLODs.

3.2.1 Notation

In the remaining of this chapter, I use bold-face letters to distinguish a vector (e.g., a point, normal, etc.) from a scalar value. In Table 3.1, I enumerate some of the notations I use throughout the chapter.

Notation	Meaning
r, r_i, r_j	Different resolutions
M_k	An LOD of a mesh M with resolution r_k
c_i	A cluster of triangles (or, specifically, a convex surface patch)
C_i	The BV of a cluster c_i (or, specifically, the convex hull)
a, b	BVs involved in collision detection
ab	A node of the BVTT, composed of BVs a and b
q	A distance query between two BVs
Q	A contact query between two objects, which consists of multiple distance queries q
d	Distance tolerance of a contact query
ϕ	Error function of a CLOD
h	Hausdorff distance
s, s_a, s_b	Surface deviations
D, D_a, D_b	Contact areas
$\mathbf{v}, \mathbf{v_1}, \mathbf{v_2}$	Vertices of a mesh
$e(\mathbf{v_1}, \mathbf{v_2})$	An edge between two vertices

Table 3.1: **Notation Table**

3.2.2 Description of the Data Structure

Assuming that an input model is described as a triangle mesh M_0, the data structure for CLODs is composed of:

- A sequence of LODs $\{M_0, M_1, ..., M_{n-1}\}$, where M_{i+1} is obtained by applying simplification operations to and removing high-resolution geometric detail from M_i.
- For each LOD M_i, a partition of the triangles of M_i into disjoint clusters $\{c_{i,0}, c_{i,1}, ..., c_{i,m}\}$.
- For each cluster $c_{i,j}$, a bounding volume $C_{i,j}$.
- A tree T formed by all the BVs of clusters, where BVs of clusters in M_i are children of BVs of clusters in M_{i+1}, and all the BVs except the ones corresponding to M_0 have at least one child.
- For every BV, $C_{i,j}$, the maximum directed Hausdorff distance $h(C_{i,j})$ from its descendant BVs.

The tree T of BVs, together with the Hausdorff distances, serves as the BVH for culling purposes in collision detection. Directed Hausdorff distances are necessary because, in the definition of CLODs, the set of BVs associated with one particular LOD may not bound the surface of previous LODs. Hausdorff distances are used to perform conservative collision tests, as will be later explained in Sec. 3.4.2.

An additional constraint is added to the data structure, such that the coarsest LOD, M_{n-1}, is partitioned into one single cluster $c_{n-1,0}$. Therefore, the root of the BVH will be the BV of the coarsest LOD. Descending to the next level of the hierarchy will yield the children BVs, whose union encloses the next LOD. At the end of the hierarchy, the leaf BVs will enclose the original surface M_0.

Multiresolution Interpretation

The CLODs of a given object comprise a multiresolution representation of that object. More specifically, they constitute a sequence of static LODs, each of which approximates the original triangular mesh at a different resolution.

Conceptually, an LOD M_j at resolution r_j of a mesh M_0 can be obtained from an LOD M_i at a higher resolution r_i by removing detail at resolutions in the range $[r_j, r_i]$. As a conclusion, an LOD at resolution r_j preserves the lower resolution geometric information while the higher resolution detail might have been culled away. The detail that is removed introduces some surface deviation respect to the original mesh, which is quantified by Hausdorff distances in the CLOD data structure.

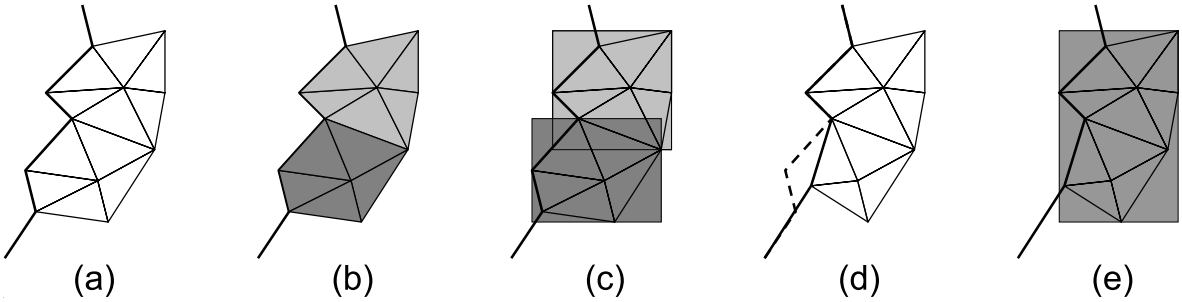

Figure 3.3: **Construction of generic CLODs:** *(a) Initial surface; (b) Clusters of triangles; (c) BVs for each cluster; (d) Mesh simplification; (e) BV of the union of clusters after some conditions are met.*

3.2.3 Generic Construction of Contact Levels of Detail

The process of creating the CLODs, depicted in Fig. 3.3, starts by grouping the triangles of the original surface into clusters. The sizes and properties of these clusters depend on the type of BV that is used for the BVH, and will be such that the performance of the collision query between two BVs is optimized. The next step in the creation of CLODs is to compute the BV of each cluster. This initialization is followed by a mesh decimation process along with bottom-up construction of the BVH, carried out by merging clusters and computing the BV of their union.

The atomic simplification operations need to satisfy the following conditions:

- **Constraints imposed by the BVH:** The containment of surface geometry inside the BVs has to be preserved after each simplification operation. This condition may impose topological and/or geometric constraints.

- **Design requirements to achieve better efficiency:** Clusters can be joined when certain conditions are met. The BVH will be more effective in collision pruning if these conditions are taken into account when designing the atomic simplification operations.

In Sec. 3.3, I present a sensation preserving simplification process that results in a hierarchy of CLODs of convex hulls. I also describe the atomic simplification operations and the constraints imposed by the selection of convex hulls as the BVs.

3.3 Sensation Preserving Simplification

In this section I describe *sensation preserving simplification*, the process for creating CLODs of convex hulls. For rigid bodies, this process can be completed offline as a preprocessing step. I first discuss the selection of convex hulls as the bounding volumes and I define the concept of resolution in the context of CLODs. Next, I present the detailed process of sensation preserving simplification, followed by a description of the atomic simplification operation *filtered edge collapse*. I end this section presenting some examples of CLODs.

3.3.1 Selection of Convex Hulls as Bounding Volumes

Overlap tests between convex hulls can be executed in expected constant time with motion coherence [LC91, Mir98b, GHZ99, EL00]. Furthermore, convex hulls provide superior fitting to the underlying geometry than OBBs [GLM96] or k-DOPs [KHM+98]. The fitting property is related to the performance of proximity queries that return distances, contact points, or contact normals. At runtime collision detection, contact information must be obtained between CLODs at the appropriate resolution. That operation implies getting contact information from the triangles of the specific CLODs. If the BVs are such as AABBs, OBBs, or k-DOPs the efficiency of getting contact information using triangles is related to the number of triangles in each cluster. With convex hulls, however, if the clusters are themselves convex surface patches, contact information at triangle level is obtained practically for free when performing the query between BVs [EL01].

I define the clusters of the initial mesh M_0 as the surface patches of its convex surface decomposition [CDST97, EL01], in order to maximize the efficiency of runtime collision detection using convex hulls as BVs. I follow the definition of convex surface patches by Ehmann and Lin [EL01], which imposes two types of convexity constraints on the process of creating CLODs:

- **Local constraints**: the interior edges of convex patches must remain convex after simplification operations are applied.
- **Global constraints**: the enclosing convex hulls cannot protrude the surface of the object.

Note that convex hulls limit the types of models that can be handled. Convex surface decomposi-

tion requires the input models to be described as 2-manifold, oriented triangle meshes.

3.3.2 Definition and Computation of Resolution

Before I explain how to generate each LOD, I define *resolution* in the context of CLODs. I follow the framework of signal processing for irregular meshes and I assume that a triangular mesh M can be considered as a sampled version of a smooth surface S, which has been reconstructed via linear interpolation. The vertices of the mesh are samples of the original surface while edges and faces are the result of the reconstruction.

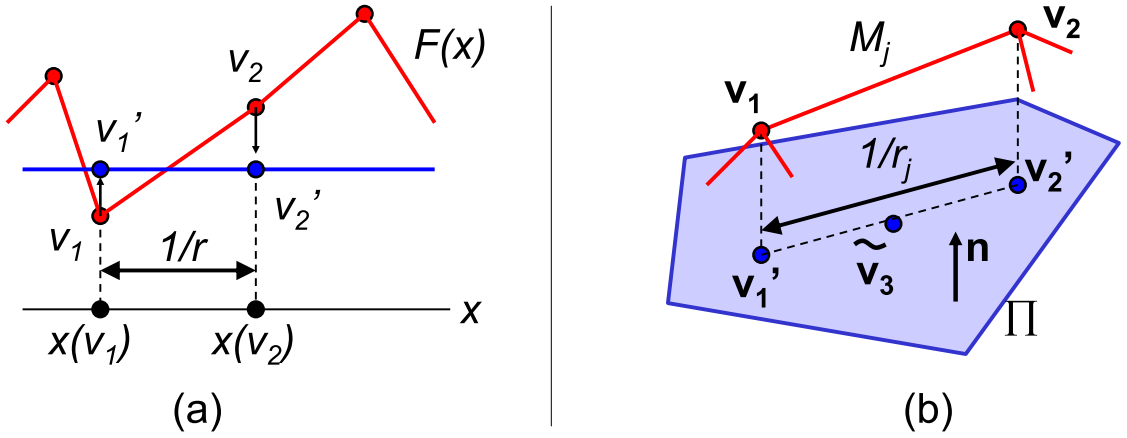

Figure 3.4: **Definition of Resolution.** *(a) Resolution r defined in the 1D setting; (b) Resolution for irregular meshes.*

My definition of sampling resolution for irregular meshes is inspired by the 1D setting. For a 1D function $F(x)$, the sampling resolution r is the inverse of the distance between two subsequent samples on the real line. This distance can also be interpreted as the projection of the segment between two samples of the function, v_1 and v_2, onto the average value of the function, as shown in Fig. 3.4-a. The average value is the low-resolution representation of the function itself and can be obtained by low-pass filtering. Extending this idea to irregular meshes, the sampling resolution of an edge $(\mathbf{v_1}, \mathbf{v_2})$ of the mesh M at resolution r_j, M_j, can be estimated as the inverse of the projected length of the edge onto a low-resolution representation of the mesh, M_{j+1}. The definition of resolution for irregular meshes is depicted in Fig. 3.4-b.

I compute the low-resolution mesh M_{j+1} locally by filtering the mesh M_j, applying the filtered

edge collapse operation to the edge $(\mathbf{v_1}, \mathbf{v_2})$. Then, I compute the normal $\mathbf{n}$ of the resulting vertex $\tilde{\mathbf{v}}_\mathbf{3}$ by averaging the normals of incident triangles. Finally, the edge is projected onto the tangent plane Π defined by $\mathbf{n}$. The resolution r is computed as the inverse of the length of the projected edge.

$$r = \frac{1}{\|(\mathbf{v_1} - \mathbf{v_2}) - ((\mathbf{v_1} - \mathbf{v_2}) \cdot \mathbf{n}) \cdot \mathbf{n}\|}. \tag{3.1}$$

3.3.3 Construction of Contact Levels of Detail

The construction of CLODs of convex hulls is initialized by performing a convex surface decomposition of the input object and computing the convex hulls of the resulting convex patches. This is followed by a simplification loop, in which atomic simplification operations are combined with merging of convex hulls.

The atomic simplification operations must take into account the convexity constraints. After each operation, the union of every pair of neighboring convex patches is tested for convexity. If the union is a valid convex patch itself, the involved patches are merged and the convex hull of the union is computed. All the BVs in LOD M_j that are merged to a common BV $C_{j+1} \in M_{j+1}$ during sensation preserving simplification will have C_{j+1} as their parent in the BVH. I have chosen to output a new LOD every time that the number of convex patches is halved.

Ideally, the process will end with one single convex patch, which serves as the root for the BVH. However, this result is rarely achieved in practice, due to topological and geometric constraints that limit the amount of simplification, and which cannot be removed by local operations. In such cases, the hierarchy is completed by unconstrained pairwise merging of patches[EL01]. The levels of the hierarchy created in this manner, denoted as "free" LODs, cannot be used to report contact information in multiresolution collision queries, but are necessary to complete the BVH.

Algorithm 3.3.1 gives the pseudo code for the process of sensation preserving simplification.

Design Options Related to the Simplification Operations

Various steps of the simplification process that are central to the construction of CLODs need to be defined:

```
Compute surface convex decomposition
n = number of convex patches
Compute resolution of edges
Output initial LOD
Initialize edges as valid
Create priority queue
while Valid(Top(queue)),
   if FilteredEdgeCollapse(Top(queue)) then
      PopTop(queue)
      Recompute resolution of affected edges
      Reset affected edges as valid
      Update priority of affected edges
      Attempt merging of convex patches
   else
      Set Top(queue) as invalid
      Update priority of Top(queue)
   endif
   if Number of patches ≤ n/2 then
      Output new LOD
      n = number of convex patches
   endif
endwhile
while Number of patches > 1,
   Binary merge of patches
endwhile
```

ALGORITHM 3.3.1: Pseudo Code of the Sensation Preserving Simplification Loop

- The type of atomic simplification operation
- The assignment of priorities for simplification
- The local retriangulation after each atomic simplification operation

I have selected edge collapse as the atomic simplification operation for two main reasons:

1. Edge collapse, accompanied by the pertinent self-intersection tests, can guarantee preservation of topology, which is a requirement for maintaining a surface convex decomposition of the object during the construction of the hierarchy.
2. Topologically, an edge collapse can be regarded as a local downsampling operation, in which two samples (i.e., vertices) are merged into a single one.

There are many possible approaches for prioritizing edges and selecting the position of the resulting vertices: minimization of energy functions[Hop96], optimization approaches[LT98], quadric error metrics for measuring surface deviation[GH97], and more. None of these approaches, however, meets the convexity constraints or takes into account the factors that maximize the efficiency of CLODs. Another possibility is to employ the *progressive hulls* representation [SGG+00], which maintains the interesting property of containment for collision detection. In the progressive hulls representation, however, successive LODs are not simply multiresolution representations of the initial mesh, because they undergo an enlargement process that can result in noticeable visual artifacts. Instead, I have designed the local simplification operation *filtered edge collapse*, inspired by multiresolution analysis and signal processing of meshes.

Resolution and the Simplification Process

As discussed in Sec. 3.2.2, an LOD M_j can be defined as the approximation of a mesh M_0 that stores all the surface detail at resolutions lower than r_j. Following this definition, I have decided to prioritize the edges to be collapsed based on their resolution.

In the construction of CLODs, and as part of the initialization, I compute the resolution of all edges, set them as valid for collapse, and insert them in a priority queue. At each simplification step, I attempt to collapse the edge with highest priority (i.e., highest resolution). If an edge collapse is successful, the affected edges update their resolutions and priorities, and they are reset as valid for collapse.

Each LOD M_j is also assigned an associated resolution r_j. This value is the coarsest resolution of all edges collapsed before M_j is generated. Geometrically, it means that the LOD M_j preserves all the detail of the original mesh at resolutions coarser than r_j.

In sensation preserving simplification for haptic rendering, the goal is to maximize the resolution at which LODs are generated. As explained in Sec. 3.4, the perceptual error for haptic rendering is measured by taking into account the resolution of the surface detail that is culled away. Multiresolution contact queries will terminate faster as a result of maximizing the resolution at which LODs are generated. This is the basis for selecting edge resolution as the priority for edge collapses.

3.3.4 Filtered Edge Collapse

In the construction of CLODs, I aim to:

1. Generate multiresolution representations with low polygonal complexity at low resolution, for accelerating contact queries;

2. Filter detail as low-resolution LODs are computed. This approach allows more aggressive simplification and enables faster merging of convex patches to build the BVH.

These two goals are achieved by merging downsampling and filtering operations in one atomic operation, denoted *filtered edge collapse*. This operation is composed of the following steps:

1. A topological edge collapse. An edge $(\mathbf{v_1}, \mathbf{v_2})$ is first topologically collapsed to a vertex $\hat{\mathbf{v}}_\mathbf{3}$. This step provides the downsampling.

2. An initialization process that sets the position of $\hat{\mathbf{v}}_\mathbf{3}$ using quadric error metrics [GH97].

3. Unconstrained relaxation to a position $\tilde{\mathbf{v}}_\mathbf{3}$, using Guskov's minimization of second order divided differences[GSS99].

4. The solution of an optimization problem in order to minimize the distance of the vertex to its unconstrained position, while taking into account the local convexity constraints.

5. A bisection search between the initial position of the vertex and the position that meets the local constraints, in order to find a location where self-intersection constraints and global convexity constraints are also met.

The unconstrained relaxation step resembles, intuitively, the minimization of dihedral angles, without much affecting the shape of the triangles [GSS99]. I have also tried other filtering techniques, such as those proposed by Taubin [Tau95], with very similar results. The selection of Guskov's filtering approach is consistent with the selection of the tangent plane of the filtered mesh as the low-resolution representation for the computation of resolutions, because linear functions are invariant under the minimization of second order differences. Next, I will describe in more detail how the convexity constraints are satisfied.

Local Convexity Constraints

Let $e \equiv (\mathbf{v_1}, \mathbf{v_2})$ be a candidate edge for filtered edge collapse. Let $\mathbf{v_3}$ represent the vertex resulting from the edge collapse. As a result of the collapse, the edges in the 1-ring neighborhood of $\mathbf{v_3}$ may change from convex to reflex and vice versa. Interior edges of convex patches are convex before the filtered edge collapse and must remain convex after it. These constraints can be expressed as linear constraints in the position of $\mathbf{v_3}$.

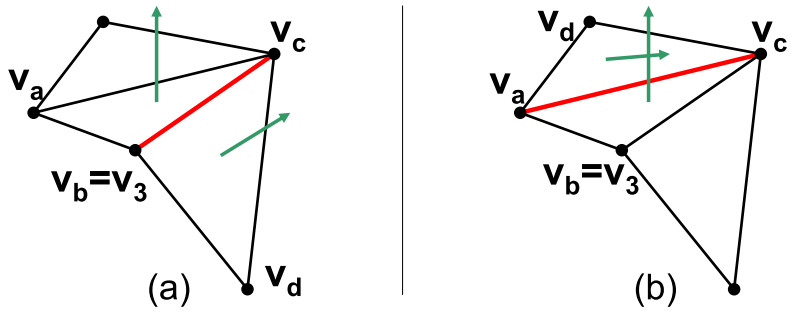

Figure 3.5: **Local Convexity Constraints.** *Assignment of vertices* $\mathbf{v_a}$, $\mathbf{v_b}$, $\mathbf{v_c}$ *and* $\mathbf{v_d}$ *for an interior edge (a) incident on* $\mathbf{v_3}$*, and (b) opposite to* $\mathbf{v_3}$.

Given e, the edge to be collapsed, two possible types of interior edges of convex patches exist: edges incident to $\mathbf{v_3}$ and edges opposite to $\mathbf{v_3}$, as shown in Fig. 3.5. However, both cases can be treated equally. Assigning $\mathbf{v_a}$, $\mathbf{v_b}$, $\mathbf{v_c}$ and $\mathbf{v_d}$ vertices as in Fig. 3.5, the convexity constraint of an edge can be expressed as a negative volume for the parallelepiped defined by the adjacent triangles:

$$((\mathbf{v_b} - \mathbf{v_a}) \times (\mathbf{v_c} - \mathbf{v_a})) \cdot (\mathbf{v_d} - \mathbf{v_a}) \leq 0. \tag{3.2}$$

In order to satisfy the convexity constraints, I have formulated an optimization program in which $\mathbf{v_3}$ is constrained to the segment between the position that minimizes surface deviation, $\hat{\mathbf{v}}_\mathbf{3}$, and the unconstrained filtered position, $\tilde{\mathbf{v}}_\mathbf{3}$. The objective function is the distance to $\tilde{\mathbf{v}}_\mathbf{3}$. This is a simple linear program in one dimension. The result position of the constrained filtered edge collapse can be written as a linear interpolation between the initial position and the goal position:

$$\begin{aligned} \mathbf{v_3} &= u \cdot \hat{\mathbf{v}}_\mathbf{3} + (1-u) \cdot \tilde{\mathbf{v}}_\mathbf{3}, \\ u &\geq 0, \\ u &\leq 1. \end{aligned} \tag{3.3}$$

The convexity constraints in Eq. 3.2 can be rewritten based on constants A and B as:

$$\begin{aligned} A \cdot u + B \geq 0, \quad \text{where} \\ A = ((\mathbf{v_d} - \mathbf{v_a}) \times (\mathbf{v_c} - \mathbf{v_a})) \cdot (\hat{\mathbf{v}}_\mathbf{3} - \tilde{\mathbf{v}}_\mathbf{3}), \\ B = ((\mathbf{v_d} - \mathbf{v_a}) \times (\mathbf{v_c} - \mathbf{v_a})) \cdot (\tilde{\mathbf{v}}_\mathbf{3} - \mathbf{v_a}). \end{aligned} \tag{3.4}$$

The resulting vertex $\mathbf{v_3}$ corresponds to the minimum value of u that meets all the constraints. When $\tilde{\mathbf{v}}_\mathbf{3}$ is not a feasible solution but a solution exists, the constrained filtered edge collapse can be regarded as a partial filter.

Global Convexity Constraints

The global convexity constraints are difficult to express explicitly in the optimization program, so they cannot be incorporated into the filtering process. Instead, they have to be verified after the filtering has been performed. I verify them by computing the convex hulls of the affected convex patches after the edge collapse and performing the required intersection tests, using OBBs [GLM96] and spatial partitioning.

If a position $\mathbf{v_3}$ that meets the local convexity constraints is found, I check the global constraints. If they are met, the edge collapse is valid. If they are not met, then I check the global constraints at $\hat{\mathbf{v}}_\mathbf{3}$. If they are not met at $\hat{\mathbf{v}}_\mathbf{3}$ either, the edge collapse is considered invalid and it is disabled. If $\hat{\mathbf{v}}_\mathbf{3}$ meets the global constraints, I perform a bisection search between $\hat{\mathbf{v}}_\mathbf{3}$ and $\mathbf{v_3}$ of up to K iterations (in the practical implementation $K = 3$), searching for the position closest to $\tilde{\mathbf{v}}_\mathbf{3}$ that meets the global convexity constraints, as shown in Fig. 3.6. The resulting vertex $\mathbf{v_3}$ is reassigned to this position.

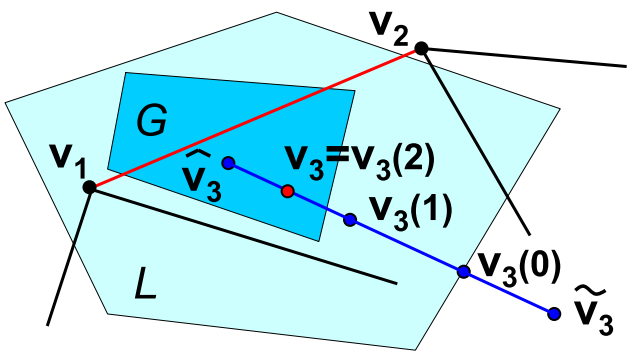

Figure 3.6: **Filtered Edge Collapse with Convexity Constraints.** *The figure shows a filtered edge collapse in which bisection search is required to find a position that meets the convexity constraints. G and L represent feasible regions of global and local constraints respectively.*

3.3.5 Parameters for Error Metrics

To perform multiresolution collision detection for haptic rendering, one must define error metrics that will dictate the selection of CLODs. The error metrics I have defined are described in Sec. 3.4, but here I enumerate the parameters that have to be computed after performing sensation preserving simplification and constructing the CLODs. Besides the resolution r of each LOD, I compute the following parameters:

1. The surface deviation, s, between every convex patch c and the original mesh M_0. This parameter is an upper bound on the size of the local geometric surface details lost during the simplification and filtering process.

2. A support area, D, for every vertex in the hierarchy. This value will later be used to estimate the contact area at run-time. For every vertex $\mathbf{v}$ of the initial mesh M_0, the support area D is computed as the projected area onto the tangent plane of $\mathbf{v}$ of the faces incident to $\mathbf{v}$, such that they are within a certain distance tolerance from $\mathbf{v}$ along the direction of the normal $\mathbf{n}$ of $\mathbf{v}$ and their normal lies inside a normal cone of $\mathbf{n}$. For this computation, I have typically used the same distance tolerance as the one used for contact queries (see Sec. 3.4). When an edge $(\mathbf{v_1}, \mathbf{v_2})$ is collapsed to a vertex $\mathbf{v_3}$, I assign to $\mathbf{v_3}$ the minimum of the two support areas of $\mathbf{v_1}$ and $\mathbf{v_2}$.

3. A Hausdorff distance, h, for every convex hull C. The value of h is computed as the maximum directed Hausdorff distance from the descendant convex hulls of C.

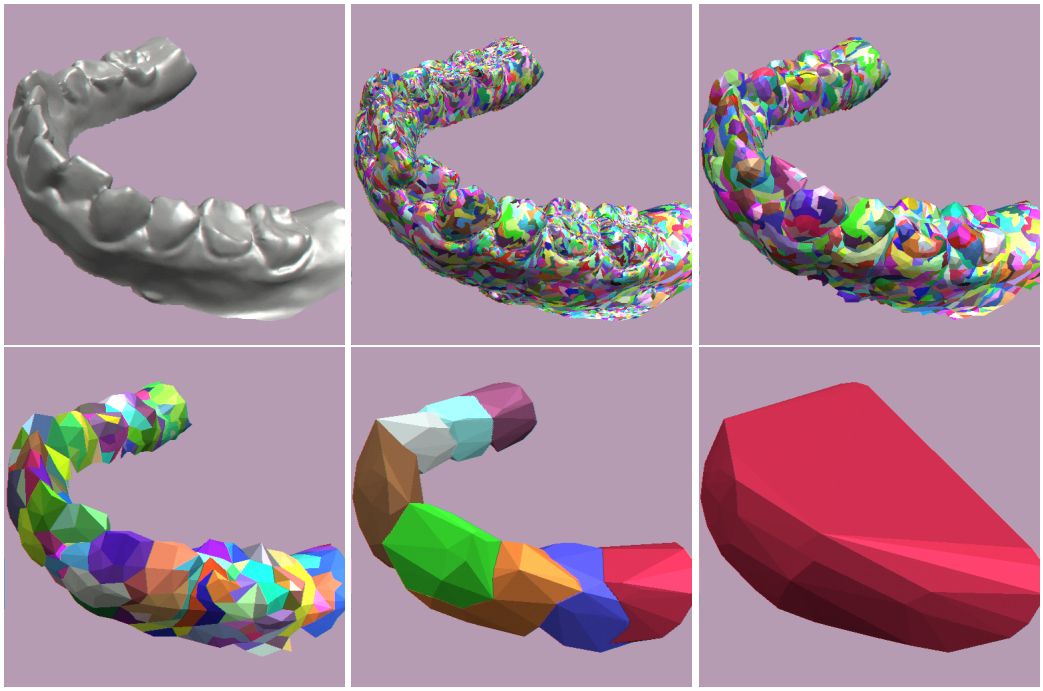

Figure 3.7: **CLODs of a Lower Jaw.** *From left to right and top to bottom, original mesh, M_0, and convex patches of M_0, M_3, M_6, M_{11}, and M_{14}.*

3.3.6 Examples of Contact Levels of Detail

Fig. 3.7 shows several of the LODs obtained when processing a model of a lower jaw (see Sec. 3.5 for statistics of this model). The LODs M_3 and M_6 shown in the figure are obtained from the original model by sensation preserving simplification. Along with the simplification process, the convex patches of the original model are successively merged in order to create the BVH. Thus, the multiresolution hierarchy itself serves as a BVH for collision detection.

Unlike in other types of BVHs, in CLODs the different levels of the BVH bound only their associated LODs; they do not necessarily bound the original surface, as may be deduced from the figures. As described later in Sec. 3.4.2, the inclusion of Hausdorff distances in the CLOD data structure will ensure conservative collision detection.

The free LODs M_{11} and M_{14} in the figure are obtained by pairwise merging of convex hulls. They serve to complete the BVH, but cannot be considered as LODs of a multiresolution hierarchy.

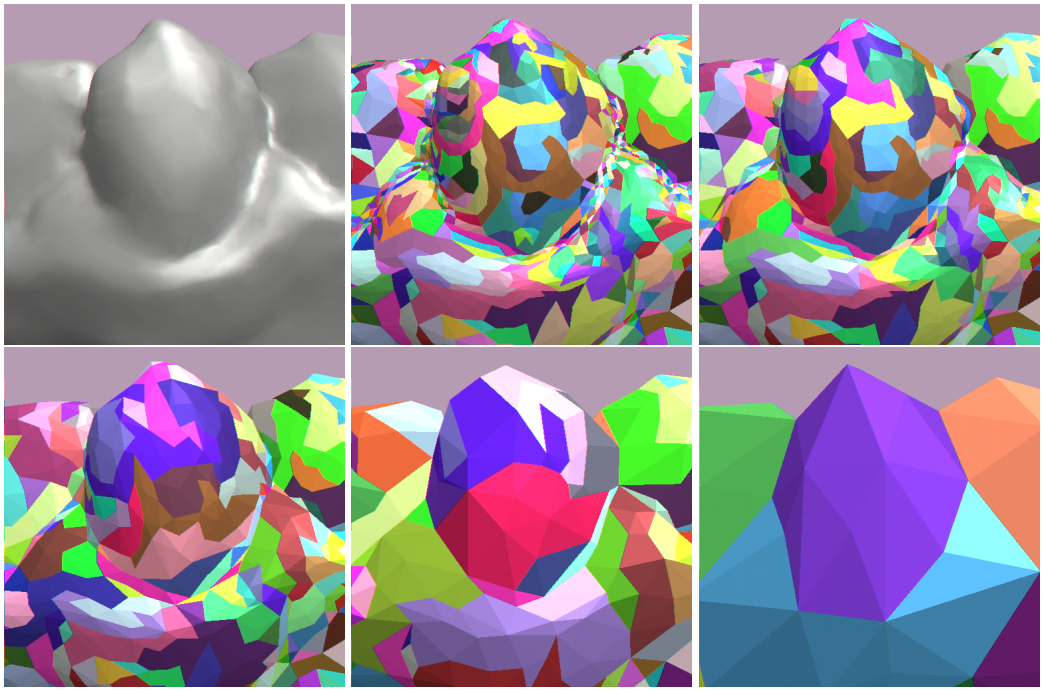

Figure 3.8: **Detail View of the CLODs of a Lower Jaw.** *From left to right and top to bottom, original mesh, M_0, and convex patches of M_0, M_1, M_2, M_4, and M_7.*

Fig. 3.8 shows a more detailed view of the simplification and merging results. Notice that, in the creation of M_1, most of the simplification and merging operations take place at the gums. The gums are, indeed, the locations with detail at the highest resolution. When the process reaches LOD M_7, one particular tooth is covered by a single convex patch, thus showing the success of the construction of the hierarchy.

3.4 Multiresolution Collision Detection

In the previous section I have described the creation of CLODs. Ultimately, CLODs are intended to be used at runtime collision detection. The scope of this dissertation is 6-DoF haptic rendering, but CLODs can also be used to accelerate contact queries in rigid body simulation.

In this section I describe a multiresolution collision detection algorithm based on CLODs, as well as novel selective refinement criteria. I first introduce the type of contact queries relevant to my 6-

DoF haptic rendering approach, and then I describe multiresolution collision detection using CLODs. I also present error metrics and define the selective refinement tests for both haptic rendering and rigid body simulation. To conclude this section, I discuss how contact information from CLODs can be used for collision response in haptic rendering and rigid body simulation.

3.4.1 Contact Query between Two Objects

The concept of collision detection covers various types of proximity queries between pairs of objects, such as intersection detection, exact distance queries, or approximate distance queries [EL01]. For example, an intersection query $Q(A,B,0)$ between two objects A and B is a boolean query that determines if A and B intersect.

For haptic rendering purposes, I define a *contact query* $Q(A,B,d)$. This query returns a set of contacts that sample the regions of A and B that are closer than a distance tolerance d. Each contact is described by a contact normal, a pair of contact points, and a distance value (separation distance or penetration depth, depending whether the objects are disjoint or penetrating in the region of contact). Specifically, I solve $Q(A,B,d)$ by performing a surface convex decomposition of A and B [EL01] and testing pairwise distances between convex BVs [GJK88, Lin93, Cam97, Mir98b, EL00]. I define the distance query $q(a,b,d)$ between two convex pieces $a \in A$ and $b \in B$ as a boolean query that returns whether a and b are closer than d. If A and B are disjoint the closest points between convex patches form a superset of the local minima of the distance between A and B. The local minimum distances have also been used by Johnson and Willemsen [JW03] for 6-DoF haptic rendering. If the objects penetrate the contact points define localized penetration depth values [KOLM03].

As mentioned in Sec. 3.1.2, the contact query can be accelerated by traversing BVHs in tandem, and it can be described by the BVTT. A node ab in the BVTT encapsulates a pair of BVs $a \in A$ and $b \in B$, which might be tested with a query $q(a,b,d)$. Performing a contact query $Q(A,B,d)$ can be understood as descending along the BVTT as long as the distance query q returns true.

3.4.2 Multiresolution Contact Query

Using CLODs, multiresolution collision detection can be implemented by slightly modifying the typical collision detection procedures based on BVHs. In multiresolution collision detection, the decision

of splitting a node *ab* of the BVTT is made as a combination of the distance query $q(a,b,d)$ and a selective refinement query. First, the distance query q is performed. If the query returns false, there is no need to descend to the children nodes. If the result of the distance query is true, the query for selective refinement is performed on *ab*. If the node *ab* must be refined, the traversal continues with the children of *ab* in the BVTT. Otherwise, contact information can directly be computed for *ab*.

Descending to children BVTT nodes involves descending to the children BVs, as occurs in any BVH, but it also involves refining the surface representation, due to the duality of CLODs. Selective refinement of nodes of the BVTT activates varying contact resolutions across the surfaces of the interacting objects, as shown in Fig. 3.1. In other words, every contact is treated independently and its resolution is selected in order to cull away negligible local surface detail.

The test for selective refinement can embed various perceptual error metrics and it determines if higher resolution is required to describe the contact information at each contact location. In Sec. 3.4.3, I describe the test for selective refinement in more detail, and suggest error metrics for 6-DoF haptic rendering and rigid body simulation.

Modification of the Distance Query

A collision detection algorithm based on BVHs must ensure that, if a leaf node *ab* of the BVTT returns true to the contact query, then all its ancestors must return true as well. This is usually achieved by ensuring that the union of the BVs at every level of a BVH fully contains the surface of the object. In CLODs this containment property may not hold, but the correctness of the collision detection can be ensured by modifying the collision distance d_{ab} between two BVs a and b. Given a distance tolerance d for a contact query $Q(A,B,d)$, the distance tolerance d_{ab} for a distance query $q(a,b,d_{ab})$ must be computed as:

$$d_{ab} = d + h(a) + h(b), \tag{3.5}$$

where $h(a)$ and $h(b)$ are maximum directed Hausdorff distances from the descendant BVs of a and b to a and b respectively. As explained in Sec. 3.3.5, these Hausdorff distances can be precomputed during the process of sensation preserving simplification.

Analysis of the Bounding Volume Test Tree

The BVTT may not be constructed at runtime, because a contact query will visit only a small fraction of its nodes. The BVTT, however, serves as a good tool to analyze the performance of a collision detection algorithm based on BVHs.

Using CLODs, when the distance query and the selective refinement test return true for a node *ab* of the BVTT, I split the BV whose children have coarser resolution. This splitting policy yields a BVTT in which the levels of the tree are sorted according to their resolution, as shown in Fig. 3.9. Nodes of the BVTT at coarser resolution are closer to the root. A resolution-based ordering of the BVTT is a key factor for maximizing the performance of runtime collision detection, because CLODs with lower resolution and larger error are stored closer to the root of the BVTT. Descending along the BVTT has the effect of refining the CLODs. The resolution-based ordering of CLODs of two different objects is possible because the definition of resolution presented in Sec. 3.3.2 is an object-independent absolute metric. If objects are scaled, the value of resolution must be scaled accordingly.

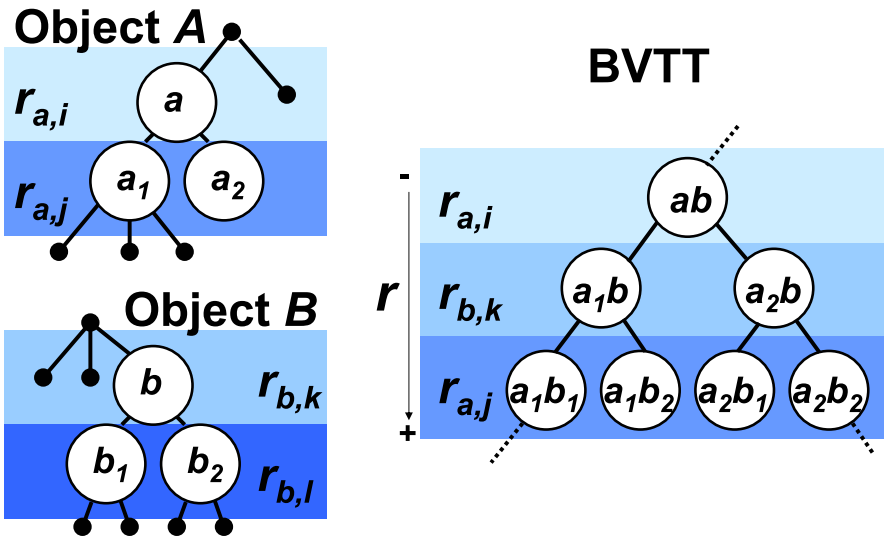

Figure 3.9: **Resolution-Based Ordering of the Bounding Volume Test Tree.** *A node splitting policy based on CLOD resolution implies a BVTT in which levels are sorted according to increasing resolution. Descending on the BVTT has the effect of increasing CLOD resolution.*

As pointed out in Sec. 3.3.3, the top levels of the BVHs are “free” LODs, obtained by unconstrained pairwise merging of convex patches. These top levels of the BVTT have no associated metric

of resolution and they always test positive for selective refinement. The boundary between free and regular LODs is indicated in Fig. 3.10 by the line λ.

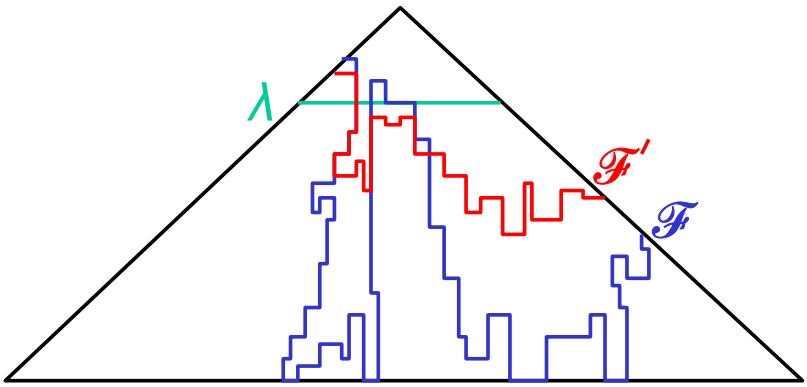

Figure 3.10: **Generalized Front Tracking of the BVTT.** *The front of the BVTT for an exact contact query, $\mathscr{F}$, is raised up to the new front $\mathscr{F}'$ using CLODs, since the recursive distance queries can stop at coarser resolutions. λ indicates the free CLODs, constructed by unconstrained merging of convex patches.*

As indicated in Sec. 3.1.2, temporal coherence can be exploited using GFT. One can store the front $\mathscr{F}$ of the BVTT where the result of the distance query q switches from true to false, as shown in Fig. 3.10. The front is recorded at the end of a contact query Q_i, and the next query Q_{i+1} proceeds by starting recursive distance queries q at every node in the front $\mathscr{F}$. The time spent by a contact query Q depends directly on the number of nodes visited in the BVTT. GFT considerably reduces the running time of Q when temporal coherence is high, which is the case in haptic rendering. Then, the time spent by Q is proportional to the size of the front $\mathscr{F}$. The cost, however, can still be $O(mn)$ in the worst case, where m and n are the numbers of convex patches of the input objects.

In a multiresolution collision detection setting, the addition of a selective refinement test further increases the performance of the query. In the BVTT, the new active front, $\mathscr{F}'$, is above the original front $\mathscr{F}$ that separates nodes that test positive to distance queries q from nodes that test negative. Using CLODs, the front does not need to reach the leaves of the BVTT, as long as the error is smaller than some tolerance, as depicted in Fig. 3.10. This approach results in a much faster processing of contact queries and, ultimately, it enables 6-DoF haptic rendering of complex objects.

3.4.3 Selective Refinement and Error Metrics

As discussed in Sec. 3.1.1, the perceptibility of surface features depends on the ratio between their size and the contact area. Following this observation, I have designed error metrics to be used in the selective refinement test.

Functions ϕ_a and ϕ_b evaluate the size of the features missed when a contact query stops at a node ab of the BVTT. ϕ_a (and similarly ϕ_b) is computed as:

$$\phi_a = \frac{s_a}{r_a^2}, \tag{3.6}$$

where s_a is the surface deviation from the convex patch bounded by a to the original surface, and r_a is the resolution of the current CLOD. Note that both values are precomputed. The function ϕ_a can be regarded as a measure of the volume of the fictitious features that are filtered out when using a as the CLOD.

The effect of contact area is taken into account by averaging the function ϕ over an estimated contact area D. Thus, I compute a weighted surface deviation s^* as:

$$s^*_{ab} = \frac{\max(\phi_a, \phi_b)}{D},$$

$$D = \max(D_a, D_b). \tag{3.7}$$

s^* can been regarded as surface deviation errors weighted by a constant that depends on both the contact area and the resolutions of local surface features.

The online computation of the contact area between a pair of convex patches is too expensive, given the runtime constraint of haptic rendering. Therefore, the contact area D is estimated by selecting the maximum support area of the contact primitives (i.e., vertex, edge, or triangle). As explained in Sec. 3.3.5, a support area D is stored for every vertex in the CLOD data structure. For edge or triangle contact primitives, I interpolate the support areas of the end vertices, using the barycentric coordinates of the contact point.

Once the weighted surface deviation s^* is computed, the selective refinement test will compare this value to an error threshold s_0. If s^*_{ab} is above the threshold, the node *ab* must be refined. Otherwise, the missing detail is considered to be imperceptible. The error threshold s_0 can be determined based upon different perceptual metrics. Next I present an error metric for haptic rendering, and error metrics for rigid body simulation inspired by the work of O'Sullivan and Dingliana [OD01]. Selective refinement using CLODs can be implemented combining any of these error metrics. I also indicate how CLODs can be used to perform time-critical collision detection [Hub94].

Haptic Error Metric

Ideally, s_0 should be a distance defined based on human perceptibility thresholds. Such metric is independent of object size and polygon count, and it may result in excessively large, intractable CLOD resolutions. Instead, I have decided to select s_0 as a metric relative to the size of the interacting objects, under the assumption that the range of motion of the haptic device covers approximately the space occupied by the objects in the virtual workspace. As a consequence, the required CLOD resolutions are independent of the scale of the objects, and the contact queries run in nearly constant time, as discussed later in Sec. 3.5.

Based on experiments described in Sec. 3.5.2, s_0 should be in the range of 2.5% to 5% of the radii of the interacting objects.

Velocity-Dependent Metric

Set s_0 as a value proportional to the relative velocity of the colliding objects at the contact location. This is based on the observation that the gap between the objects is less noticeable as the objects move faster [OD01].

View-Dependent Metric

Determine s_0 based on screen-space errors. Given N pixels of admissible error, a distance l from the camera to the contact location, a distance n to the near plane of the view frustum, a size f of the frustum in world coordinates, and a size i of the image plane in pixels,

$$s_0 = \frac{N \cdot l \cdot f}{n \cdot i}. \tag{3.8}$$

Constant Frame Rate

One important feature of CLODs is the fact that they can be used for time-critical collision detection. The error metrics, computed at every potential contact location, can be used to prioritize the refinement. To achieve a guaranteed frame rate for real-time applications, the collision detection algorithm will perform as many distance queries as possible, within a fixed time interval. The query event queue will be prioritized based on $\frac{s^*}{s_0}$.

3.4.4 Solving Discontinuities in Collision Response

A major issue in systems that use multiresolution representations is the discontinuity that arises when the algorithm switches between different LODs. This problem is known as "popping" in multiresolution (visual) rendering. In multiresolution collision detection, the way to tackle discontinuities depends on the type of collision response:

a) Application of penalty forces based on contact information.

b) Detection of collision events and computation of valid velocities (and accelerations).

Next, I present some interpolation techniques to resolve discontinuities in each of these cases.

Interpolation of Contact Information

The 6-DoF haptic rendering approach presented in this dissertation computes penalty contact forces at each simulation time step, based on the contact information returned by the contact query (i.e., separation distance, penetration depth, contact points, and contact normals). The effects of switching CLODs between time steps are discontinuities in the net contact force and torque, which are eventually perceived by the user.

The discontinuities are solved by interpolating contact information from different CLODs. When the sensation preserving selective refinement determines that the current resolution is accurate enough,

I perform a conservative refinement step and compute contact information for the children of the current node of the BVTT. The contact information is interpolated between the two levels.

Naturally, CLOD interpolation increases the number of nodes of the BVTT that are visited. For complex models and/or complex contact scenarios, however, CLODs still outperform exact collision detection, as presented in Sec. 3.5.

Interpolation of Collision Events

Some methods for rigid body simulation are based on time-stepping algorithms that search for the time instants when collision events occur. When a collision takes place, the numerical integration of object states is interrupted and new valid velocities (and accelerations) are computed.

Many dynamic factors determine the selection of CLODs in rigid body simulation, such as the velocity of the objects, the contact area, and the distance to the camera. Special treatment is necessary so that switching CLODs does not generate inconsistencies or deadlock situations in the time-stepping algorithm. Given a node ab_i of the BVTT, with negative distance query at times t_i and t_{i+1} of the simulation, and a node ab_{i+1}, child of ab_i, with positive distance query at both time instants, if the refinement test of ab_i switches from false to true at $t \in [t_i, t_{i+1}]$, the time stepping method will encounter an inconsistency. It will try to search for a nonexistent collision event in the interval $[t_i, t_{i+1}]$.

I solve this problem by estimating a collision time t_c, interpolating the separation distance of the node ab_i at t_i and the penetration depth of the node ab_{i+1} at t_{i+1}. Collision response can be applied at t_c with ab_i as the active CLOD, and the numerical integration continues.

3.5 Experiments and Results

In this section I describe experiments conducted to test and analyze CLODs. I first describe benchmark models used in the experiments and present statistics of the CLOD data structures for those models. Then I discuss the selection of tolerance values for multiresolution 6-DoF haptic rendering using CLODs, based on experimental analysis. Last, I present performance results on 6-DoF haptic rendering and rigid body simulation.

3.5.1 Benchmark Models

I have created CLOD representations for the models listed in Table 3.2. This table shows the original complexity of the models (Orig. Tris and Orig. BVs), the complexity of the coarsest CLOD obtained by sensation preserving simplification (Simp. Tris and Simp. BVs), the normalized resolution (for unit object radius) of the finest and coarsest CLODs, and the number of "free" and total CLODs. As described in Sec. 3.3.3, the BVHs are completed with free CLODs that cannot be used to report contact information.

Models	Orig. Tris	Orig. BVs	Simp. Tris	Simp. BVs	r_1	r_λ	Free CLODs	Total CLODs
Lower Jaw	40,180	11,323	386	64	144.5	12.23	6	15
Upper Jaw	47,339	14,240	1,038	222	117.5	19.21	8	15
Ball Joint	137,060	41,913	122	8	169.9	6.75	3	17
Golf Club	104,888	27,586	1,468	256	157.6	8.31	8	16
Golf Ball	177,876	67,704	826	64	216.3	7.16	6	18
Cup	64,000	15,490	1,532	241	71.0	7.70	7	14
Spoon	86,016	16,125	1,074	61	230.1	13.57	5	14

Table 3.2: **Benchmark Models for CLODs and Associated Statistics.** *The numbers of triangles (Orig. Tris) and the numbers of convex patches (Orig. BVs) of the initial meshes of the models; the numbers of triangles (Simp. Tris) and the numbers of convex patches (Simp. BVs) of the coarsest CLODs obtained by sensation preserving simplification; resolution (r_1 and r_λ) of the finest and coarsest CLODs; and free CLODs and total number of CLODs.*

Note that the models are simplified to coarsest CLODs with 122 to $1,532$ triangles. The number of BVs in the coarsest CLODs ranges from an extreme case of 8 BVs, for the ball joint model, to 256 BVs. As a result, the sensation preserving selective refinement can be applied at early stages in the contact query, and this allows more aggressive culling of parts of the BVTT whenever the perceptible error is small. The visual complexity and surface detail of the benchmark models is reflected in Figs. 3.12, 3.13, 3.14, and 3.17.

3.5.2 Experiments on Perceptible Contact Information

The performance of CLODs in haptic rendering is heavily determined by the selection of the threshold of weighted surface deviation s_0. If the chosen value is too high, the perceived contact information will deviate too much from the exact contact information. On the other hand, if the value is too low

and the selected CLODs are moderately complex (i.e., consisting of more than a thousand convex patches), the contact query will no longer be executable at the required rate. This severely degrades the realism of haptic perception.

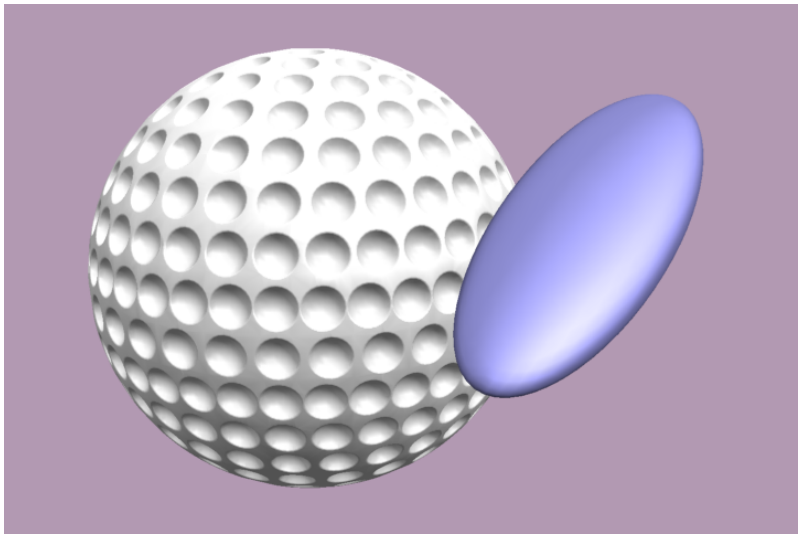

Figure 3.11: **Exploration of a Multiresolution Golf Ball with an Ellipsoid.** *Scenario of the experiments for identifying haptic error tolerances with CLODs.*

I have conducted an informal experiment to test the validity of CLODs for haptic rendering, and also to identify what are the error tolerances for which the missing surface detail is not perceptible to users of the system. The scenario of the experiment consists of a golf ball (please refer to Table. 3.2 for statistics of the model) that is explored with an ellipsoid, as shown in Fig. 3.11. The ellipsoid consists of $2,000$ triangles, and it is fully convex. The ellipsoid has varying curvature, implying a wide range of contact scenarios, and the selective refinement will stop at varying CLODs.

For simplicity, I created a CLOD representation of the golf ball only, and left the ellipsoid invariant. Thus, the fidelity of the contact forces relies only on the adequacy of the resolution of the golf ball that is selected at each contact. 12 users were asked to identify the value of the threshold s_0 of the haptic error metric at which the perception of surface detail of the golf ball started deviating. The values of s_0 were in the range from 0.05% to 20% of the radius of the ball.

Table 3.3 indicates how many subjects picked each threshold value. Based on the results of the experiments, the value of s_0 for haptic simulations should be in the range of 2.5% to 5% of the radii of

s_0	$\geq 10\%$	5%	2.5%	1%	$\leq 0.5\%$
no. users	0	4	7	1	0

Table 3.3: **Experiments on Error Metrics.** *A majority of subjects reported a threshold of* 2.5% *to* 5% *of the radius of the golf ball for the haptic error metric.*

the models. The users also reported that the main characteristic they explored was the perceptibility of the dimples of the golf ball.

3.5.3 Performance Experiments in 6-DoF Haptic Rendering

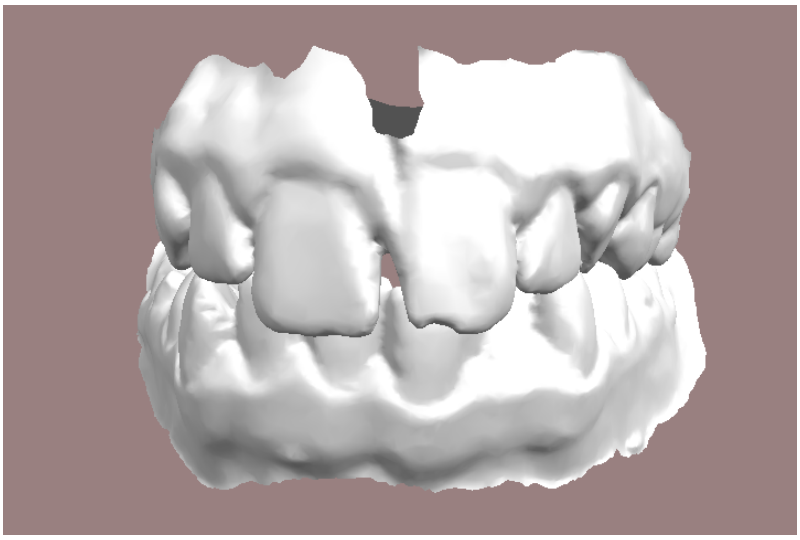

Figure 3.12: **Upper and Lower Jaws.** *A benchmark scenario for 6-DoF haptic rendering using CLODs.*

I have successfully applied CLODs to 6-DoF haptic rendering on the following benchmark scenarios:

- Moving upper and lower jaws (See Fig. 3.12).
- Interlocking ball joints (See Fig. 3.13).
- Golf club tapping a golf ball (See Fig. 3.14).

Statistics of the CLOD data structures of the models have been given in Table. 3.2.

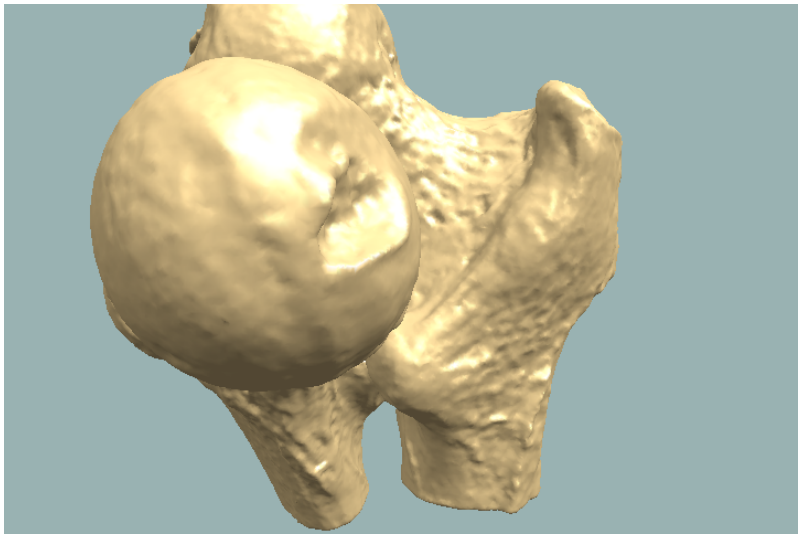

Figure 3.13: **Ball Joints.** *A benchmark scenario for 6-DoF haptic rendering using CLODs.*

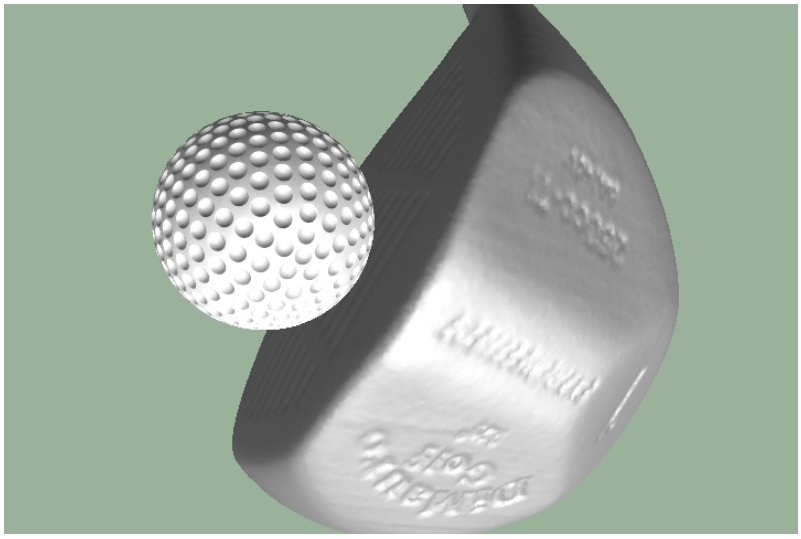

Figure 3.14: **Golf Club and Ball.** *A benchmark scenario for 6-DoF haptic rendering using CLODs.*

I have analyzed contact forces and running time on the benchmarks of the moving jaws and the golf club and ball. In particular, I have compared force profiles and statistics of the contact query between interactive haptic simulations and more accurate offline simulations. The interactive haptic simulations were executed using CLODs and error tolerances of $s_0 < 5\%$ of the radii of the models. The motions of the upper jaw and the golf club were controlled using a haptic device, which also displayed the contact forces to the user. The trajectories were recorded in the interactive simulations, and played back to perform more accurate simulations offline. The full accuracy corresponds to

offline simulations in which the contact queries were computed using the publicly available libraries SWIFT++ [EL01] and DEEP [KLM02]. In the graphs shown later, I refer to these simulations as *exact*. In the *exact* and low-error simulations, collision detection runs at update rates of tens of Hz, which are too low for interactive haptic rendering of stiff contacts. Next, I describe implementation details and the performance results.

Implementation Details

The haptic demonstrations have been performed using a 6-DoF *Phantom*^{TM} haptic device, a dual Pentium-4 2.4GHz processor PC with 2.0 GB of memory and Windows2000 OS. The implementation, both for preprocessing and for the haptic rendering, has been developed using C++. The implementation of multiresolution collision detection based on CLODs uses distance and penetration depth queries between convex patches from the publicly available libraries SWIFT++ [EL01] and DEEP [KLM02].

To validate CLODs in haptic rendering, the results of the contact queries must be used to compute collision response and output force and torque in haptic simulations. I employed the direct haptic rendering pipeline described in [KOLM03]. In this rendering pipeline, contacts computed in the contact query are clustered, and then a penalty force proportional to penetration depth is computed for each cluster. The net penalty force is output directly to the user, without a stabilizing intermediate representation. In this way, the experiments do not get distorted by the use of intermediate representations, and the analysis can focus on the fidelity of the contact forces. Later in Chapter 5, I will present a more stable haptic rendering pipeline that uses contact information from CLODs.

Following the approach developed with Kim et al. [KOLM03], in the experiments I applied penalty forces if the interacting objects came closer than a contact tolerance d. I chose the value of d so that the maximum force of the haptic device was exerted for a zero contact distance with the optimal value of stiffness.

Performance Results and Analysis

Fig. 3.15 shows the contact profile, including the force profile, the query time, and the size of the front of the BVTT, for 200 frames of the moving jaws simulation. The profiles of contact forces are similar

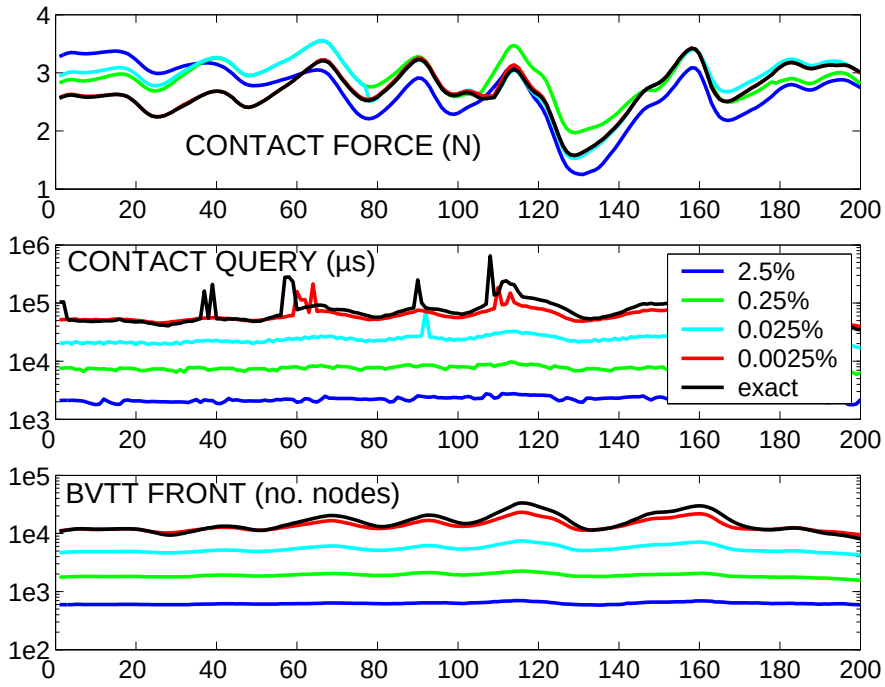

Figure 3.15: **Contact Profile for Moving Jaws.** *Top: The profiles of the contact forces displayed using CLODs, with varying error tolerances up to* 2.5*% of the radii of the jaws, all show very similar patterns. This similarity implies that the sensations of shape provided to the user are nearly identical. Middle: A log plot of contact query time using CLODs with various error tolerances shows up to two orders of performance improvement. Bottom: The number of nodes in the front of the BVTT is also reduced by more than a factor of 10.*

for all error tolerances up to 2.5% of the radii of the jaws. There are some deviations on the average force, but the patterns are similar. With different error tolerances, and using penalty-based rendering methods, the perception of shape properties is almost invariant, only the perceived surface location varies in a noticeable way. I reach this conclusion because the second derivatives of the force profiles are almost identical in all cases, and shape properties such as curvature depend on second derivatives of the surface.

The time spent by the contact queries goes down from more than 100ms using *exact* contact queries, to slightly more than 2ms with CLODs and an error tolerance of 2.5% of the radii of the jaws. This drastic decrease of the query times enables interactive 6-DoF haptic rendering.

Fig. 3.16 shows the contact profile for 300 frames of simulation of the golf scene. In the benchmark of the golf club and ball there is a speed-up of nearly two orders of magnitude in the query time

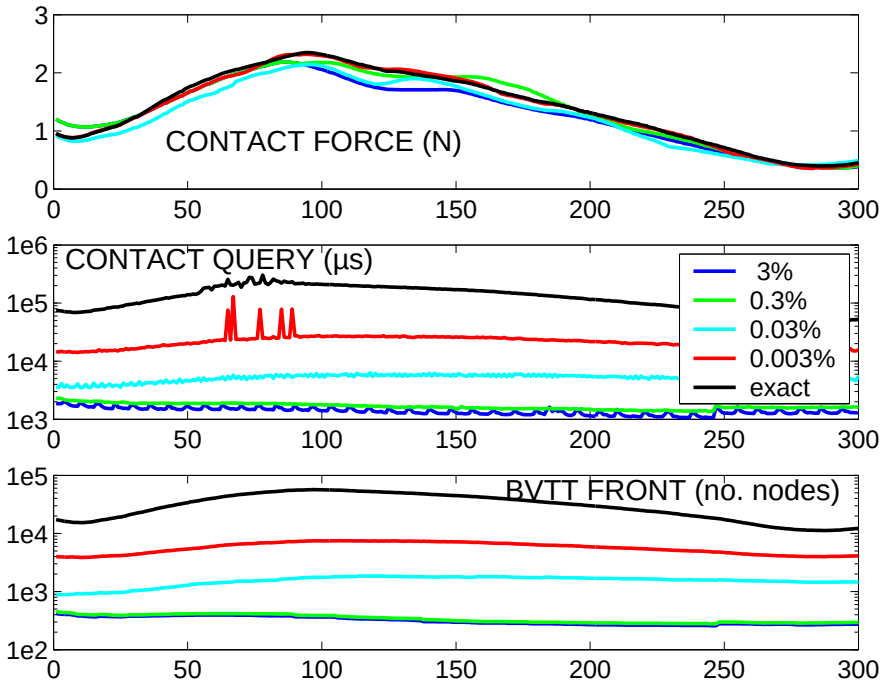

Figure 3.16: **Contact Profile for Golf Scene.** *Top: The profiles of the contact forces displayed using CLODs, with varying error tolerances up to* 3*% of the radius of the ball, show nearly identical patterns. Middle: A log plot of contact query time using CLODs with various error tolerances shows more than two orders of performance improvement. Bottom: The number of nodes in the front of the BVTT is reduced by nearly a factor of 100.*

between interactive haptic rendering using CLODs and *exact* offline simulation. Notice that the query time is roughly proportional to the number of nodes in the BVTT front.

The size of the BVTT front varies monotonically with the contact force. Due to the use of penalty methods, the force is higher when the club and the ball are closer. That explains the increase in the size of the BVTT front, because larger areas of the objects are in close proximity. As reflected in the graphs, however, the size of the BVTT front (and therefore the query time) is more susceptible to lack of coherence when the error tolerance is lower. As a result, CLODs with acceptable error tolerances provide almost constant-time contact queries. Please note that the spikes in the contact query time present in Fig. 3.15 and Fig. 3.16 are due to context switching in the CPU.

Comparison with Related Work

As mentioned in Sec. 3.1.2, the running time of any collision detection algorithm depends on both the input and output size of the problem. Regarding previous approaches to 6-DoF haptic rendering, the systems presented by Gregory et al. [GME+00] and Kim et al. [KOLM03] employ exact collision detection methods [EL01], and they are limited to relatively simple models or modestly complex contact scenarios, and do not scale well to highly complex object-object interaction. The discretized approximation presented by McNeely et al. [MPT99] can avoid direct dependency on the input size of the problem by limiting the number of points sampled and the number of voxels generated. This approach, however, is limited by the fact that the objects are sampled at a constant resolution, selected a priori, not based on the contact configuration.

In contrast, CLODs, by reducing the combinatorial complexity of the input based on the contact configuration at each local neighborhood of (potential) collision, automatically decrease the output size as well. In addition, the selection of CLODs is contact-dependent in order to minimize the perceived shape difference while maximizing the amount of simplification and performance gain possible. Multiresolution collision detection based on CLODs is perhaps the first "contact-dependent simplification" algorithm.

From the analysis of the contact profiles, I draw two main conclusions regarding the validity of CLODs for 6-DoF haptic rendering:

- The contact information obtained with error tolerances derived from perceptual experiments provides shape cues that are nearly identical to those provided by exact collision detection methods. This resemblance supports the observation that perception of features depends on the ratio between their size and the contact area.

- With the same error tolerances, the running time of the contact queries is almost 2 orders of magnitude faster than the running time of exact collision detection methods. For the complex scenarios presented in the benchmarks, my multiresolution approach enables force update rates suitable for interactive haptic rendering.

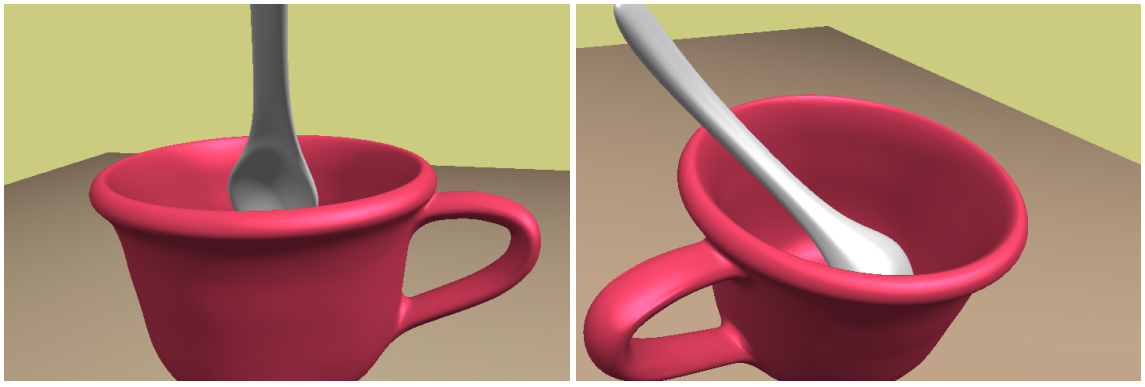

Figure 3.17: **Spoon in a Cup.** *Snapshots of benchmark scenario for rigid body simulation using CLODs.*

3.5.4 Performance Experiments in Rigid Body Simulation

I have tested the application of CLODs on a rigid body simulation of a spoon sliding inside a cup (See Fig. 3.17). Statistics of the CLOD data structures of the models have been given in Table. 3.2. I have compared average contact query times using CLODs (with different thresholds for the haptic error metric) and using the *exact* collision detection library SWIFT++ [EL01].

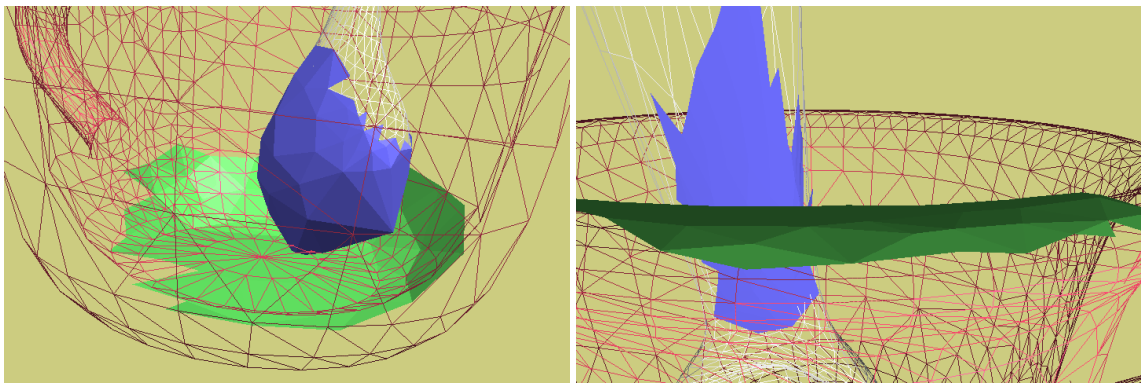

Figure 3.18: **Application of CLODs with Velocity-Dependent Metric.** *In blue and green, the vicinity of the contact locations shown at the resolution of the adaptively selected CLODs. Right: coarse resolution is selected when the spoon falls quickly inside the cup; Left: finer resolution is selected when the spoon slides slowly along the side of the cup.*

I have also evaluated CLODs with velocity- and view-dependent error metrics. In Fig. 3.18 coarse CLODs are selected when the spoon falls on the bottom of the cup, and fine resolution CLODs (up to 4 levels finer at some places) are selected when the spoon slides along the side of the cup. In the first case the polygon counts of the representations are roughly 10 times larger than in the second case.

Next, I describe implementation details as well as the performance results with the haptic error metric.

Implementation Details

The rigid body simulation of the spoon falling inside the cup has been implemented using impulse-based methods [MC95, Mir96]. Collision response detects collision events and applies impulses that ensure a valid kinematic state after the collision. Once the spoon collides several times with the cup, its trajectory may deviate considerably using different error thresholds. Therefore, the query time has been compared when the spoon hits the cup for the first time.

Unlike in the experiments of 6-DoF haptic rendering described in the previous section, the distance tolerance for collision detection d can be set to visually imperceptible values (in the order of $1/1000$ of the radii of the objects).

The simulation was performed on a Pentium-4 2.4GHz processor PC with 2.0GB of memory and Windows2000 OS.

Performance Results and Analysis

The timing profile of Fig. 3.19 shows average query times and the number of nodes in the BVTT front for 35 frames of the simulation of the spoon sliding inside the cup. Each simulation frame corresponds to 30ms of simulation time, and the query time is averaged over the many contact queries that may take place during each frame.

As the timings show, CLODs with $s_0 = 3.5\%$ perform at least as good as the exact algorithm for most of the simulation duration. During several time intervals, the performance gain is almost one order of magnitude.

As a result of the conservative modification to the distance queries introduced in Sec. 3.4.2, the multiresolution collision detection algorithm using CLODs cannot prune as efficiently as the exact algorithm [EL01] when the objects are separated by a distance notably larger than d. When they come close to contact, however, CLODs outperform the exact queries.

Notice that the size of the BVTT front is almost constant with CLODs, but it varies considerably with exact collision detection. This behavior implies that CLODs benefit more from temporal coherence, and it also explains the spikes present in the timings for the exact algorithm. These spikes take

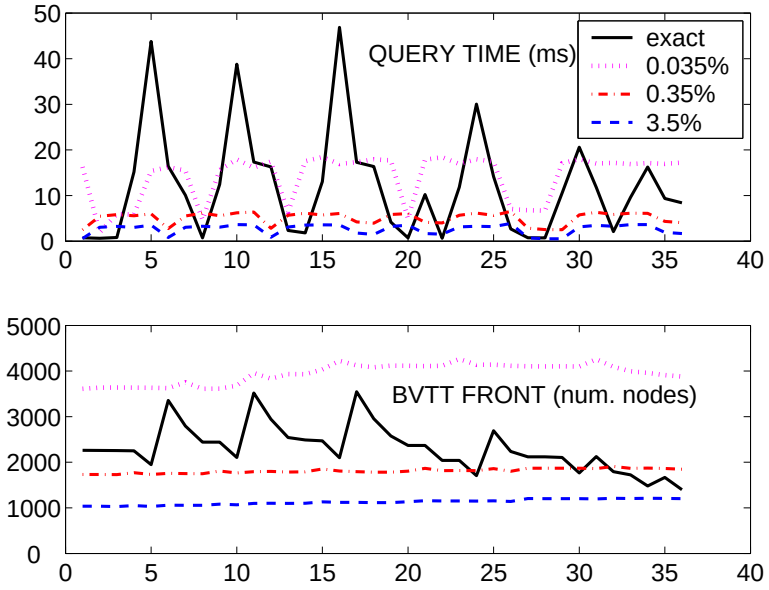

Figure 3.19: **Query Profile for Cup and Spoon Scene.** *Top: The average contact query with CLODs and an error threshold of* 3.5% *outperforms the exact contact query by one order of magnitude in some intervals. Bottom: the front of the BVTT exhibits less coherence with exact collision detection, producing spikes in the contact query time.*

place at the instants when the objects are about to interpenetrate and the BVTT front evolves rapidly.

3.6 Summary and Limitations

In this chapter I have presented *contact levels of detail* (CLODs), a multiresolution collision detection algorithm. The purpose of CLODs is to accelerate collision detection in 6-DoF haptic rendering, by selecting the appropriate object resolution at each contact location. As proved in the experiments, CLODs are also applicable to rigid body simulation.

The cost of collision detection is higher when large areas of the objects are in close proximity. However, perceptual observations indicate that the perceptibility of surface features decreases if the contact area is large. This relationship sets the basis for the use of multiresolution collision detection in 6-DoF haptic rendering.

In this chapter I have presented a generic data structure for multiresolution collision detection, integrating multiresolution object representations with BVHs in one single dual hierarchy. I denote each level of this hierarchy as a CLOD. Moreover, I have described a particular implementation of the

data structure using convex hulls as the BVs. The CLODs of convex hulls are constructed following a *sensation preserving simplification* process that combines atomic surface decimation and filtering operations with merging of convex patches. I have defined an atomic operation denoted as *filtered edge collapse* that filters high-resolution geometric detail subject to convexity constraints.

CLODs are used at runtime to perform contact queries between pairs of objects. Multiresolution collision detection is achieved by combining distance queries between BVs with a selective refinement test. I have designed different error metrics for selective refinement: haptic metric, velocity-dependent metric, and view-dependent metric.

I have performed experiments to test the performance of CLODs in haptic rendering and rigid body simulation. In 6-DoF haptic rendering, CLODs enable interactive display of complex contact scenarios at force update rates higher than 300Hz with little degradation of the contact forces. This implies a performance speed-up of up to two orders of magnitude compared to exact collision detection methods.

Next I discuss the type of situations that CLODs are best suited for, as well as related limitations.

3.6.1 Adequacy of CLODs

CLODs are best suited in the following situations:

- Large-area contacts between complex models.
- 6-DoF haptic rendering or rigid body simulation methods where the collision response is based on penalty methods.

In both cases, the reason for the adequacy of CLODs is that large areas of the objects are in parallel close proximity [GLM96]. These are, in fact, some of the most challenging contact scenarios.

If the collision response acts by detecting collision events, or if contacts occur at small contact areas, the front of the BVTT is inherently small with exact collision detection, so CLODs will not provide such important performance gain. In these cases, especially if the application is rigid body simulation, CLODs may be more effective using velocity- or view-dependent error metrics.

3.6.2 Limitations Related to the Construction of CLODs

From the perspective of constructing a multiresolution representation, sensation preserving simplification can be compared with both mesh decimation techniques and mesh filtering techniques. These techniques may offer better results than sensation preserving simplification in certain aspects.

- **Surface deviation.** LODs created by filtered edge collapse operations will have larger surface deviation than LODs of traditional mesh decimation [Hop96, GH97, LT98]. This deviation inevitably results from combining decimation and filtering. In sensation preserving simplification, detail at high resolution is filtered independently of its magnitude, while mesh decimation techniques will preserve detail to minimize surface deviation. The elimination of detail benefits the creation of the BVH and does not detract from the output quality of haptic rendering, since the filtered detail is quantified and taken into account in runtime collision detection. Multiresolution representations obtained through mesh decimation techniques are not able by themselves to support efficient contact queries.

- **Visual smoothness.** Representations obtained through filtering [Tau95, GSS99] appear smoother than those obtained by sensation preserving simplification. The decrease in visual smoothness in CLODs is due to the use of fewer samples (i.e., vertices) to represent meshes with the same frequency content. This approach is advantageous, because the ultimate goal is to accelerate collision detection.

In the creation of CLODs using sensation preserving simplification there are also issues that influence the applicability and efficiency of multiresolution collision detection, and these are worth exploring too.

- **Lack of containment.** As introduced in Sec. 3.2.2, in CLODs, a level of the multiresolution representation may not bound the original surface. This has two drawbacks: (1) it requires a modification of the contact queries, as explained in Sec. 3.4.2, and (2) it implies that multiresolution collision detection will not be conservative, in the sense that contact points at coarse resolution may be inside the full-resolution objects. Progressive hulls [SGG$^+$00], as mentioned in Sec. 3.3.3, can be used to enforce containment of fine CLODs in coarse CLODs, but they

do not ensure containment of individual patches in the convex hulls of their parents. This requirement can only be fulfilled by adding offsets to the BVs, but that would imply using BVs other than convex hulls, with accompanying problems for obtaining contact information. In applications where object interpenetration is forbidden, currently CLODs have to be used in conjunction with large collision tolerances. For such applications, it would be interesting to implement CLODs with a procedure other than sensation preserving simplification, enforcing containment of the original object in successive CLODs.

- **Existence of free CLODs.** As mentioned in Sec. 3.3.3, the BVH may contain some levels that cannot be used to report multiresolution contact information, because they cannot be considered as low-resolution representations of the input object. The existence of free CLODs reduces the applicability of multiresolution collision detection, because the aggressiveness of the run-time culling will be limited. The culling efficiency will be maximized if the topological and geometric constraints can be removed during the creation of the CLODs.

- **Static LODs.** A surface patch may undergo several atomic simplification operations between two consecutive CLODs, which introduce discontinuities in the multiresolution representation. In Sec. 3.4.4, I suggest interpolation techniques for avoiding discontinuities in collision response induced by the use of static LODs, but another possibility would be to design an implementation of CLODs with dynamic LODs.

Recently, Yoon et al. [YSLM04] have proposed a data structure similar to CLODs using OBBs as the BVs. Yoon's data structure is based on a cluster hierarchy of progressive meshes [Hop96], with the additional advantage of dynamic LODs. Yoon's implementation relaxes the geometric constraints in the construction of the CLODs, but loses many of the benefits of convex hulls for obtaining contact information (see Sec. 3.3.1).

3.6.3 Inherent Limitations of Multiresolution Collision Detection

In situations of sliding, rolling and/or twisting contact between textured surfaces, the observation that perceptibility of features decreases with larger contact area does not hold. Small but highly correlated features may provide important haptic cues that are erroneously filtered away using CLODs (or any

other multiresolution collision detection algorithm based on local refinement). This type of situation is problematic for all collision detection methods, because of the high sampling density (i.e., object resolution) required, and it is the focus of Chapter 4 of this dissertation.

Chapter 4

Haptic Texture Rendering

Rendering of surface texture (i.e., fine geometric features on an object's surface) is an important topic in haptics that has received increasing attention. The intrinsic surface property of texture is among the most salient haptic characteristics of objects. It can be a compelling cue to object identity and it can strongly influence forces during manipulation [KL02]. In medical applications with limited visual feedback, such as minimally-invasive or endoscopic surgery [Sal99], and virtual prototyping applications of mechanical assembly and maintainability assessment [WM03], accurate haptic feedback of surface detail is a key factor for successful dexterous operations.

Most of the existing haptic rendering algorithms have focused on force rendering of rigid or deformable untextured models. In 6-DoF haptic rendering of rigid bodies, collision detection has a dominant computational cost. The performance of collision detection algorithms depends on the size of the input models, which in turn depends on the sampling density of the models, both for polygonal representations [RK00, KOLM03, JW03] and for voxel-based representations [MPT99, WM03]. To be correctly represented, surfaces with high-frequency geometric texture detail require higher sampling densities, thereby increasing the cost of collision detection. Effective physically based force models have been proposed to render the interaction between the tip (a point) of a haptic probe and a textured object [Min95, HBS99]. However, no technique is known to display interaction forces and torques between two textured models. In fact, computation of texture-induced forces using full-resolution geometric representations of the objects and handling contacts at microgeometric scale is computationally prohibitive, and new representations must be considered.

In Chapter 3, I presented *contact levels of detail* (CLODs), a multiresolution collision detection

algorithm especially designed for 6-DoF haptic rendering that minimizes the computational impact of collision detection and selects the appropriate object resolution at each contact location. CLODs, however, filter out high-resolution geometric features, thus ignoring texture effects arising in sliding, rolling, and twisting motion. This chapter of the dissertation addresses the computation of forces and torques due to the interaction *between two textured objects*.

Similar to graphical texture rendering [Cat74], objects with high combinatorial complexity (i.e., with a high polygon count) can be described by coarse representations and texture images that store fine geometric detail. I refer to these texture images as *haptic textures*. In this chapter I introduce a new approach to 6-DoF haptic rendering that enables the display of intricate interaction due to fine surface details, using simplified object representations and haptic textures. Contact information is first computed at coarse resolutions, using CLODs, and then refined accounting for the geometric detail captured in haptic textures.

A central part of the novel 6-DoF haptic rendering approach is a force model that captures texture effects. Recently Klatzky and Lederman (see [KL02] for a summary of their work) have presented several important findings on perception of roughness through an intermediate object. In this chapter I present the synthesis and analysis of a perceptually inspired force model for haptic texture rendering. Force and torque are computed based on the gradient of the directional penetration depth between two textured models. I also introduce an algorithm for approximating directional penetration depth between textured objects using haptic textures and a parallel implementation on programmable graphics hardware that enables interactive haptic display of forces and torques between complex textured models.

I have successfully tested and demonstrated the 6-DoF haptic texture rendering algorithm and implementation on several complex textured models. One example, consisting of a textured hammer interacting with a rough CAD part, is shown in Fig. 4.1. Subjects that experienced that example were able to perceive roughness induced by surface texture of the objects.

I have analyzed the influence of the perceptual factors identified by psychophysics studies on the vibratory motion induced by the force model. The experiments demonstrate a qualitative match between roughness perception in earlier experimental observations and the forces simulated using my model. I have also evaluated the effectiveness of the rendering algorithm for conveying roughness

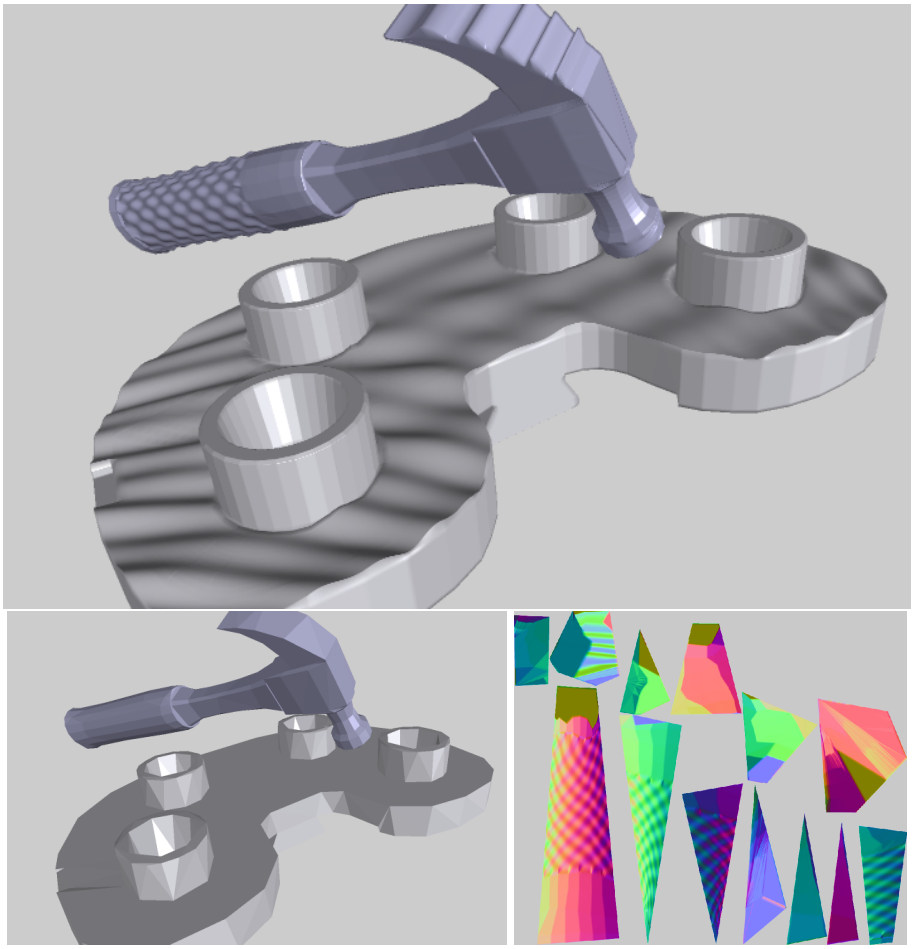

Figure 4.1: **Haptic Rendering of Interaction between Textured Models.** *Top: high-resolution textured hammer (433K polygons) and CAD part (658K polygons); Bottom left: low-resolution models (518 & 720 polygons); Bottom right: hammer texture with fine geometric detail.*

sensations during both translational and rotational motion. Finally, I have tested the performance of the algorithm and its implementation on complex benchmarks, obtaining force update rates higher than 100Hz.

This chapter compiles work and results previously published in [OJSL04] and [OL04]. The rest of the chapter is organized as follows. In Sec. 4.1, I define the notation used throughout the chapter and key terminology related to the concept of penetration depth. Sec. 4.2 presents the foundations of the rendering algorithm and the force model, which is described in Sec. 4.3. Sec. 4.4 introduces a simple yet effective algorithm for approximating directional penetration depth and its parallel implementa-

tion on graphics processors. Then I describe experiments and results in Sec. 4.5, and in Sec. 4.6, I summarize the chapter and conclude with a discussion on limitations of this work.

4.1 Definitions and Terminology

In this section I introduce notations used throughout the chapter and present definitions related to penetration depth, which is an essential element of the force model for haptic texture rendering.

4.1.1 Notations

A *height field H* is defined as a set $H = \{(x,y,z) \in \mathbb{R}^3 \mid z = h(x,y)\}$. I call $h : \mathbb{R}^2 \rightarrow \mathbb{R}$ a *height function*.

Let $\mathbf{p}$ denote a point in $\mathbb{R}^3$, let $\mathbf{p}_{xyz} = (p_x\ p_y\ p_z)^T$ denote the coordinates of $\mathbf{p}$ in a global reference system, and $\mathbf{p}_{uvn} = (p_u\ p_v\ p_n)^T$ its coordinates in a rotated reference system $\{\mathbf{u},\mathbf{v},\mathbf{n}\}$. A surface patch $S \subset \mathbb{R}^3$ can be represented as a height field along a direction $\mathbf{n}$, if $p_n = h(p_u,p_v), \forall \mathbf{p} \in S$. Then, one can define a mapping $g : D \rightarrow S, D \subset \mathbb{R}^2$, as $g(p_u,p_v) = \mathbf{p}_{xyz}$, where:

$$\mathbf{p}_{xyz} = g(p_u,p_v) = (\mathbf{u}\ \ \mathbf{v}\ \ \mathbf{n})\,(p_u\ \ p_v\ \ h(p_u,p_v))^T. \tag{4.1}$$

The inverse of the mapping g is the orthographic projection of S onto the plane $(\mathbf{u},\mathbf{v})$ along the direction $\mathbf{n}$. Given the mapping g, the height function h can be computed as:

$$h(p_u,p_v) = \mathbf{n} \cdot g(p_u,p_v). \tag{4.2}$$

4.1.2 Definitions of Penetration Depth

Penetration depth δ between two intersecting polyhedra A and B is typically defined as the minimum translational distance required for separating them (see Fig. 4.2-b). As mentioned in Sec. 2.3.2, this distance is equivalent to the distance from the origin to the Minkowski sum of A and $-B$. *Directional penetration depth* $\delta_{\mathbf{n}}$ along the direction $\mathbf{n}$ is defined as the minimum translation along $\mathbf{n}$ to separate the polyhedra (see Fig. 4.2-c). The penetration depth between two intersecting surface patches will be referred to as *local penetration depth*.

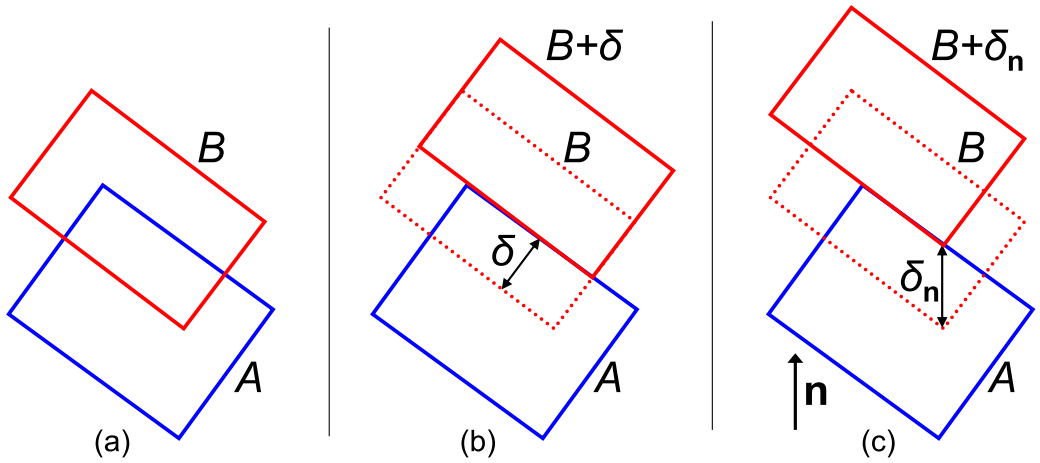

Figure 4.2: **Definitions of Penetration Depth.** *(a) Intersecting objects A and B, (b) global penetration depth δ, and (c) directional penetration depth δ_n along* **n**.

Let us assume that two intersecting surface patches S_A and S_B can be represented as height fields along a direction **n**. Consequently, S_A and S_B can be parameterized by orthographic projection along **n**, as expressed in Sec. 4.1.1. The parameterization yields mappings $g_A : D_A \rightarrow S_A$ and $g_B : D_B \rightarrow S_B$, as well as height functions $h_A : D_A \rightarrow \mathbb{R}$ and $h_B : D_B \rightarrow \mathbb{R}$. The directional penetration depth $\delta_{\mathbf{n}}$ of the surface patches S_A and S_B is the maximum height difference along the direction **n**, as illustrated in Fig. 4.3 by a 2D example. Therefore, the directional penetration depth $\delta_{\mathbf{n}}$ can be defined as:

$$\delta_{\mathbf{n}} = \max_{(u,v)\in(D_A\cap D_B)} (h_A(u,v) - h_B(u,v)) . \tag{4.3}$$

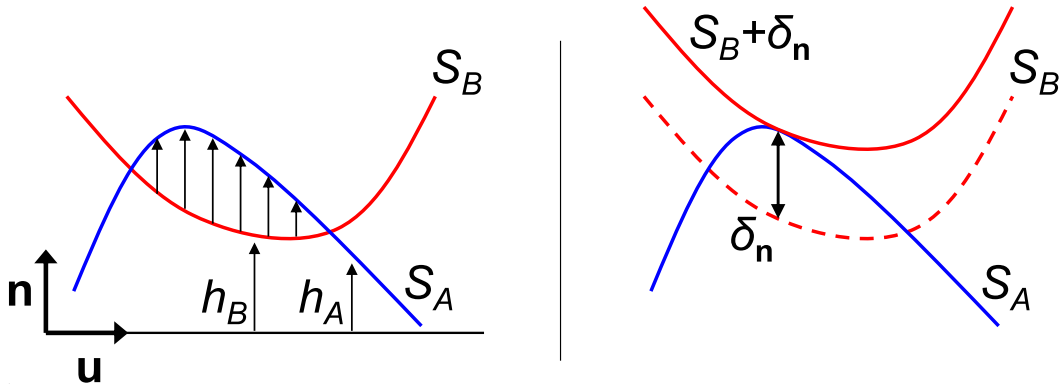

Figure 4.3: **Penetration Depth of Height Fields.** *Directional penetration depth of surface patches expressed as height difference.*

4.2 Foundations of a 6-DoF Haptic Texture Rendering Algorithm

In this section I present the foundations of a force model for 6-DoF haptic texture rendering and a rendering algorithm in which objects are represented by coarse geometric approximations and haptic textures. In Sec. 2.1.2, I have summarized the results of psychophysics studies on perception of roughness that guide the design of the force model. In this section I extend the conclusions from those studies to more general settings and I introduce the rendering pipeline based on haptic textures.

4.2.1 Offset Surfaces and Penetration Depth

Klatzky et al. [KLH+03] stated that the perception of roughness is intimately related to the trajectory traced by the probe. In particular, they identified the value of texture spacing at which the probe can exactly fall between two texture dots as *drop point*. The peak of roughness perception occurs approximately at the drop point, and it depends on geometric (i.e., probe diameter) and dynamic factors (i.e., speed).

For a spherical probe, and in the absence of dynamic effects, the surface traced by the probe during exploration constitutes an offset surface, as shown in Fig. 4.4. The oscillation of the offset surface produces the vibratory motion that encodes roughness. The idea of offset surfaces has also been used by Okamura and Cutkosky [OC01] to model interaction between robotic fingers and textured surfaces.

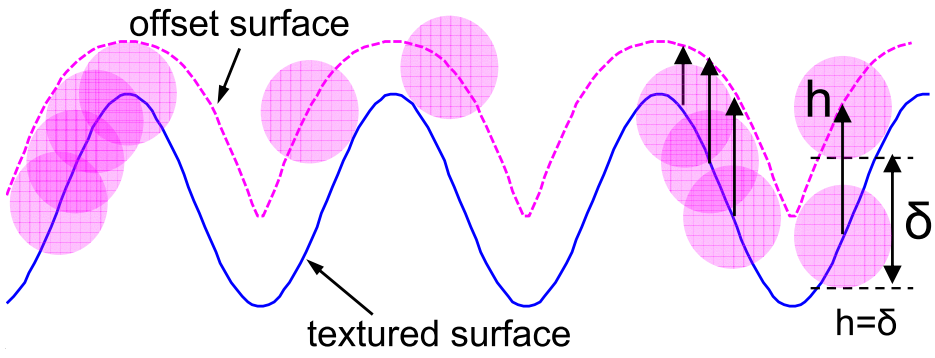

Figure 4.4: **Offset Surfaces.** *Left: offset surface computed as the convolution of a surface with a sphere; Center: sphere whose trajectory traces an offset surface; Right: correspondence between vertical penetration depth (δ) and height of the offset surface (h).*

In the design of a force model for haptic texture rendering, one faces the question: How can the concept of offset surface be generalized to the interaction between two arbitrary surfaces? To

answer this question, let us consider the case of a spherical probe whose center moves along a textured surface, as depicted in Fig. 4.4. In this situation, the probe penetrates the textured surface. The *vertical penetration depth* δ is the vertical translation required to separate the probe from the textured surface, and it is the same as the height of the offset surface *h*. Unlike the height of offset surfaces, (directional) penetration depth is a metric that can be generalized to the interaction between arbitrary surfaces.

The relationship between offset surfaces and penetration depth can also be explained through the concept of Minkowski sums. An offset surface corresponds to the boundary of the Minkowski sum of a given surface and a sphere. Therefore, the height of the offset surface at a particular point is the distance to the boundary of the Minkowski sum for a particular position of the probe, which is the same as the penetration depth. Actually, the height of the offset surface is the distance to the surface along a particular direction (i.e., vertical), so the distance to the boundary of the Minkowski sum must also be measured along a particular direction. This distance is known to be the *directional penetration depth*.

Since, for spherical probes, perception of roughness is tightly coupled with the undulation of the traced offset surface, in the force model for general surfaces I take into account the variation of penetration depth (i.e., its gradient). The validity of the gradient of a height field as a descriptor for texture-induced forces has been shown for 3-DoF rendering methods [Min95, HBS99]. The use of the gradient of penetration depth in 6-DoF haptic rendering can be considered as a generalization of the concept used in 3-DoF haptic rendering.

4.2.2 Haptic Rendering Pipeline Using Haptic Textures

Perception of shape and perception of texture have been classified as two psychologically different tactile cues [KL03]. From a geometric perspective, some authors have also created distinct categories of geometric information, based on the scale of the data: shape (or form), features (or waviness) and texture (or roughness) [Cos00, Whi94]. 3-DoF haptic texture rendering methods have also demonstrated that the separation of shape and texture can yield successful results from the computational perspective.

Therefore, I have also opted to design a 6-DoF haptic texture rendering algorithm in which geometric models are composed of simplified representations along with texture images storing fine

geometric detail. In the context of haptic rendering, I denote these texture images as *haptic textures*. Contact information between two objects represented by simplified representations and haptic textures will be computed in two main steps:

1. Obtain approximate contact information from simplified geometric representations.

    1.1 Perform collision detection between the low-resolution meshes.

    1.2 Identify each pair of intersecting surface patches as *one contact*.

    1.3 Characterize each contact by a pair of contact points on the patches and a penetration direction **n**.

2. Refine this contact information using detailed geometric information stored in haptic textures.

    2.1 For each contact, compute approximate directional penetration depth along **n**, using haptic textures.

    2.2 Compute force and torque, using a novel force model for texture rendering.

The 6-DoF haptic texture rendering algorithm presented in this chapter of the dissertation deals with the computation of force and torque to be applied to the virtual object governed through a haptic device (i.e., the *probe object*). The display of stable and responsive force and torque to the user is treated later in Chapter 5.

Integration with Contact Levels of Detail

The 6-DoF haptic texture rendering algorithm can be applied to two different types of objects:

- Objects whose models are given as low-resolution representations along with haptic textures.
- Objects described by high-resolution models that can be represented by contact levels of detail (CLODs).

Both types of objects are treated in a uniform way. The input models must be parameterized and, in case of using CLODs, the parameterization must be consistent across all levels of the hierarchy,

and distortion must be minimized. I have integrated parameterization procedures based on existing techniques [SSGH01, COM98] in the *sensation preserving simplification* process for creating CLODs.

For the models represented by CLODs, the low-resolution contact information will be obtained following the multiresolution collision detection algorithm described in Chapter 3.

4.3 Force Model

In this section I describe a force model for 6-DoF haptic texture rendering. First I describe some design considerations. Then, I detail the force and torque equations based on the gradient of directional penetration depth, and I discuss the solution of the gradient using finite differences.

4.3.1 Design Considerations

In 6-DoF haptic rendering, the forces transmitted to the user are a result of the collision response applied between the probe object and the rest of the virtual objects. Ideally, one would apply no force when the objects are disjoint, and compute impulses and/or constraint-based analytical forces when the objects collide or touch, thus preventing interpenetration. However, this approach is very time-consuming, because it requires detecting collision events, usually implemented by performing iterative contact queries every simulation frame. Instead, I adopt the penalty-based method, which computes contact forces proportional to penetration depth, thus reducing the cost of dynamic simulation.

A second consideration for the synthesis of the force model is that it need not account for certain dynamic effects. The influence of exploratory speed highlighted in perceptual studies is mainly determined by the motion and impedance characteristics of the subject. Haptic simulation is a human-in-the-loop system, therefore dynamic effects associated with grasping factors need not be modeled explicitly. Nevertheless, I have analyzed the dynamic behavior of the force model, observing that vibratory motion produced by simulated forces behaves in a way similar to physical roughness perception. The experiments are described in detail in Sec. 4.5.1.

The third consideration is that the effects of probe geometry and normal force identified by the perceptual studies must be accounted for directly in the force model. Geometric factors are addressed by computing force and torque proportional to the gradient of penetration depth. The influence of

normal force is captured by making tangential forces and torques proportional to the normal force. Note that perception of roughness grows monotonically with normal force, and this relation is captured qualitatively by the force model.

4.3.2 Penalty-Based Texture Force

For two objects A and B in contact, a penalty-based force is a force proportional to the penetration depth δ between them. Penalty-based forces are conservative, and they define an elastic potential field. I have extended this principle to compute texture-induced forces between two objects.

I define an elastic penetration energy U with stiffness k as:

$$U = \frac{1}{2} k \delta^2. \tag{4.4}$$

Based on this energy, force **F** and torque **T** are defined as:

$$\begin{pmatrix} \mathbf{F} \\ \mathbf{T} \end{pmatrix} = -\nabla U = -k\delta\left(\nabla\delta\right), \tag{4.5}$$

where $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}, \frac{\partial}{\partial \theta_x}, \frac{\partial}{\partial \theta_y}, \frac{\partial}{\partial \theta_z}\right)$ is the gradient in 6-DoF configuration space.

As described in Sec. 4.2.2, each contact between objects A and B can be described by a pair of contact points $\mathbf{p_A}$ and $\mathbf{p_B}$, and by a penetration direction **n**. I assume that, locally, the penetration depth between objects A and B can be approximated by the directional penetration depth $\delta_{\mathbf{n}}$ along **n**. Then, I rewrite Eq. 4.5 for $\delta_{\mathbf{n}}$ in a reference system $\{\mathbf{u}, \mathbf{v}, \mathbf{n}\}$ located at the center of mass of A. The axes **u** and **v** may be selected arbitrarily as long as they form an orthonormal basis with **n**. Eq. 4.5 reduces to:

$$\begin{pmatrix} F_u & F_v & F_n & T_u & T_v & T_n \end{pmatrix}^T = -k\delta_{\mathbf{n}} \begin{pmatrix} \frac{\partial \delta_{\mathbf{n}}}{\partial u} & \frac{\partial \delta_{\mathbf{n}}}{\partial v} & 1 & \frac{\partial \delta_{\mathbf{n}}}{\partial \theta_u} & \frac{\partial \delta_{\mathbf{n}}}{\partial \theta_v} & \frac{\partial \delta_{\mathbf{n}}}{\partial \theta_n} \end{pmatrix}^T, \tag{4.6}$$

where θ_u, θ_v and θ_n are the rotation angles around the axes **u**, **v** and **n** respectively.

The force and torque on object A (and similarly on object B) for each contact can be expressed in the global reference system as:

$$\mathbf{F_A} = (\mathbf{u}\ \mathbf{v}\ \mathbf{n})\,(F_u\ F_v\ F_n)^T\,,$$
$$\mathbf{T_A} = (\mathbf{u}\ \mathbf{v}\ \mathbf{n})\,(T_u\ T_v\ T_n)^T\,. \tag{4.7}$$

Forces and torques of all contacts are summed up to compute the net force and torque.

Generalizing Minsky's approach [Min95], the tangential forces F_u and F_v are proportional to the gradient of penetration depth. However, I also define a penalty-based normal force and gradient-dependent torque that describe full 3D object-object interaction. In addition, the tangential force and the torque are proportional to the normal force, which is consistent with the results of psychophysics studies, showing that perceived roughness increases with the magnitude of the normal force [KL02].

4.3.3 Penetration Depth and Gradient

Penetration depth functions δ and $\delta_{\mathbf{n}}$ are sampled at discrete points on a 6-DoF configuration space. I have opted for central differencing over one-sided differencing to approximate $\nabla\delta_{\mathbf{n}}$, because it offers better interpolation properties and higher order approximation. The partial derivatives are computed as:

$$\frac{\partial\delta_{\mathbf{n}}}{\partial u} = \frac{\delta_{\mathbf{n}}(u+\Delta u, v, n, \theta_u, \theta_v, \theta_n) - \delta_{\mathbf{n}}(u-\Delta u, v, n, \theta_u, \theta_v, \theta_n)}{2\Delta u}, \tag{4.8}$$

and similarly for $\frac{\partial\delta_{\mathbf{n}}}{\partial v}$, $\frac{\partial\delta_{\mathbf{n}}}{\partial\theta_u}$, $\frac{\partial\delta_{\mathbf{n}}}{\partial\theta_v}$ and $\frac{\partial\delta_{\mathbf{n}}}{\partial\theta_n}$.

$\delta_{\mathbf{n}}(u+\Delta u,\ldots)$ can be obtained by translating object A a distance Δu along the $\mathbf{u}$ axis and computing the directional penetration depth. A similar procedure is followed for other penetration depth values.

4.4 Haptic Textures for Approximation of Penetration Depth

In this section I present an algorithm for approximating local directional penetration depth for textured models using haptic textures and I describe a parallel implementation on graphics hardware.

4.4.1 Directional Penetration Depth

A contact between objects A and B is defined by two intersecting surface patches S_A and S_B. The surface patch S_A is approximated by a low-resolution surface patch $\hat{S}_A$ (and similarly for S_B). Let us define $f_A : \hat{S}_A \rightarrow S_A$, a mapping function from the low-resolution surface patch $\hat{S}_A$ to the surface patch S_A.

As expressed in Sec. 4.2.2, collision detection between two low-resolution surfaces patches $\hat{S}_A$ and $\hat{S}_B$ returns a penetration direction $\mathbf{n}$. Let us assume that both S_A and $\hat{S}_A$ (and similarly for S_B and $\hat{S}_B$) can be represented as height fields along $\mathbf{n}$, following the definition in Sec. 4.1.1. Given a rotated reference system $\{\mathbf{u},\mathbf{v},\mathbf{n}\}$, S_A and $\hat{S}_A$ are projected orthographically along $\mathbf{n}$ onto the plane $(\mathbf{u},\mathbf{v})$. This projection yields mappings $g_A : D_A \rightarrow S_A$ and $\hat{g}_A : \hat{D}_A \rightarrow \hat{S}_A$. I define $\bar{D}_A = D_A \cap \hat{D}_A$.

The mapping function g_A can be approximated by a composite mapping function $f_A \circ \hat{g}_A : \bar{D}_A \rightarrow S_A$ (See Fig. 4.5). From Eq. 4.2, I define an approximate height function $\hat{h}_A : \bar{D}_A \rightarrow \mathbb{R}$ as:

$$\hat{h}_A(u,v) = \mathbf{n} \cdot (f_A \circ \hat{g}_A(u,v)). \tag{4.9}$$

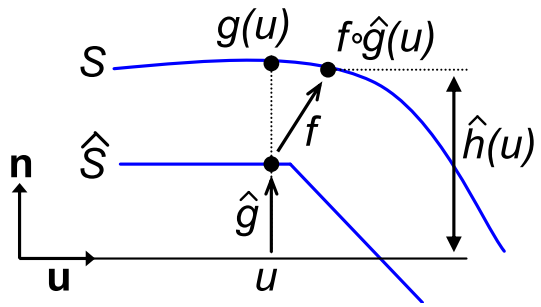

Figure 4.5: **Approximate Height Function.** *Height function of a surface patch approximated by a composite mapping function.*

Given approximate height functions $\hat{h}_A$ and $\hat{h}_B$, a domain $D = \bar{D}_A \cap \bar{D}_B$, and Eq. 4.3, the directional penetration depth $\delta_{\mathbf{n}}$ of S_A and S_B can be approximated by:

$$\hat{\delta}_{\mathbf{n}} = \max_{(u,v)\in D} \left(\hat{h}_A(u,v) - \hat{h}_B(u,v)\right). \tag{4.10}$$

Even though the computation of $\hat{\delta}_{\mathbf{n}}$ can be realized on CPUs, it is best suited for implementation on graphics processors (GPUs), as I will present next.

4.4.2 Computation on Graphics Hardware

As shown in Eq. 4.6, computation of 3D texture-induced force and torque according to the novel texture force model requires the computation of directional penetration depth $\delta_{\mathbf{n}}$ and its gradient at every contact. From Eq. 4.8, this requirement reduces to computing $\delta_{\mathbf{n}}$ all together at 11 configurations of object A [1]. As pointed out in section 2.3.2, computation of penetration depth using exact object-space or configuration-space algorithms is too expensive for haptic rendering applications. Instead, the approximation $\hat{\delta}_{\mathbf{n}}$ according to Eqs. 4.9 and 4.10 leads to a natural and efficient image-based implementation on programmable graphics hardware. The mappings $\hat{g}$ and f correspond, respectively, to orthographic projection and texture mapping operations, which are best suited for parallel and grid-based computation using GPUs.

For every contact, I first compute $\hat{h}_B$, and then perform two operations for each of the 11 object configurations: (1) compute $\hat{h}_A$ for the transformed object A, and (2) find the penetration depth $\hat{\delta}_{\mathbf{n}} = \max(\Delta\hat{h}) = \max\left(\hat{h}_A - \hat{h}_B\right)$ [2].

Height Computation

In the GPU-based implementation, the mapping $f : \hat{S} \rightarrow S$ is implemented as a texture map that stores geometric detail of the high-resolution surface patch S. I refer to f as a *haptic texture*. The mapping $\hat{g}$ is implemented by rendering $\hat{S}$ using an orthographic projection along $\mathbf{n}$. The height function $\hat{h}$ is computed in a fragment program. Points in S are obtained by looking up the haptic texture f and projecting the position onto $\mathbf{n}$. The result is stored in a floating point texture t.

I choose geometric texture mapping over other methods for approximating h (e.g., rendering S directly or performing displacement mapping) in order to maximize performance. The input haptic texture f is stored as a floating point texture.

[1] Due to the use of central differencing to compute partial derivatives of $\delta_{\mathbf{n}}$, object A must be transformed to two different configurations, where $\delta_{\mathbf{n}}$ is recomputed. All together the force model requires the computation of $\delta_{\mathbf{n}}$ itself and 5 partial derivatives, hence 11 configurations.

[2] I denote the height difference at the actual object configuration by $\Delta\hat{h}(0)$, and the height differences at the transformed configurations by $\Delta\hat{h}(\pm\Delta u)$, $\Delta\hat{h}(\pm\Delta v)$, $\Delta\hat{h}(\pm\Delta\theta_u)$, $\Delta\hat{h}(\pm\Delta\theta_v)$ and $\Delta\hat{h}(\pm\Delta\theta_n)$.

Max Search

The *max* function in Eq. 4.10 could be implemented as a combination of frame buffer read-back and CPU-based search. Expensive read-backs, however, can be avoided by posing the *max* function as a binary search on the GPU [GLW+04]. Given two height functions $\hat{h}_A$ and $\hat{h}_B$ stored in textures t_1 and t_2, I compute their difference and store it in the depth buffer. Then I scale and offset the height difference to fit in the depth range. Height subtraction and copy to depth buffer are performed in a fragment program, by rendering a quad that covers the entire buffer. For a depth buffer with N bits of precision, the search domain is the integer interval $[0, 2^N)$. The binary search starts by querying if there is any value larger than 2^{N-1}. I render a quad at depth 2^{N-1} and perform an occlusion query [3], which will report if any pixel passed the depth test, i.e., the stored depth was larger than 2^{N-1}. Based on the result, the depth of a new quad is set, and the binary search continues.

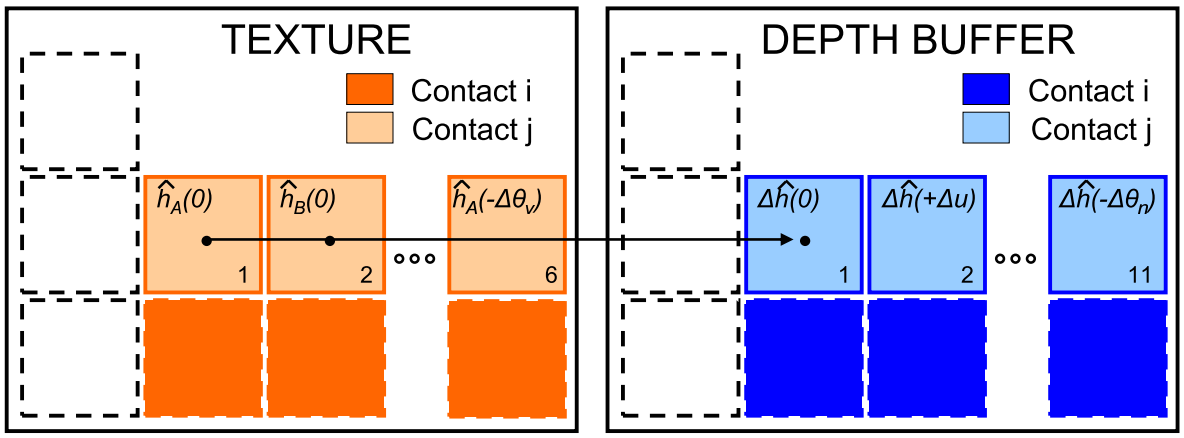

Figure 4.6: **Tiling in the GPU.** *Tiling of multiple height functions and contacts to minimize context switches between target buffers.*

Gradient Computation

The height functions $\hat{h}_A(\pm\Delta u)$, $\hat{h}_A(\pm\Delta v)$ and $\hat{h}_A(\pm\Delta\theta_n)$ may be obtained by simply translating or rotating $\hat{h}_A(0)$. As a result, only 6 height functions $\hat{h}_A(0)$, $\hat{h}_B(0)$, $\hat{h}_A(\pm\Delta\theta_u)$ and $\hat{h}_A(\pm\Delta\theta_v)$ need to be computed for each pair of contact patches. These 6 height functions are tiled in one single texture t to minimize context switches and increase performance (See Fig. 4.6). Moreover, the domain of each

[3] http://www.nvidia.com/dev_content/nvopenglspecs/GL_NV_occlusion_query.txt

height function is split into 4 quarters, each of which is mapped to one of the RGBA channels. This optimization exploits vector computation capabilities of fragment processors. As shown in Fig. 4.6, I also tile 11 height differences per contact in the depth buffer.

Multiple Simultaneous Contacts

The computational cost of haptic texture rendering increases linearly with the number of contacts between the interacting objects. However, performance can be further optimized. In order to limit context switches, I tile the height functions associated with multiple pairs of contact patches in one single texture t, and I also tile the height differences in the depth buffer, as shown in Fig. 4.6. The cost of *max search* operations is further minimized by performing occlusion queries on all contacts in parallel.

4.5 Experiments and Results

In order to analyze the force model and rendering algorithm for 6-DoF haptic texture rendering, I have performed two types of experiments. First, I describe offline experiments that analyze the influence of the factors highlighted by perceptual studies on the vibratory motion induced by the force model. Then, I present interactive experiments that test the effectiveness of the force model and the performance of its implementation.

4.5.1 Comparison with Perceptual Studies

As mentioned in Sec. 2.1.2, Klatzky and Lederman conducted experiments where users explored textured plates with spherical probes, and they reported subjective values of perceived roughness. I have created simulated replicas of the physical setups of Klatzky and Lederman's experiments in order to analyze the vibratory motion induced by the force model. The virtual experiments required the simulation of probe-plate interaction as well as human dynamics.

Description of Offline Experiments

The spherical probe is modeled as a circular disk of diameter D and the textured plate as a sinusoidal curve, as shown in Fig. 4.7. The circular disk moves along a horizontal line, which represents a low-resolution approximation of the sinusoidal curve. At each position of the disk I compute the vertical penetration depth $\delta_{\mathbf{n}}$ with respect to the sinusoidal curve.

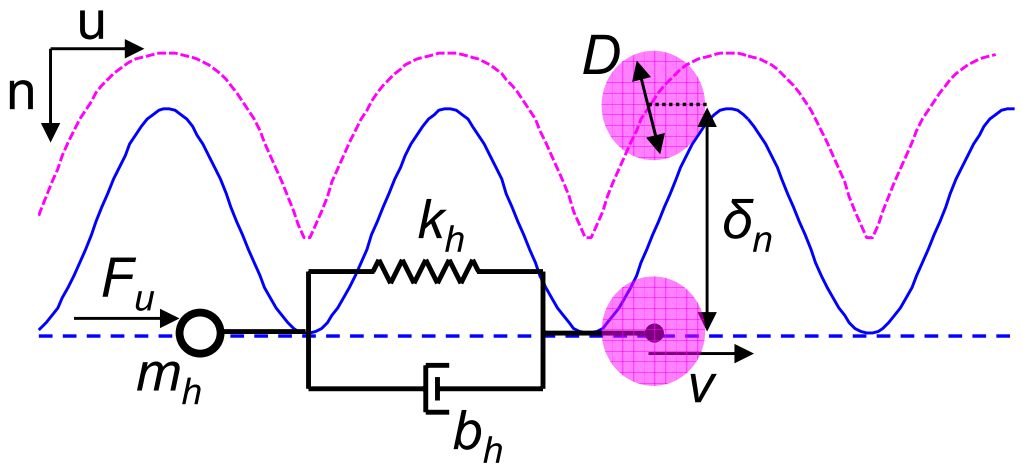

Figure 4.7: **Model of Probe-Surface Interaction and Grasping Dynamics.** *A disk moves on a sinusoidal texture at constant speed v while dragging a mass m_h. A texture force F_u, based on penetration depth $\delta_{\mathbf{n}}$, is applied to the mass.*

Following the force model for haptic texture rendering, texture-induced normal and tangential forces are defined as:

$$F_n = -k\delta_{\mathbf{n}}, \tag{4.11}$$

$$F_u = -k\delta_{\mathbf{n}}\frac{d\delta_{\mathbf{n}}}{du}. \tag{4.12}$$

The normal force F_n is one of the factors studied by Lederman et al. [LKHG00]. I will consider it as an input in our experiments. Then, I rewrite:

$$F_u = F_n\frac{d\delta_{\mathbf{n}}}{du}. \tag{4.13}$$

I model human dynamics as a system composed of mass m_h, spring k_h, and damper b_h [HC02].

The mass is linked through the spring and damper to a point moving at constant speed v on the textured surface. The dragging force imposed by the point accounts for the influence of exploration speed, which is a factor analyzed by Lederman et al. [LKHR99]. Figure 4.7 shows a diagram of the simulated dynamic system.

The texture force F_u also acts on the mass that models the human hand. In the presence of a textured surface, F_u will be an oscillatory force that will induce a vibratory motion on the mass. The motion of the mass is described by the following differential equation:

$$m_h \frac{d^2u}{dt^2} = k_h (vt - u) + b_h \left(v - \frac{du}{dt} \right) - F_u. \tag{4.14}$$

The experiments summarized by Klatzky and Lederman [KL02] reflect graphs of perceived roughness vs. texture spacing, both in logarithmic scale. I have simulated the motion of the hand model in Matlab, based on Eq. 4.14. Subjective roughness values cannot be estimated in the simulations. Instead, knowing that roughness is perceived through vibration, I have quantified the vibration during simulated interactions by measuring maximum tangential acceleration values. More specifically, I have measured $max(\frac{d^2u}{dt^2})$ once the motion of the mass reaches a periodic state.

Effects of Probe Diameter

In Fig. 4.8, I compare the effect of probe diameter on perceived roughness and on maximum simulated acceleration. The first conclusion is that the graph of acceleration vs. texture spacing can be well approximated by a quadratic function in a logarithmic scale. The second conclusion is that the peaks of acceleration and roughness functions behave in the same way as a result of varying probe diameter: both peaks of roughness and acceleration are higher and occur at smaller texture spacing values for smaller diameters.

Effects of Applied Force

The graphs in Fig. 4.9 compare the effect of applied force on perceived roughness and on simulated acceleration. In both cases the magnitude under study grows monotonically with applied force, and the location of the peak is almost insensitive to the amount of force.

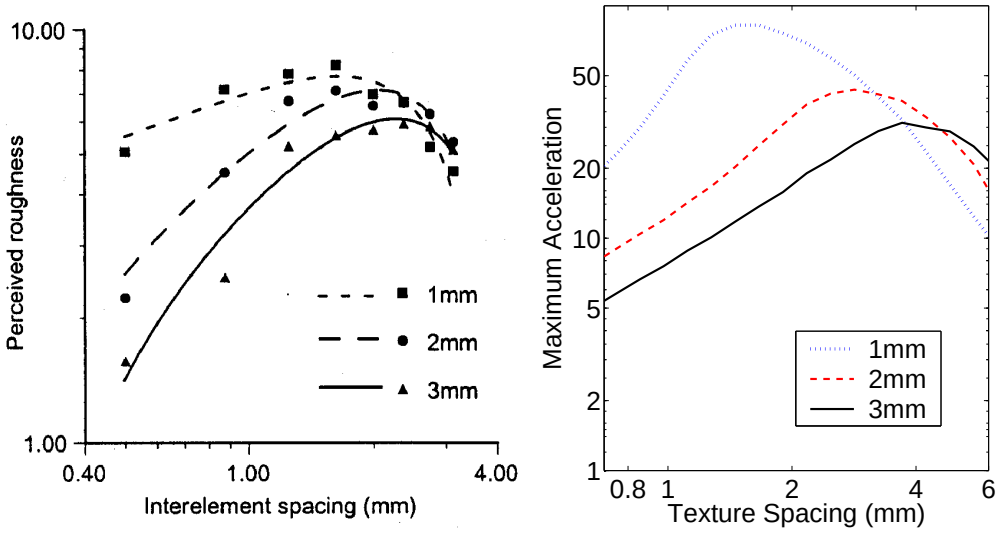

Figure 4.8: **Effects of Probe Diameter.** *Left: results of psychophysics studies by Lederman et al. [2000] (printed with permission of ASME and authors); Right: simulation results using a novel force model.*

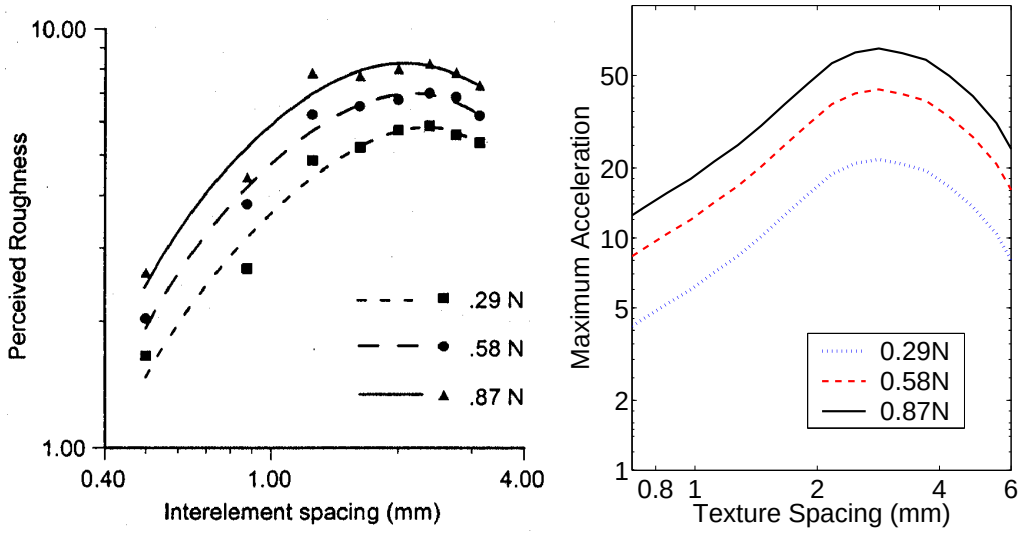

Figure 4.9: **Effects of Applied Force.** *Left: results of psychophysics studies by Lederman et al. [2000] (printed with permission of ASME and authors); Right: simulation results using a novel force model.*

Effects of Exploratory Speed

Fig. 4.10 compares the effects of exploratory speed on perceived roughness and on simulated acceleration. At large values of texture spacing, both perceived roughness and simulated acceleration increase as speed increases. The effects, however, do not match at small values of texture spacing. One would

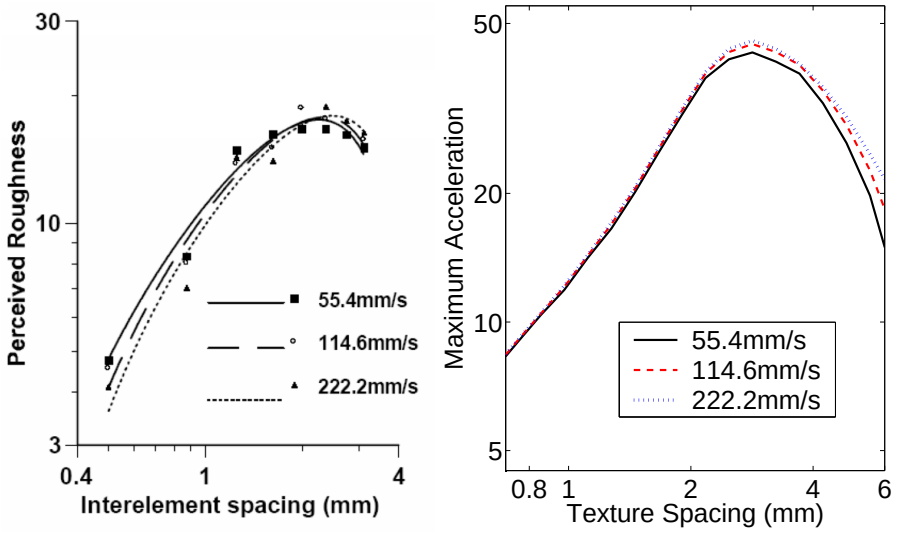

Figure 4.10: **Effects of Exploratory Speed.** *Left: results of psychophysics studies by Lederman et al. [1999] (printed with permission of Haptics-e and authors); Right: simulation results using a novel force model.*

expect simulated acceleration to be larger at lower speeds, but it remains almost constant.

Discussion

The effects of probe diameter and applied force on the motion induced by the force model for texture rendering presented in Sec. 4.3.2 match in a qualitative way the effects of these factors on perceived roughness of real textures. The results exhibit some differences on the effects of exploratory speed. These differences may be caused by limitations of the force model or limitations of the dynamic hand model employed in the simulations.

But the reason for these differences may also be that roughness is perceived as a combination of several physical variables, not solely acceleration. The complete connection between physical parameters, such as forces and motion, and a subjective metric of roughness is still unknown. Nevertheless, the analysis of the force model has been based on qualitative comparisons of locations and values of function maxima. This approach relaxes the need for a known relationship between acceleration and roughness. For example, if perceived roughness depends monotonically on acceleration in the interval of study, the maxima of roughness and acceleration will occur at the same values of texture spacing. This correlation is basically what I have found in the experiments.

4.5.2 Interactive Tests with Complex Models

I have performed experiments to test the performance of the texture force computation and the rendering algorithm in interactive demonstrations. The first set of experiments evaluates the conveyance of roughness effects under translational and rotational motion. The second set of experiments tests the performance of the haptic texture rendering algorithm and its GPU-based implementation in scenarios with complex contact configurations.

Besides these experiments, several subjects have used the haptic texture rendering system to identify texture patterns through haptic cues only. The reported experiences are promising, as subjects were able to successfully describe regular patterns such as ridges, but had more difficulty with irregular patterns. This result is what one expects when real, physical textured models are explored.

Implementation Details

The experiments have been performed using a 6-DoF *Phantom*^{TM} haptic device, a dual Pentium-4 2.4GHz processor PC with 2.0 GB of memory and an NVidia GeForce FX5950 graphics card, and Windows2000 OS. The penetration depth computation on graphics hardware is implemented using OpenGL plus OpenGL's ARB_fragment_program and GL_NV_occlusion_query extensions. The visual display of the scene cannot stall the haptic texture rendering process; hence, it requires a dedicated graphics card. The full-resolution scene is displayed on a separate commodity PC.

As described in Sec. 4.2.2, the first step in the computation of collision response is to find contact information between coarse-resolution models. In a general case, I do this using contact levels of detail (CLODs) for multiresolution collision detection, as described in Chapter 3. In these experiments, and for the purpose of testing the haptic texture rendering algorithm independently from CLODs, the models were simply described by coarse representations and haptic textures. For each benchmark model, I computed a bounding volume hierarchy (BVH) of convex hulls, equivalent to creating CLODs where all levels of the hierarchy are "free" CLODs (see Sec. 3.3.3).

Following the approach developed with Kim et al. [KOLM03], the contacts returned by the contact query are clustered, and contact points and penetration direction are computed for each cluster. This information is passed to the refinement step, where texture forces are computed, using the force

model and GPU-based implementation presented in this chapter. During texture force computation, I compute each value of penetration depth between contact patches on a 50×50, 16-bit depth buffer. This resolution proved to be sufficient based on the results.

The contact forces and torques of all contact patches are added to compute net force and torque, which are directly displayed to the user without a stabilizing intermediate representation. In this way the experiments do not get distorted by the use of intermediate representations, and the analysis can focus on the performance of the force model and the rendering algorithm. In Chapter 5, I will present how the texture force model is integrated with a stable and responsive force rendering algorithm.

Benchmarks Models and Scenarios

Models	Full Res. Tris	Low Res. Tris	Low Res. BVs
Block	65,536	16	1
Gear	25,600	1,600	1
Hammer	433,152	518	210
CAD Part	658,432	720	390
File	285,824	632	113
Torus	128,000	532	114

Table 4.1: **Complexity of Benchmark Models.** *Number of triangles at full resolution (Full Res. Tris) and low resolution (Low Res. Tris), and number of bounding volumes at low resolution (Low Res. BVs).*

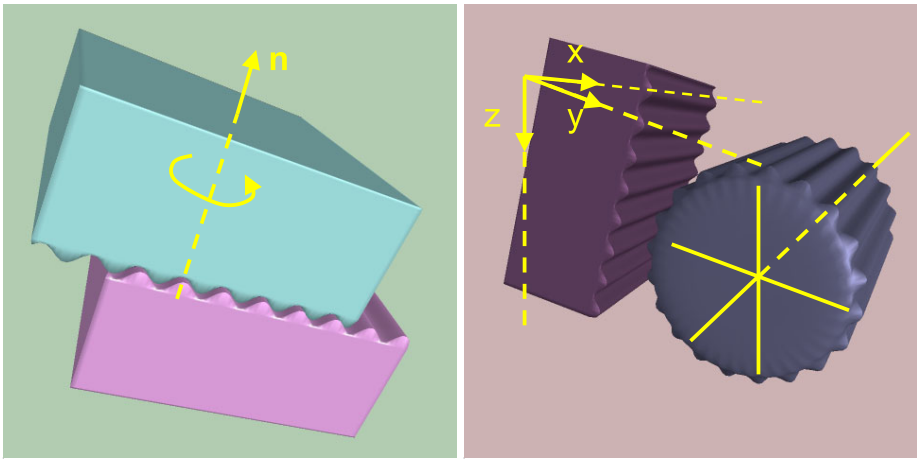

Figure 4.11: **Benchmark Models for Experiments on Conveyance of Roughness.** *Left (a): textured blocks; Right (b): block and gear.*

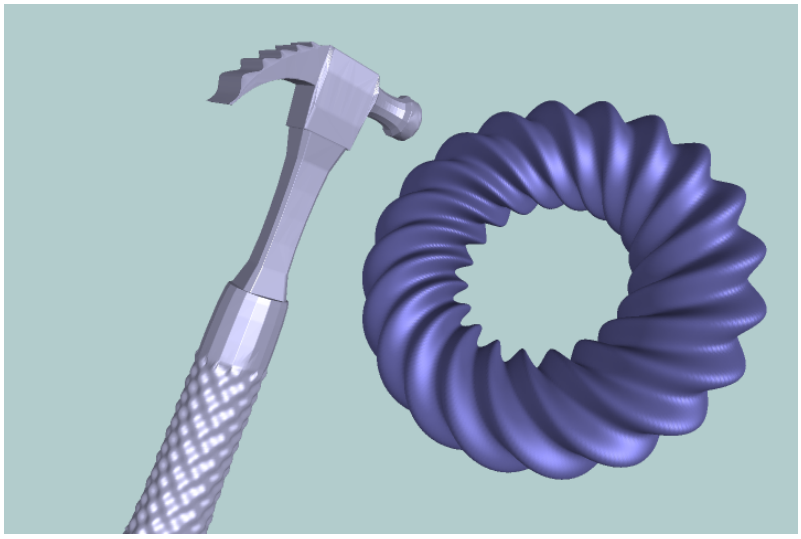

Figure 4.12: **Textured Hammer and Helicoidal Torus.** *Benchmark scenario for performance tests.*

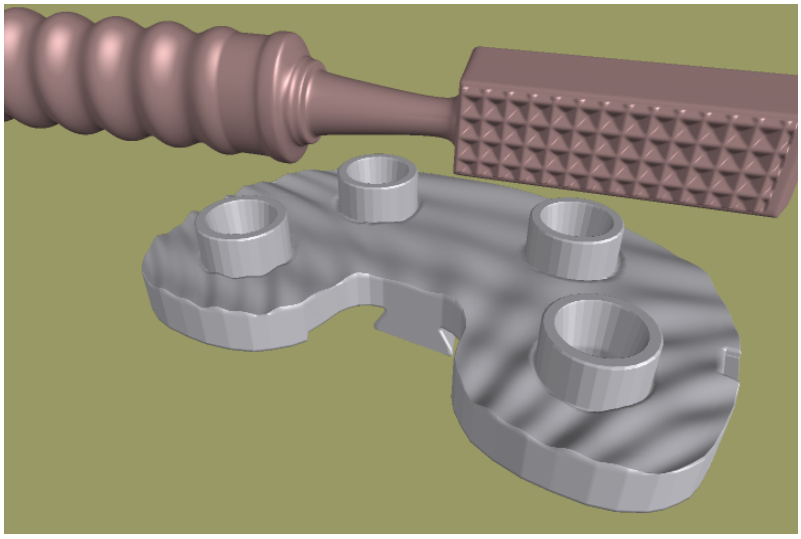

Figure 4.13: **File and CAD Part.** *Benchmark scenario for performance tests.*

For the experiments on conveyance of roughness, I have used the models shown in Fig. 4.11. The performance tests were executed on the models shown in Figs. 4.12 and 4.13. The complexities of the full-resolution textured models and their coarse resolution approximations are listed in Table 4.1.

Notice the drastic simplification of the low-resolution models. At this level all texture information is eliminated from the geometry, but it is stored in 1024×1024-size floating point textures. The number of BVs at coarse resolution reflects the geometric complexity for the collision detection module. Also notice that the *block* and *gear* models are fully convex at coarse resolution. The interaction

between these models is described by one single contact, so they are better suited for analyzing force and motion characteristics in the simulations.

Conveyance of Roughness under Translation

The gear and block models present ridges that interlock with each other. One of the experiments consisted of translating the block in the 3 Cartesian axes, while keeping it in contact with the fixed gear, as depicted in Fig. 4.11-b. Fig. 4.14 shows the position of the block and the force exerted on it during $1,500$ frames of interactive simulation (approx. 3 seconds).

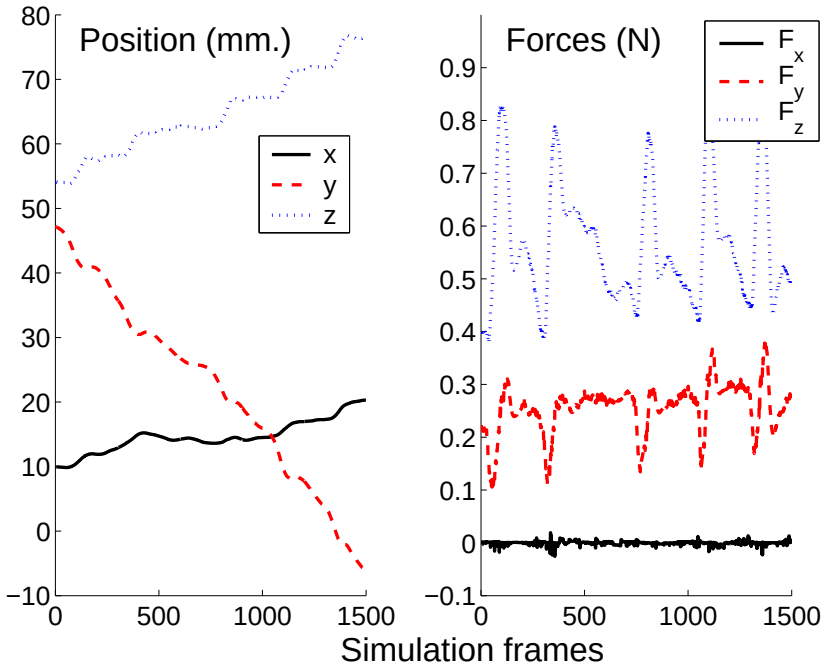

Figure 4.14: **Roughness under Translation.** *Position and force profiles generated while translating the model of a textured block in contact with a gear model, as shown in Fig. 4.11-b. Notice the staircase-like motion in z, and the correlation between force and position changes.*

Notice that the force in the *x* direction, which is parallel to the ridges, is almost zero. The texture force model successfully yields this expected result, because the derivative of the penetration depth is zero along the *x* direction. Notice also the staircase-like motion in the *z* direction, which reflects how the block rests for short periods of time on the ridges of the gear. The wide frequency spectrum of staircase-like motion is possible due to the fine spatial resolution of penetration depth and gradient computation. Last, the forces in *y* and *z* are correlated with the motion profiles.

Conveyance of Roughness under Rotation

Two identical striped blocks were placed interlocking each other, as shown in Fig. 4.11-a. Then I performed small oscillating rotations of the upper block around the direction **n**, and observed the induced translation along that same direction. Fig. 4.15 shows the rotation and translation captured during $6,000$ frames of interactive haptic simulation (approx. 12 seconds). Notice how the top block rises along **n** as soon as it is slightly rotated, thus producing a motion very similar to the one that occurs in reality. Previous point-based haptic rendering methods are unable to capture this type of effect. The texture force model presented in Sec. 4.3 successfully produces the desired effect by taking into account the local penetration depth between the blocks. Also, the derivative of the penetration depth produces a physically based torque in the direction **n** that opposes the rotation.

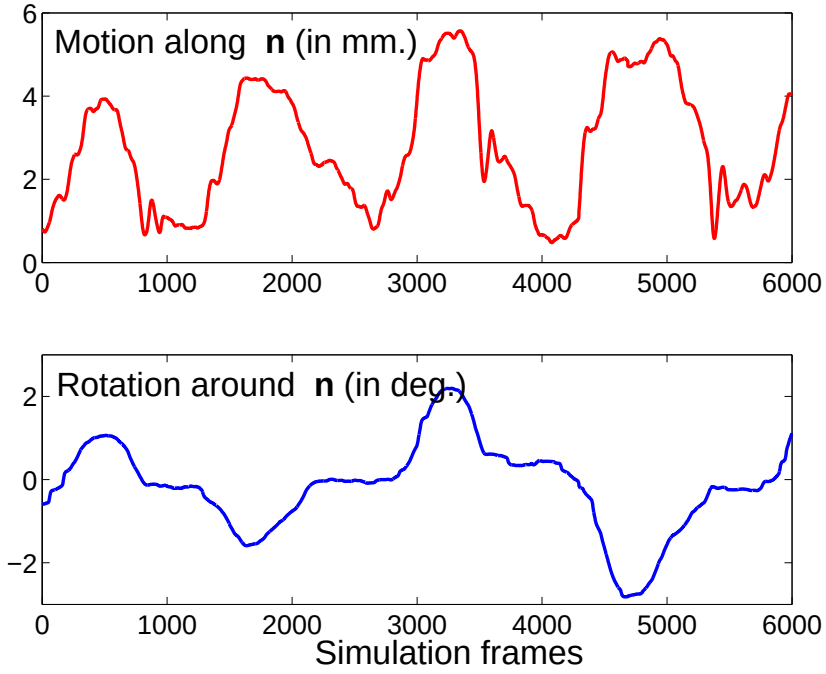

Figure 4.15: **Roughness under Rotation.** *Motion profile obtained by rotating one textured block on top of another one, as depicted in Fig. 4.11-a. Notice the translation induced by the interaction of ridges during the rotational motion.*

Performance Tests

In the experiments on conveyance of roughness, collision detection between the low-resolution models can be executed using fast algorithms that exploit the convexity of the models. As explained earlier, low-resolution contact is described by one contact point in each scenario, and the haptic update rate is approximately 500Hz.

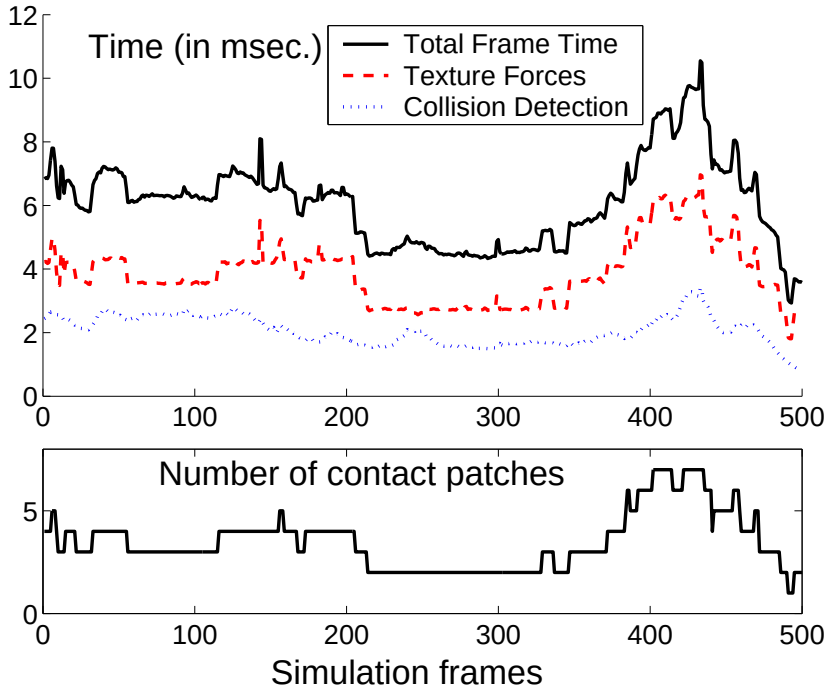

Figure 4.16: **Timings.** *Performance analysis and number of clustered contact patches during* 500 *simulation frames of a file model scraping a CAD part, as shown in Fig. 4.13. In this complex contact scenario the haptic frame rate varies between 100Hz and 200Hz.*

I have also tested the performance of the haptic texture rendering algorithm and its implementation in scenarios where the coarse resolution models present complex contact configurations. These scenarios consist of a file scraping a rough CAD part, and a textured hammer touching a wrinkled torus (See Figs. 4.13 and 4.12).

In particular, Fig. 4.16 shows timings for 500 frames of the simulation of the file interacting with the CAD part. The graph reflects the time spent on collision detection between the coarse-resolution models (an average of 2ms), the time spent on haptic texture rendering, and the total time per frame, which is approximately equal to the sum of the previous two. In this experiment, the

penetration depth for each contact is computed on a 50×50 16-bit buffer (See Sec. 4.4.2). As shown by the roughness conveyance experiments, this resolution proved to be sufficient to display convincing roughness stimuli.

In this particularly challenging experiment the haptic update rate varied between 100Hz and 200Hz. The dominant cost corresponds to haptic texture rendering, and it depends almost linearly on the number of contacts. The achieved force update rate may not be high enough to render textures with high spatial frequency, but, as shown above, the proposed force model enables perception of roughness stimuli that were not captured by earlier methods. Moreover, Fig. 4.16 shows performance results for a contact configuration in which large areas of the file at many different locations are in close proximity with the CAD part. In fact, collision detection using coarse-resolution models reports an average of 104 pairs of convex patches in close proximity, which are later clustered into as many as 7 contacts. Using the full-resolution models, the number of contact pairs in close proximity would increase by several orders of magnitude, and simply handling collision detection would become infeasible at the desired haptic rendering frame rates. Furthermore, as the support for programming on GPUs and capabilities of GPUs continue to grow at a rate faster than Moore's Law, the performance of 6-DoF haptic texture rendering is expected to reach kHz update rates in the near future.

4.6 Summary and Limitations

In this chapter I have presented a method to haptically render the interaction between textured objects. The interacting objects are described by simplified geometric representations, along with *haptic textures* (i.e., texture images storing geometric detail). The rendering algorithm first computes approximate contact information using the simplified representations, and then the contact information is refined using haptic textures.

In particular, I have proposed an image-based algorithm for computing directional penetration depth using haptic textures, along with a GPU-based implementation. Interobject penetration depth and its gradient are the central components of a perceptually inspired force model for 6-DoF haptic texture rendering. In this chapter I have described the design of the force model, guided by the results of perceptual studies by Klatzky and Lederman [KL02].

Offline simulations and analysis of induced acceleration have shown that the force model captures the influence of geometry and applied force on roughness perception, at least regarding the aspects described by psychophysics studies. Dynamic effects are also captured to some extent, but further analysis is necessary. The haptic rendering methodology and the force model have also proved to be successful in interactive haptic rendering, as demonstrated by the experiments on conveyance of roughness. Interactive rendering is enabled by the GPU-based implementation of penetration depth computation. Performance tests show haptic update rates of a few hundred Hz on complex benchmarks. Overall, and to the best of my knowledge, this is the first 6-DoF haptic rendering method capable of capturing texture effects during interaction between two objects.

The haptic texture rendering method is easily integrated with contact levels of detail (CLODs), described in Chapter 3. The low-resolution contact information is obtained by multiresolution collision detection, using CLODs, and then the penetration depth is refined and the texture force model is applied to compute contact forces. In Chapter 5, I will describe a stable and responsive rendering algorithm that will complete the system for haptic simulation.

The force model and implementation described in this chapter present a few limitations, some of which are common to existing haptic rendering methods. Next I discuss these limitations.

4.6.1 Limitations of the Force Model

In some contact scenarios with large contact areas, the definition of a local and directional penetration depth is not applicable. An example is the problem of screw insertion. In certain situations, such as contact between interlocking features, local geometry cannot be represented as height fields and the gradient of directional penetration depth may not capture the interlocking effects.

As shown in Sec. 4.5, in practice the force model generates forces that create a realistic perception of roughness for object-object interaction; however, one essential limitation of penalty-based methods and impedance-type haptic devices (such as the 6-DoF $Phantom^{TM}$ used in the experiments) is the inability to enforce motion constraints. The texture force model attempts to do so by increasing tangential contact stiffness when the gradient of penetration depth is high. But the stiffness delivered to the user must be limited, for stability purposes. New constraint-based haptic rendering techniques and perhaps other haptic devices [PC99] will be required to properly enforce constraints.

An important issue in every force model for haptic rendering is its stability. Choi and Tan [CT03a] have shown that even passive rendering algorithms may suffer from a problem called *aliveness*, induced by geometric discontinuities. Using haptic textures, discontinuities may arise if the contact patches cannot be described as height fields along the penetration direction, and these are possible sources of aliveness.

4.6.2 Frequency and Sampling Issues

As with other sample-based techniques, the haptic texture rendering algorithm is susceptible to aliasing problems. Here I discuss different aliasing sources and suggest some solutions.

Input textures

The resolution of input textures must be high enough to capture the highest spatial frequency of input models, although input textures can be filtered as a preprocessing step to downsample and reduce their size.

Image-based computation

In the height function computation step, buffer resolution must be selected so as to capture the spatial frequency of input models. Buffer size, however, has a significant impact in the performance of force computation.

Discrete derivatives

Penetration depth may not be a smooth function. This property results in an infinitely wide frequency spectrum, which introduces aliasing when sampled. Differentiation aggravates the problem, because it amplifies higher frequencies. The immediate consequence in the image-based implementation is that the input texture frequencies have to be low enough so as to faithfully represent their derivatives. This limitation is common to existing point-based haptic rendering methods [Min95] as well.

Temporal sampling

Force computation also undergoes temporal sampling. The Nyquist rate depends on object speed and spatial texture frequency. Image-based filtering prior to computation of penetration depth may remove undesirable high frequencies, but it may also remove low frequencies that would otherwise appear due to the non-linearity of the max search operation. In other words, filtering a texture with high frequency may incorrectly remove all torque and tangential forces. Temporal supersampling appears to be a solution to the problem, but is often infeasible due to the high update rates required by haptic simulation.

Chapter 5

Simulation of Rigid Body Dynamics with Haptic Manipulation

In this chapter, I present a 6-DoF haptic rendering pipeline that provides stable and responsive interaction. The quality of force-and-torque feedback in 6-DoF haptic rendering can be measured in terms of stability and responsiveness. On the one hand, stability is achieved by limiting the stiffness perceived by the user. On the other hand, responsiveness to collisions is achieved by using large stiffness values. A high update rate of force and torque feedback maximizes responsiveness by enabling stability at high stiffness values.

Collision detection is often the bottleneck in 6-DoF haptic rendering, and a fast execution of collision queries is crucial for maximizing the responsiveness of the system. In Chapters 3 and 4, I have presented fast algorithms for computing contact information between complex (textured) polygonal models. Direct haptic rendering of the collision response offers little control over the stiffness delivered to the user and the update rate of force feedback.

Several researchers have proposed *virtual coupling* [CSB95, AH98a, MPT99] for interfacing the synthesis of force feedback with the computation of contact forces. The object grasped by the user is not rigidly linked to the position of the haptic device. Instead, it suffers the action of both contact forces and coupling forces. Virtual coupling offers simple control of the stiffness delivered to the user, but the quality of force-and-torque feedback depends on the stability and responsiveness of the motion of the grasped object.

I propose implicit integration for rigid body simulation, along with virtual coupling and a lin-

earized model of penalty contacts, for computing stable and responsive motion of the grasped object. Implicit integration enables stable simulation with low mass and high stiffness values, thereby producing responsive motion. A linearized contact model permits a multirate architecture in which the computation of the motion of the grasped object is not subject to the bottleneck of collision detection. I have formulated linear approximations of coupling forces, penalty-based forces, and texture forces w.r.t. the state variables of a rigid body. These linear approximations are used by both implicit integration and the linearized contact model.

I have tested the stability and responsiveness of the algorithm in both free-space motion and contact state. I have tested the contribution of implicit integration and the linearized contact model on the performance of the system, by comparing simulation data under different settings. I have also successfully incorporated the multiresolution collision detection based on *contact levels of detail* (CLODs) into the rendering algorithm. Finally, I have derived an implicit formulation of 6-DoF haptic texture rendering that exhibits promising results but currently suffers from resolution limitations.

The rest of this chapter is organized as follows. In Sec. 5.1, I describe the multirate architecture of the 6-DoF haptic rendering pipeline and I introduce the notation used throughout the chapter. In Sec. 5.2, I formulate the implicit solution of the equations of motion of the grasped object. In Sec. 5.3, I present the force and torque equations of virtual coupling and their integration with the implicit formulation. Similarly, In Sec. 5.4, I present the integration of collision response with the implicit formulation, and I describe a contact clustering algorithm as well. In Sec. 5.5, I describe the experiments and results, and in Sec. 5.6, I discuss limitations of the proposed 6-DoF haptic rendering pipeline.

5.1 System Overview

In this section I give an overview of the entire haptic rendering pipeline, which integrates the techniques for contact determination and collision response presented in previous chapters, with modules responsible for handling user interaction and the motion of the manipulated object. I also present a multirate architecture that enables high force update rates. To conclude the section, I introduce some notation and terminology.

5.1.1 6-DoF Haptic Rendering Pipeline

Throughout this chapter I assume the use of an impedance-type haptic device. This means that the rendering pipeline receives the position, orientation, and velocities of the haptic device as inputs, and outputs force and torque commands to the device controller.

6-DoF haptic rendering implies bidirectional interaction with a virtual environment. On the one hand, the rendering system is responsible for computing the motion of the object grasped by the user, subject to geometric constraints imposed by the rest of the scene. On the other hand, it must synthesize force and torque fed back to the user. As discussed in Sec. 2.2, rendering contact with stiff virtual surfaces is a challenging task, prone to suffering instability problems. One of the key factors for successful display of stiff contact is very frequent updates of the feedback forces and the motion of the grasped object.

In the past, researchers in the haptic community have succeeded in enhancing the performance of rendering algorithms by dividing them into different modules, communicated through various interfaces. The haptic rendering pipeline presented in this dissertation follows the same divide-and-conquer strategy. As shown in Fig. 5.1, it presents three main components: virtual coupling, rigid body simulation, and collision detection.

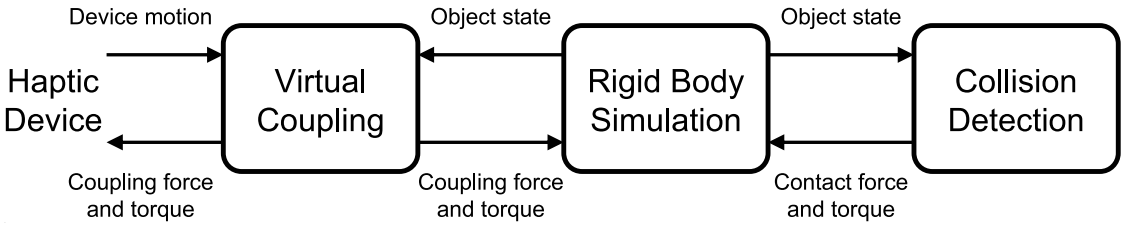

Figure 5.1: **Overview of the 6-DoF Haptic Rendering Pipeline.** *The haptic rendering pipeline is decomposed into* 3 *modules: virtual coupling, rigid body simulation and collision detection.*

Collision detection is often the bottleneck in rigid body simulation and, in this dissertation, I propose a rendering pipeline that decouples the computation of rigid body simulation from the computation of contact information. I follow the concept of *intermediate representation* [AKO95, MRF+96], creating a linearized contact model that is used for rigid body simulation. Wan and McNeely [WM03] also suggest a linearized contact model for 6-DoF haptic rendering, as part of a quasi-static approxi-

mation of rigid body motion. Chapters 3 and 4 present fast algorithms for computing contact information between complex polygonal models, based on multiresolution approaches that account for haptic perceptual factors. In Sec. 5.4 in this chapter, I describe the integration of these algorithms in the complete pipeline, and the computation of a linear approximation of contact forces.

Virtual coupling [CSB95, AH98a], a technique that separates the computation of the motion of the grasped object from the computation of feedback forces, ensures stable interaction if the simulation of the virtual environment is passive from an energy-transfer perspective [CS94]. The virtual coupling receives the state of the device and the grasped object as inputs, and generates coupling force and torque that are fed to both the rigid body simulation and the device controller. The parameters of the virtual coupling play a crucial role in the range of stable impedances or Z-width [CB94]. For example, when the grasped object collides with a virtual surface, the stiffness perceived by the user corresponds to the stiffness of the virtual coupling. In Sec. 5.3, I describe a viscoelastic 6-dimensional coupling in detail, and its integration in the pipeline.

As previously mentioned, faster update of the motion of the grasped object enables higher coupling stiffness. It is especially important to be able to maintain a nearly constant force update rate and, in this regard, penalty-based methods offer important advantages over other techniques for solving rigid body simulation, as discussed in Sec. 2.4. Therefore, in this chapter, I introduce a solution for the motion of the grasped object based on penalty methods. I combine this approach with implicit numerical integration, which provides attractive properties, such as passivity [CSB95] and higher stability under high contact stiffness [BW98].

5.1.2 Multirate Architecture

As indicated before, I have used an intermediate representation that decouples the computation of contact forces from the simulation of rigid body dynamics and synthesis of force feedback. These two tasks can run asynchronously, and the updates of contact forces are fed to the module that simulates rigid body dynamics. In this way, contact determination is not a bottleneck for the force synthesis, allowing higher update rates. I subdivide the haptic rendering pipeline into two main threads, shown in Fig. 5.2: a *haptic thread*, and a *contact thread*.

The haptic thread runs at a high frequency (1kHz in the experiments described in Sec. 5.5), com-

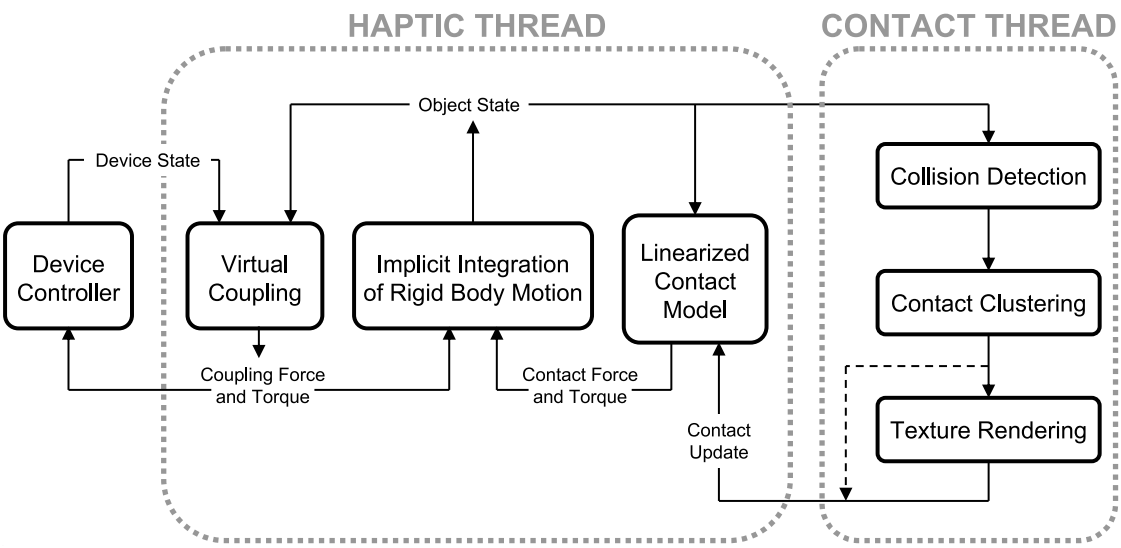

Figure 5.2: **Main Threads in Multirate Architecture.** *A haptic thread runs at force update rates simulating the rigid body dynamics of the grasped object and computing force feedback, while a contact thread runs asynchronously and updates contact forces.*

puting rigid body simulation and force feedback. Knowing the state of the grasped object at time t_{i-1}, the haptic thread computes the state of the grasped object at time t_i, and synthesizes force and torque commands sent to the device controller. Each frame, the haptic thread executes the following sequence of operations:

1. Read state of the haptic device.
2. Linearize the coupling force and torque at time t_{i-1}.
3. Linearize the contact force and torque at time t_{i-1}.
4. Solve the state of the grasped object at time t_i, by implicit integration.
5. Compute the coupling force and torque at time t_i.
6. Send the coupling force and torque to the device controller.

The contact thread runs asynchronously, at the highest frequency possible given the complexity of the contact scenario. To limit the cost of collision detection and response, while capturing perceptually relevant information, I have presented a two-step algorithm. First, I perform multiresolution

collision detection based on *contact levels of detail* (CLODs), obtaining approximate contact information. Second, I refine this contact information, taking into account surface texture details stored in *haptic textures*. The computation of approximate contact information using CLODs is accompanied of a clustering operation that outputs a set of representative contacts. Specifically, the contact thread performs the following sequence of operations every loop:

1. Fetch the state of the grasped object.
2. Perform multiresolution collision detection using CLODs.
3. Cluster contacts and compute cluster representatives.
4. For each cluster representative, compute texture force and torque, using haptic textures.
5. For each cluster representative, compute a linear approximation of the contact force and torque.

If the objects involved in collision detection do not present relevant geometric detail at texture level, the texture rendering step can be skipped, and one can compute a linear approximation of contact force and torque directly from the contact information of the cluster representatives. Similarly, the 6-DoF haptic rendering algorithm can be applied to both complex and simple models. For simple models, one could employ other collision detection algorithms, instead of CLODs.

5.1.3 Notation

In the remaining of this chapter, I use lower-case bold-face letters to represent vectors and quaternions, and upper-case letters to represent matrices. In matrix operations, vectors are in column form, and quaternions are treated as 4×1 vectors, unless I explicitly indicate that they are involved in quaternion products. Unless otherwise specified, all magnitudes are expressed in global coordinates of the virtual world. The superscript * applied to a vector, as in $\mathbf{u}^*$, indicates the skew-symmetric matrix used to represent a cross product as a matrix-vector product (see Sec. A.1.3 in the appendix). The appendix of this dissertation compiles together useful differentiation rules for vectors, matrices and rotations. In Table 5.1, I enumerate some of the notations I use throughout the chapter.

Notation	Meaning
$\mathbf{x}$	Position of the (center of mass of the) grasped object
$\mathbf{x}_h$	Position of the haptic device
$\mathbf{v}$	Linear velocity of the (center of mass of the) grasped object
$\mathbf{v}_h$	Linear velocity of the haptic device
$\mathbf{q}$	Orientation of the grasped object, expressed as a quaternion
$\mathbf{q}_h$	Orientation of the haptic device, expressed as a quaternion
R	Orientation of the grasped object, expressed as a rotation matrix
θ	Orientation of the grasped object, expressed as Euler angles
ω	Angular velocity of the grasped object
ω_h	Angular velocity of the haptic device
$\mathbf{P}$	Linear momentum
$\mathbf{L}$	Angular momentum
$\mathbf{F}$	Force
$\mathbf{F}_c$	Force exerted by the virtual coupling
$\mathbf{F}_p$	Penalty-based contact force
$\mathbf{F}_t$	Texture force
$\mathbf{T}$	Torque
$\mathbf{T}_c$	Torque exerted by the virtual coupling
$\mathbf{T}_p$	Penalty-based contact torque
$\mathbf{T}_t$	Texture torque
m	Mass of the grasped object
M	Mass matrix of the grasped object, expressed in its local frame
k	Contact stiffness
b	Contact damping
k_c	Linear stiffness of the virtual coupling
b_c	Linear damping of the virtual coupling
k_θ	Angular stiffness of the virtual coupling
b_θ	Angular damping of the virtual coupling
C	A contact
S	A contact cluster

Table 5.1: **Notation Table**

5.2 Simulation of Rigid Body Dynamics

This section discusses the simulation of rigid body dynamics based on penalty methods. First, I formulate the equations of motion; then I present an implicit solution. Since implicit numerical integration requires the evaluation of the Jacobian of a system of ODEs, I also introduce the formulation of this Jacobian, and I derive the terms induced by the non-linearity of rotation. I conclude the section with the formulation of the linearized contact model.

5.2.1 Equations of Rigid Body Motion

The Newton-Euler equations (see Eq. 2.2) define the motion of a rigid body as a function of external forces and torques. As noted by Mirtich [Mir96], the Euler equation defines the derivative of the angular velocity expressed in a local frame of the object. Baraff and Witkin [BW01] point out that the discretization and numerical integration of rigid body motion can be implemented more efficiently if one expresses the differential equations in terms of the derivatives of linear and angular momentum, instead of velocities. Moreover, they propose quaternions for describing object orientation more efficiently. Following the numerical integration scheme suggested by Baraff and Witkin, I formulate the state of a rigid body in terms of the position of the center of mass, **x**, a quaternion describing the orientation, **q**, the linear momentum, **P**, and the angular momentum, **L**. This formulation yields a state vector **y** with 13 variables. Consequently, the motion of the rigid body is described as a function of external forces **F** and torques **T** by the following set of ODEs:

$$\dot{\mathbf{y}}(t) = \begin{pmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{q}} \\ \dot{\mathbf{P}} \\ \dot{\mathbf{L}} \end{pmatrix} = \begin{pmatrix} \frac{1}{m}\mathbf{P} \\ \frac{1}{2}\omega_q \mathbf{q} \\ \mathbf{F} \\ \mathbf{T} \end{pmatrix}, \tag{5.1}$$

where m is the mass of the body. The term ω_q indicates a quaternion with scalar part 0 and vector part the angular velocity ω. Please refer to Sec. A.2.6 for an explanation of the equation of the derivative of the quaternion, $\dot{\mathbf{q}}$. Given the rotation matrix R and the mass matrix M of the body, its angular velocity ω can be expressed in terms of state variables as:

$$\omega = RM^{-1}R^T\mathbf{L}. \tag{5.2}$$

This dissertation focuses on 6-DoF haptic rendering for virtual exploration and manipulation. As discussed in Sec. 1.1.3 in the introduction, assembly and maintainability assessment in virtual prototyping, as well as surgical training, are some of the virtual manipulation tasks that can benefit from 6-DoF haptic feedback. In these applications the environment is often static. Based on this fact, I simplify the problem of rigid body simulation by assuming that the only moving object is the grasped

object.

In the scope of this dissertation, the external forces (and similarly for the torques) comprise the weight of the object, contact forces, and the coupling force. Other terms, such as friction, could also be added. The simplest way to incorporate friction into the formulation of external forces would be by using a local friction model [HA00]. Contact forces may be computed as simple penalty-based forces (see Sec. 5.4.3), or as texture-induced forces (see Sec. 5.4.4). The total force and torque on the grasped object, assuming contacts between both textured and non-textured surfaces, are computed as:

$$\begin{aligned} \mathbf{F} &= \mathbf{F}_c + \sum_{i=1}^{m} \mathbf{F}_{p,i} + \sum_{j=1}^{n} \mathbf{F}_{t,j} + m\mathbf{g}, \\ \mathbf{T} &= \mathbf{T}_c + \sum_{i=1}^{m} \mathbf{T}_{p,i} + \sum_{j=1}^{n} \mathbf{T}_{t,j}, \end{aligned} \tag{5.3}$$

where $\mathbf{F}_c$ and $\mathbf{T}_c$ represent the force and torque exerted by the virtual coupling, $\mathbf{F}_{p,i}$ and $\mathbf{T}_{p,i}$ represent the penalty force and torque at the i-th contact, and $\mathbf{F}_{t,j}$ and $\mathbf{T}_{t,j}$ represent the texture-induced force and torque at the j-th contact.

5.2.2 Implicit Integration

The system of ODEs describing rigid body motion can be represented in a vector form as:

$$\dot{\mathbf{y}}(t) = \mathbf{f}(t). \tag{5.4}$$

Implicit discretization of the ODEs using the Backward Euler formula yields the following equation for the state vector:

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t \dot{\mathbf{y}}_n. \tag{5.5}$$

Substituting Eq. 5.1 in Eq. 5.5 leads to a non-linear equation in the state variables **x**, **q**, **P** and **L**. A non-linear solver, such as Newton's method, can be used to find the exact solution to this system. However, I have decided to trade accuracy for speed, and linearly approximate Eq. 5.5 using the Taylor expansion of **f**. This approximation leads to a semi-implicit Backward Euler discretization, in which

$\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ is the Jacobian of the equations of rigid body motion.

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t \left(\mathbf{f}_{n-1} + \frac{\partial \mathbf{f}}{\partial \mathbf{y}} (\mathbf{y}_n - \mathbf{y}_{n-1}) \right). \tag{5.6}$$

Rearranging terms, this linear system of equations can be expressed in the form:

$$\left(I - \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \right) \Delta \mathbf{y} = \Delta t \mathbf{f}_{n-1}. \tag{5.7}$$

Under the assumption that the grasped object is the only moving object, $\left(I - \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \right)$ is a 13×13 dense and non-symmetric matrix. The linear system can be solved by Gaussian elimination. To summarize, semi-implicit Backward Euler integration of a system of ODEs requires the following steps per frame:

1. Compute the derivatives for the previous frame, $\mathbf{f}_{n-1}$.
2. Update the Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$.
3. Solve the linear system for $\mathbf{y}_n - \mathbf{y}_{n-1}$.

Following Eq. 5.1, the evaluation of $\mathbf{f}_{n-1}$ consists mainly of computing coupling and contact force and torque. The remaining of this section focuses on the formulation of the Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$.

5.2.3 Jacobian of Rigid Body Motion

I decompose the Jacobian into different blocks, in a way similar to Larsen [Lar01]. From Eq. 5.1, the Jacobian can be expressed as:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{y}} = \begin{pmatrix} \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{x}} & \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{q}} & \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{P}} & \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{L}} \\ \frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{x}} & \frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{q}} & \frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{P}} & \frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{L}} \\ \frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{x}} & \frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{q}} & \frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{P}} & \frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{L}} \\ \frac{\partial \dot{\mathbf{L}}}{\partial \mathbf{x}} & \frac{\partial \dot{\mathbf{L}}}{\partial \mathbf{q}} & \frac{\partial \dot{\mathbf{L}}}{\partial \mathbf{P}} & \frac{\partial \dot{\mathbf{L}}}{\partial \mathbf{L}} \end{pmatrix} = \begin{pmatrix} 0 & 0 & \frac{1}{m} I & 0 \\ 0 & \frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{q}} & 0 & \frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{L}} \\ \frac{\partial \mathbf{F}}{\partial \mathbf{x}} & \frac{\partial \mathbf{F}}{\partial \mathbf{q}} & \frac{\partial \mathbf{F}}{\partial \mathbf{P}} & \frac{\partial \mathbf{F}}{\partial \mathbf{L}} \\ \frac{\partial \mathbf{T}}{\partial \mathbf{x}} & \frac{\partial \mathbf{T}}{\partial \mathbf{q}} & \frac{\partial \mathbf{T}}{\partial \mathbf{P}} & \frac{\partial \mathbf{T}}{\partial \mathbf{L}} \end{pmatrix}. \tag{5.8}$$

As one can deduce from combining Eqs. 5.3 and 5.8, the evaluation of the Jacobian requires the computation of the Jacobians of external forces (and torques). Sections 5.3 and 5.4 deal, respectively,

with coupling forces and contact forces.

Non-linearity of the Orientation

The expression of the derivative of orientation, $\dot{\mathbf{q}}$, is highly non-linear and leads to two non-zero blocks in the Jacobian, as shown in Eq. 5.8. The expression of $\dot{\mathbf{q}}$ can be rewritten as a matrix-vector multiplication:

$$\dot{\mathbf{q}} = \frac{1}{2}\omega_q \mathbf{q} = Q\omega. \tag{5.9}$$

The definition of the 4×4 matrix Q can be found in Eq. A.25 in the appendix. Substituting ω from Eq. 5.2 leads to a linear equation in $\mathbf{L}$, from which the derivative w.r.t. $\mathbf{L}$ is easily obtained:

$$\dot{\mathbf{q}} = QRM^{-1}R^T\mathbf{L}, \tag{5.10}$$

$$\frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{L}} = QRM^{-1}R^T. \tag{5.11}$$

The term $\frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{q}}$ of the Jacobian is also derived from Eq. 5.10. It is convenient to express its derivative w.r.t. each component q_i of $\mathbf{q}$ independently as:

$$\frac{\partial \dot{\mathbf{q}}}{\partial q_i} = \frac{\partial Q}{\partial q_i}\omega + Q\frac{\partial \omega}{\partial q_i}. \tag{5.12}$$

The derivatives of ω are computed as:

$$\frac{\partial \omega}{\partial q_i} = \left(\frac{\partial R}{\partial q_i}M^{-1}R^T + RM^{-1}\frac{\partial R}{\partial q_i}^T\right)\mathbf{L}. \tag{5.13}$$

The derivatives $\frac{\partial Q}{\partial q_i}$ and $\frac{\partial R}{\partial q_i}$ can easily be derived from the matrices Q and R, and are defined in the appendix.

5.2.4 Linearized Contact Model

In Sec. 5.1.1, I have proposed the use of a linearized contact model as an intermediate representation between the computation of collision response and the computation of rigid body motion. In complex contact configurations collision detection may easily run at rates notably slower than the update of rigid body motion. In such cases, linear approximations of the contact forces increase the accuracy of the derivatives of state variables, and thereby the stability of implicit integration. Assuming that the last update of contact force and torque took place at time t, each of the terms $\mathbf{F}_{p,i}$ in Eq. 5.3 (and similarly for $\mathbf{T}_{p,i}$, $\mathbf{F}_{t,j}$, and $\mathbf{T}_{t,j}$) at time $t+\Delta t$ can be linearly approximated using their Taylor expansion as:

$$\mathbf{F}_{p,i}(t+\Delta t) = \mathbf{F}_{p,i}(t) + \frac{\partial \mathbf{F}_{p,i}}{\partial \mathbf{y}}(t)\left(\mathbf{y}(t+\Delta t) - \mathbf{y}(t)\right). \tag{5.14}$$

The Jacobians of contact forces and torques w.r.t. state variables must also be computed for the semi-implicit formulation of Backward Euler. Therefore, the computation of the linearized contact model has little additional cost.

5.3 Virtual Coupling

In this section I describe the equations of coupling force and torque that enable bidirectional interaction with a grasped object. I also formulate the linear approximation of the coupling force and torque, which is used in the implicit integration of rigid body motion. To conclude the section, I discuss issues associated with the synthesis of force feedback from a virtual coupling.

5.3.1 Coupling Force and Torque

When an object is grasped, the state of the haptic device in the virtual world is recorded as a coupling frame (coupling position $\mathbf{c}$ and coupling orientation $\mathbf{q}_c$) in the local coordinates of the object. The action of "grasping" an object with the haptic device is depicted in Fig. 5.3-a. During manipulation, as seen in Fig. 5.3-b, a viscoelastic link between the state of the haptic device and the state of the coupling frame serves as a virtual coupling, which produces bidirectional interaction. On the one

hand, the virtual coupling exerts coupling force and torque on the grasped object, so that it "follows" the haptic device. On the other hand, the same coupling force and torque are sent as commands to the device controller, in order to produce kinesthetic feedback.

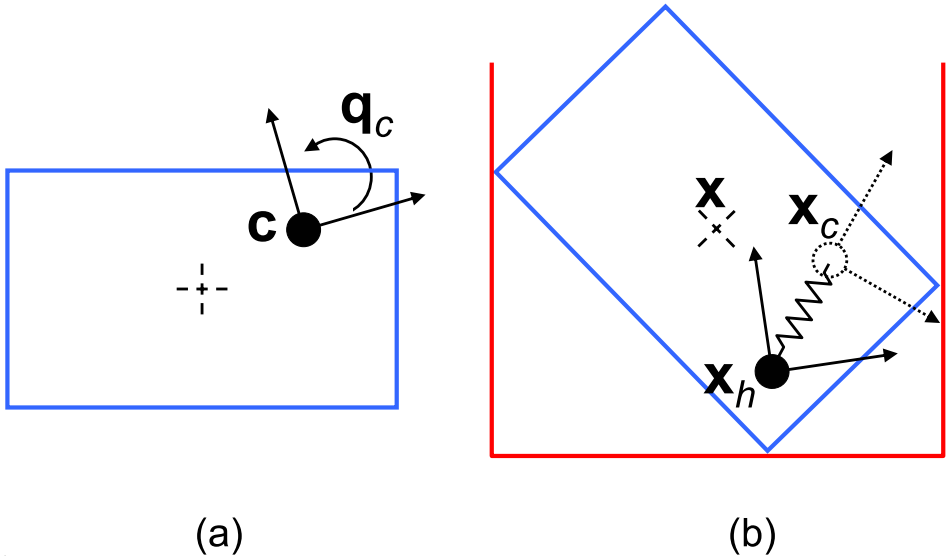

Figure 5.3: **Virtual Coupling.** *(a) The coupling position* **c** *and the coupling orientation* $\mathbf{q}_c$ *are set when the object is grasped; (b) During object manipulation, a coupling force is exerted based on the coupling deviation between the position of the haptic device* $\mathbf{x}_h$ *and the position of the coupling point* $\mathbf{x}_c$.

I assume that an object can be grasped by attaching a virtual coupling at any point in the object. In this way, the coupling force is set as a viscoelastic link between the current position of the haptic device and the position of the coupling point. The coupling torque is composed of the torque induced by the coupling force $\mathbf{F}_c$, and a viscoelastic rotational link between the current orientation of the haptic device and the current orientation of the coupling frame. The rotational link can be expressed in terms of its equivalent axis of rotation, $\mathbf{u_c}$. The magnitude of $\mathbf{u_c}$ represents the coupling angle. The coupling force and torque equations are:

$$\begin{aligned}\mathbf{F}_c &= k_c(\mathbf{x}_h - \mathbf{x}_c) + b_c(\mathbf{v}_h - \mathbf{v}_c) \\ &= k_c(\mathbf{x}_h - \mathbf{x} - R\mathbf{c}) + b_c(\mathbf{v}_h - \mathbf{v} - \omega \times \mathbf{c}),\end{aligned} \tag{5.15}$$

$$\mathbf{T}_c = (R\mathbf{c}) \times \mathbf{F}_c + k_\theta \mathbf{u_c} + b_\theta(\omega_h - \omega), \tag{5.16}$$

where k_c and b_c represent linear stiffness and damping respectively; k_θ and b_θ represent angular stiffness and damping respectively; and $\mathbf{x}_h$, $\mathbf{v}_h$, and ω_h represent the position, linear velocity, and angular velocity of the haptic device.

As described by Colgate et al. [CSB95], and later generalized by Adams and Hannaford [AH98a], viscoelastic virtual coupling not only produces bidirectional interaction in a very simple way, but it also simplifies the design of a stable haptic rendering system. The coupling stiffness k_c is set as high as possible, while guaranteeing stability of the complete human-in-the-loop system. The coupling damping is tuned to obtain critically damped behavior. Other types of interaction paradigms are also possible, such as constraining the coupling point to be the center of mass of the grasped object [MPT99].

Equivalent Axis of Rotation

I refer to the rotation between the haptic device and the current orientation of the coupling frame as $\Delta\mathbf{q}$, with vector part $\Delta\mathbf{q}_{xyz}$ and scalar part Δq_s. This quaternion can be defined in terms of the equivalent axis of rotation as:

$$\Delta\mathbf{q} = (\Delta\mathbf{q}_{xyz}, \Delta q_s) = \left(\sin\left(\frac{\|\mathbf{u_c}\|}{2}\right)\frac{\mathbf{u_c}}{\|\mathbf{u_c}\|}, \cos\left(\frac{\|\mathbf{u_c}\|}{2}\right)\right). \tag{5.17}$$

Reversing the definition yields:

$$\mathbf{u_c} = 2\cos^{-1}(\Delta q_s)\Delta\mathbf{q}_{xyz}. \tag{5.18}$$

$\Delta\mathbf{q}$ can be expressed in terms of the current orientations and the coupling orientation as a product of quaternions:

$$\Delta\mathbf{q} = \mathbf{q}_h\mathbf{q}_c^{-1}\mathbf{q}^{-1}. \tag{5.19}$$

Or, as a linear transformation on the current orientation:

$$\Delta\mathbf{q} = C\mathbf{q}, \qquad \Delta\mathbf{q}_{xyz} = C_{123}\mathbf{q}, \qquad \Delta\mathbf{q}_s = C_4\mathbf{q}, \tag{5.20}$$

where C_{123} is the 3×4 submatrix built with the first 3 rows of C, and C_4 represents the last row of C.

5.3.2 Jacobian of Virtual Coupling

As explained in Sec. 5.2.3, the evaluation of the Jacobian of the equations of rigid body motion requires the computation of the Jacobian of coupling force and torque every frame. Here I list the derivatives of coupling force and torque w.r.t. the different state variables.

Derivatives w.r.t. Position

Following Eq. 5.15, the coupling force is simply linear on the position of the probe, so the term in the Jacobian is easily obtained as:

$$\frac{\partial \mathbf{F}_c}{\partial \mathbf{x}} = -k_c I \tag{5.21}$$

The torque term only depends on the position through the coupling force, so the corresponding term in the Jacobian is:

$$\frac{\partial \mathbf{T}_c}{\partial \mathbf{x}} = (R\mathbf{c})^* \frac{\partial \mathbf{F}_c}{\partial \mathbf{x}} = -k_c (R\mathbf{c})^*. \tag{5.22}$$

Derivatives w.r.t. Quaternion

The coupling force, as written in Eq. 5.15, depends on the orientation both through the stiffness and the damping terms. The derivative of the force w.r.t. each component q_i of the quaternion is of the form:

$$\frac{\partial \mathbf{F}_c}{\partial q_i} = -k_c \frac{\partial R}{\partial q_i}\mathbf{c} + b_c \mathbf{c}^* \frac{\partial \omega}{\partial q_i}. \tag{5.23}$$

Each column of the torque term, derived from Eq. 5.16, is expressed as:

$$\frac{\partial \mathbf{T}_c}{\partial q_i} = (R\mathbf{c})^* \frac{\partial \mathbf{F}_c}{\partial q_i} - \mathbf{F}_c{}^* \frac{\partial R}{\partial q_i}\mathbf{c} + k_\theta \frac{\partial \mathbf{u_c}}{\partial q_i} - b_\theta \frac{\partial \omega}{\partial q_i}. \tag{5.24}$$

It remains to compute the derivative of the rotation axis. From Eqs. 5.18 and 5.20, one can obtain the following derivative:

$$\frac{\partial \mathbf{u_c}}{\partial \mathbf{q}} = 2\cos^{-1}\Delta q_s C_{123} - \frac{2}{\sqrt{1-\Delta q_s{}^2}}\Delta \mathbf{q}_{xyz} C_4. \tag{5.25}$$

Derivatives w.r.t. Linear Momentum

The terms corresponding to the linear momentum present some similarities to the position terms, and are defined as:

$$\frac{\partial \mathbf{F}_c}{\partial \mathbf{P}} = -\frac{b_c}{m} I, \tag{5.26}$$

$$\frac{\partial \mathbf{T}_c}{\partial \mathbf{P}} = (R\mathbf{c})^* \frac{\partial \mathbf{F}_c}{\partial \mathbf{P}} = -\frac{b_c}{m}(R\mathbf{c})^*. \tag{5.27}$$

Derivatives w.r.t. Angular Momentum

The force term can be obtained substituting Eq. 5.2 in Eq. 5.15:

$$\frac{\partial \mathbf{F}_c}{\partial \mathbf{L}} = b_c \mathbf{c}^* R M^{-1} R^T. \tag{5.28}$$

The torque term contains both a force-related term, and a purely rotational term:

$$\begin{aligned}\frac{\partial \mathbf{T}_c}{\partial \mathbf{L}} &= (R\mathbf{c})^* \frac{\partial \mathbf{F}_c}{\partial \mathbf{L}} - b_\theta \frac{\partial \omega}{\partial \mathbf{L}} \\ &= (b_c (R\mathbf{c})^* \mathbf{c}^* - b_\theta I) R M^{-1} R^T.\end{aligned} \tag{5.29}$$

5.3.3 Synthesis of Force Feedback

As noted at the beginning of this section, virtual coupling enables bidirectional interaction, by exerting forces on the grasped object and, at the same time, synthesizing force feedback values. Following the sequence of operations in the haptic thread, as listed in Sec. 5.1.2, every frame I compute the coupling force and torque for the previous frame, then I formulate the derivatives involved in the Jacobian, and I solve the state of the grasped object for the present frame. The next step is to compute the coupling force and torque for the current frame, based on Eqs. 5.15 and 5.16, but using the newly computed object state. These force and torque values are sent to the device controller as feedback commands.

Scaling the Workspace

In many practical applications, the limited workspace of the haptic device must be scaled so that it can cover appropriately the virtual workspace. In that case, feedback forces must undergo an inverse scaling, to ensure that the coupling stiffness perceived by the user is the same as the coupling stiffness in the virtual world.

Non-linear Coupling

Haptic devices present physical limitations that should also be accounted for in the design of virtual coupling. Force (and torque) saturation is a clear example. When the user pushes against a virtual surface and the device reaches its maximum force value, the user feels no difference as a result of pushing further. The coupling force in the rigid body simulation, however, keeps growing, and so does object interpenetration. To avoid this, I suggest modeling the coupling stiffness as a non-linear function. This technique is applicable both to translational and rotational stiffness and, for reference, here I show the implications in the formulation of the translational coupling. Wan and McNeely [WM03] followed a similar approach, motivated by the fact that their voxel-based collision detection module does not detect interpenetration once that the grasped object penetrates further than one voxel.

One possibility is to simply limit the value of $\mathbf{F}_{cx}$, the stiffness-related term of the coupling force, to be the maximum force that can be exerted by the device, scaled appropriately. Wan and McNeely suggest a stiffness function k_c that decays exponentially with the coupling deviation $\Delta\mathbf{x}$. Instead, I

propose a spline force function: (1) for small deviations, under the saturation value, a linear force equation; (2) a cubic interpolating force equation; (3) and, for large deviations, a constant saturated force. Fig. 5.4 shows an example of force function. The cubic polynomial can easily be obtained by Hermite interpolation.

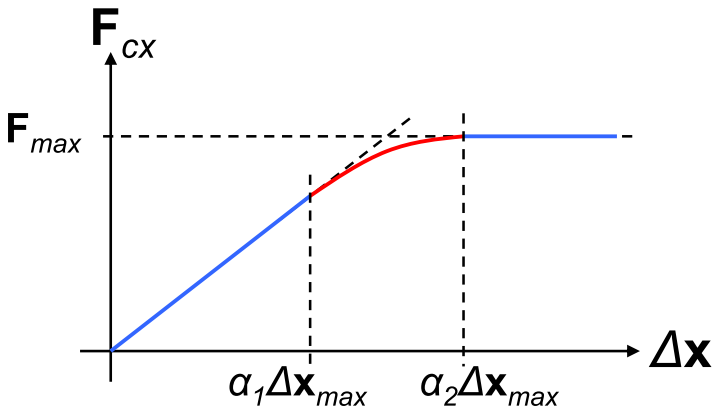

Figure 5.4: **Non-linear Coupling Stiffness.** *Piecewise cubic coupling force for resolving effects of force saturation.*

The Jacobian of virtual coupling must be revised, to account for the non-linearity of the stiffness. From Eq. 5.15, the stiffness-related term of the coupling force, $\mathbf{F}_{cx}$, is:

$$\mathbf{F}_{cx} = k_c(\mathbf{x}_h - \mathbf{x} - R\mathbf{c}) = k_c\Delta\mathbf{x}. \tag{5.30}$$

Considering k_c to be a non-linear function of $\Delta\mathbf{x}$ itself, the Jacobian of $\mathbf{F}_{cx}$ is expressed (following differentiation rules shown in Sec. A.1.6 in the appendix) as:

$$\frac{\partial \mathbf{F}_{cx}}{\partial \mathbf{y}} = \left(k_c I + \Delta\mathbf{x}\frac{\partial k_c}{\partial \Delta\mathbf{x}}\right)\frac{\partial \Delta\mathbf{x}}{\partial \mathbf{y}}. \tag{5.31}$$

5.4 Collision Response

The haptic thread, which computes the dynamics of the grasped object, receives the values of the contact force and torque and their Jacobians as the outputs from a linearized contact model. This contact model is updated asynchronously in the contact thread. In this section I describe the collision response module, which computes the contact forces and torques and their Jacobians. First, I define

the contact data output by the collision detection module. Second, I describe a contact-clustering algorithm that generates representative contacts for collision response. Last, I present two types of collision response: viscoelastic penalty-based force computation, and computation of texture-induced forces based on the force model presented in Chapter 4. In both cases, I formulate the contact forces and torques and their Jacobians w.r.t. the state variables of the grasped object.

5.4.1 Contact Determination

For the purpose of synthesizing penalty-based collision response, I describe a *contact C* between the grasped object and an object in the scene by means of the following parameters:

- A point **p** in the surface of the grasped object.
- A point $\mathbf{p}_0$ in the surface of the object in the scene.
- The contact normal **n**, pointing out of the grasped object.
- The penetration depth δ for the contact.

Contacts are obtained as the result of a *contact query* between the grasped object and the rest of the scene using CLODs. As described in Sec. 3.4.1, a contact query returns a set of contacts that sample the regions of the objects that are closer than a distance tolerance d. The existence of a tolerance implies that the penetration depth δ may be positive (if **p** lies inside the scene object) or negative (if **p** lies outside, but closer than d). Fig. 5.5-(a) shows an example of contact between the grasped object and the scene, as well as the contact parameters.

A contact query may output multiple contacts to describe each contact region. Penalty-based collision response, as described in Sec. 5.4.3, produces a viscoelastic force at each contact, and the forces of all contacts must be added together and applied to the grasped object. With penalty-based methods, discontinuities in the number of contacts affect the stability of the simulation, because the total stiffness depends on the number of contacts. McNeely et al. [MPT99] suggested limiting the total stiffness after reaching a certain number of contacts. Kim et al. [KOLM03] proposed a proximity-based clustering technique that reduces the number of representative contacts. Similarly, I propose a proximity-based clustering technique that computes a set of representative contacts, one per cluster,

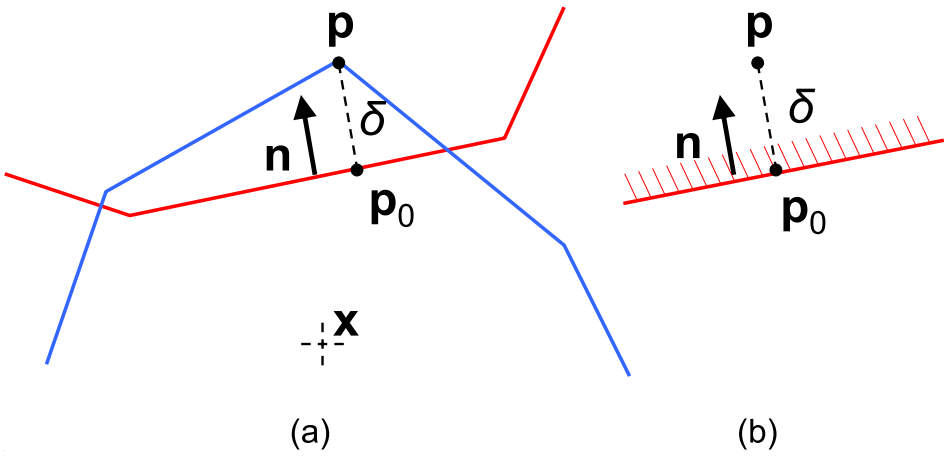

Figure 5.5: **Penalty-Based Contact Model.** *(a) As the grasped object (in blue) penetrates an object in the scene (in red), the contact is defined by the penetration depth, δ, the contact normal* **n***, and contact points* **p** *and* $\mathbf{p}_0$*; (b) Modeling the contact as a plane constraint.*

but I limit the number of output clusters in order to bound the total contact stiffness applied to the grasped object.

5.4.2 Contact Clustering

I propose a contact clustering technique based on *K*-means clustering [JMF99]. Given a set of *n* contacts $\{C_0, C_1, ...C_{n-1}\}$, I define *K* clusters $\{S_0, S_1, ...S_{K-1}\}$, and I compute a representative contact for each cluster. Contact forces, either simple penalty-based or texture-based, are computed at the representative contacts. If the number of input contacts is $n < K$, I only create *n* clusters.

I define the clusters implicitly, by storing an additional parameter along with each contact *C*: the cluster it belongs to, *S*. Then, I define a contact *C* as a tuple $(\mathbf{p}, \mathbf{p}_0, \mathbf{n}, \delta, S)$. In the description of the clustering algorithm, I will reference each parameter of a contact as $C.parameter$ (e.g., $C.\mathbf{p}$). Similarly, I define a cluster *S* as a tuple $(\mathbf{p}, \mathbf{p}_0, \mathbf{n}, \delta, w)$, where **p**, $\mathbf{p}_0$, **n**, and δ are the contact parameters of the cluster representative, and *w* is the accumulated weight of the cluster.

I formulate the *K*-means clustering based on the Euclidean distance between each contact point $C.\mathbf{p}$ and the representative of the cluster it belongs to, $C.S.\mathbf{p}$. I have defined a probability function that assigns higher probabilities to contacts with larger penetration depth δ. This strategy is beneficial for increasing the smoothness of penalty-based collision response. The cost function of the *K*-means

clustering problem can be written as:

$$f = \frac{\sum_i^{n-1} \left((C_i.\delta + d) \| C_i.\mathbf{p} - C_i.S.\mathbf{p} \|^2 \right)}{\sum_i^{n-1} (C_i.\delta + d)}. \tag{5.32}$$

This cost function is minimized when the cluster representatives are located at the centroids of the clusters. This property is exploited by Lloyd's method [Llo57], a greedy algorithm that solves the *K*-means clustering problem by interleaving one step of centroid computation with one step of reclustering until the clusters converge. I have adapted Lloyd's method to compute contact clusters, as described in Algorithm 5.4.1. Algorithms 5.4.2, 5.4.3, and 5.4.4 describe the complete algorithm in more detail.

$\{S\} \leftarrow$ CLUSTER_CONTACTS$(\{C\}, \{S'\})$

Input: The set of new contacts $\{C_0, C_1, ...C_{n-1}\}$ and the set of old clusters $\{S'_0, S'_1, ..., S'_{l-1}\}$, assuming that the old clusters are ordered according to decreasing δ.
Output: The set of new clusters $\{S_0, S_1, ...S_{m-1}\}$.

Initialize $\{S\}_0 \leftarrow$ INITIALIZE_CLUSTERS$(\{C\}, \{S'\})$
repeat
  $\{C\}_{i+1} \leftarrow$ ASSIGN_CLUSTERS$(\{C\}_i, \{S\}_i)$
  $\{S\}_{i+1} \leftarrow$ COMPUTE_REPRESENTATIVES$(\{C\}_{i+1}, \{S\}_i)$
until $\{S\}_{i+1} = \{S\}_i$
for each cluster $S_j \in \{S\}_i$
  Compute parameters of the representative: $S_j.\delta$, $S_j.\mathbf{n}$, and $S_j.\mathbf{p}_0$
Return $\{S\} \leftarrow (\{S\}_i)$

ALGORITHM 5.4.1: Contact Clustering

Contacts must be clustered at every execution of the contact thread. The clustering information from the previous frame can be used to initialize the iterative process of Lloyd's method. As indicated in Algorithm 5.4.2, the first step of the initialization is to determine the number of output clusters *m*. Then, if *m* is smaller than the number of input clusters, *l*, I drop the input clusters with smallest penetration depth. Next, I initialize the positions of the representatives of $\min(m, l)$ output clusters at the contact points that are closest to the representatives of the remaining input clusters. If *m* is larger than the number of input clusters, I still must initialize the representatives of $m - l$ output clusters. I place these representatives at the contact points that are furthest from the output cluster representatives

$\{S\} \leftarrow$ INITIALIZE_CLUSTERS$(\{C\},\{S'\})$

Input: A set of contacts $\{C_0, C_1, ...C_{n-1}\}$ and the set of old clusters $\{S'_0, S'_1, ..., S'_{l-1}\}$.
Output: A new set of clusters $\{S_0, S_1, ...S_{m-1}\}$ with initial representative positions.

$m = \min(K, n)$
if $l > m$
  Remove clusters with small δ $\{S'_m, ...S'_{l-1}\}$ from $\{S'\}$
for each new cluster S_i s.t. $i < \min(l, m)$ **do**
  Find closest pair $(C_j, S'_k) \leftarrow \min_{C_j \in \{C\}} \min_{S'_k \in \{S\}} \|C_j.\mathbf{p} - S'_k.\mathbf{p}\|$
  Remove C_j from $\{C\}$
  Remove S'_k from $\{S'\}$
  Assign representative $S_i.\mathbf{p} \leftarrow C_j.\mathbf{p}$
  Add S_i to $\{S\}$
for each new cluster S_i s.t. $l \leq i < m$ **do**
  Find furthest contact $C_j \leftarrow \max_{C_j \in \{C\}} \min_{S_k \in \{S\}} \|C_j.\mathbf{p} - S_k.\mathbf{p}\|$
  Remove C_j from $\{C\}$
  Assign representative $S_i.\mathbf{p} \leftarrow C_j.\mathbf{p}$
  Add S_i to $\{S\}$

ALGORITHM 5.4.2: Initialization of Clusters

that are already initialized. Initializing the representatives at contact points ensures that every cluster contains at least one contact.

$\{C\} \leftarrow$ ASSIGN_CLUSTERS$(\{C'\},\{S\})$

Input: The set of contacts $\{C'_0, C'_1, ...C'_{n-1}\}$ and the set of clusters $\{S_0, S_1, ..., S_{m-1}\}$.
Output: The new set of contacts $\{C_0, C_1, ...C_{n-1}\}$ with updated clusters.

Initialize $\{C\} \leftarrow \{C'\}$
for each contact C_i **do**
  Assign cluster $C_i.S \leftarrow \min_{S_j \in \{S\}} \|S_j.\mathbf{p} - C_i.\mathbf{p}\|$

ALGORITHM 5.4.3: Proximity-Based Reclustering

As part of Lloyd's method, the representative of each cluster is recomputed at every iteration as the weighted centroid of all the contact points in the cluster. The expression for the position of the representative is:

$\{S\} \leftarrow$ COMPUTE_REPRESENTATIVES$(\{C\}, \{S'\})$

Input: The set of contacts $\{C_0, C_1, ...C_{n-1}\}$ and the set of clusters $\{S'_0, S'_1, ..., S'_{m-1}\}$.
Output: A new set of clusters $\{S_0, S_1, ...S_{m-1}\}$ with updated representatives.

Initialize $\{S\} \leftarrow \{S'\}$
for each cluster S_i **do**
  Initialize centroid $S_i.\mathbf{p} \leftarrow \mathbf{0}$
  Initialize weight $S_i.w \leftarrow 0$
for each contact C_i **do**
  Add $(C_i.\delta + d)C_i.\mathbf{p}$ to $C_i.S.\mathbf{p}$
  Add $C_i.\delta + d$ to $C_i.S.w$
for each cluster S_i **do**
  Compute representative as weighted average $S_i.\mathbf{p} \leftarrow S_i.\mathbf{p}/S_i.w$

ALGORITHM 5.4.4: Computation of Cluster Representatives

$$S.\mathbf{p} = \frac{\sum_{i,\ C_i.S=S}((C_i.\delta + d)C_i.\mathbf{p})}{\sum_{i,\ C_i.S=S}(C_i.\delta + d)}. \tag{5.33}$$

The contact clusters will be used for computing collision response. Therefore, once the clusters converge, I compute the remaining parameters of the representative contact for each cluster (i.e., $\mathbf{p}_0$, δ, and $\mathbf{n}$), based on the following expressions:

$$S.\delta = \frac{\sum_{i,\ C_i.S=S}((C_i.\delta + d)C_i.\delta)}{\sum_{i,\ C_i.S=S}(C_i.\delta + d)}, \tag{5.34}$$

$$S.\mathbf{n} = \frac{\hat{\mathbf{n}}}{\|\hat{\mathbf{n}}\|},$$

$$\hat{\mathbf{n}} = \frac{\sum_{i,\ C_i.S=S}((C_i.\delta + d)C_i.\mathbf{n})}{\sum_{i,\ C_i.S=S}(C_i.\delta + d)}, \tag{5.35}$$

$$S.\mathbf{p}_0 = S.\mathbf{p} - S.\delta(S.\mathbf{n}). \tag{5.36}$$

5.4.3 Penalty-Based Collision Response

In general, penalty-based collision response refers to the computation of contact forces as a function of object interpenetration [MW88]. I define collision response based on a planar constraint that induces

a viscoelastic force. Before contact clustering, contact normals are defined based on pairs of surface primitives (e.g., edge-edge, vertex-face, etc.). After contact clustering, however, the contact normal **n** is a representative value, but does not capture exact information about surface features. I have opted to model each contact as a planar constraint, as shown in Fig. 5.5-(b). The constraint is represented by the plane with normal **n** and passing through $\mathbf{p}_0$. Note that it is also convenient to represent **p** based on its coordinates in the local frame of the grasped object, **r**. I compute viscoelastic penalty-based force and torque as:

$$\mathbf{F}_p = -kN(\mathbf{x} + R\mathbf{r} - \mathbf{p}_0) - kd\mathbf{n} - bN(\mathbf{v} + \omega \times \mathbf{r}), \tag{5.37}$$

$$\mathbf{T}_p = (R\mathbf{r}) \times \mathbf{F}_p, \tag{5.38}$$

where N is a matrix that projects a vector onto the normal of the constraint plane, and it is computed as $\mathbf{n}\,\mathbf{n}^T$.

As noted in Sec. 5.2.3, implicit integration of rigid body simulation requires a linear approximation of the contact forces. This implies the computation of the Jacobian of penalty force and torque. For the most part, the terms for the torque Jacobian can easily be obtained from the force Jacobian following the differentiation rules for cross products shown in Sec. A.1.3 in the appendix.

As addressed earlier, the contact normal is a representative value resulting from the clustering step. I will assume that the plane constraint remains constant during one frame of simulation. If the contact information were obtained directly from surface primitives, it would be possible to consider the variation of the contact normal as a result of the rotation of the grasped object.

Derivatives w.r.t. Position

The force equation is linear on the position of the center of mass, so the corresponding terms in the Jacobian are easily obtained as:

$$\frac{\partial \mathbf{F}_p}{\partial \mathbf{x}} = -kN, \tag{5.39}$$

$$\frac{\partial \mathbf{T}_p}{\partial \mathbf{x}} = (R\mathbf{r})^* \frac{\partial \mathbf{F}_p}{\partial \mathbf{x}}. \tag{5.40}$$

Derivatives w.r.t. Quaternion

The force equation depends on the orientation through the rotation matrix R, and the torque depends on the orientation in both terms of the cross product. Based on the derivatives of a rotation matrix, given in Eq. A.23 in the appendix, and the derivatives of the angular velocity, given in Eq. 5.13, each column of the force and torque Jacobians can be written as:

$$\frac{\partial \mathbf{F}_p}{\partial q_i} = -kN\frac{\partial R}{\partial q_i}\mathbf{r} - bN\omega^* \frac{\partial R}{\partial q_i}\mathbf{r} + bN(R\mathbf{r})^* \frac{\partial \omega}{\partial q_i}, \tag{5.41}$$

$$\frac{\partial \mathbf{T}_p}{\partial q_i} = (R\mathbf{r})^* \frac{\partial \mathbf{F}_p}{\partial q_i} - \mathbf{F}_p{}^* \frac{\partial R}{\partial q_i}\mathbf{r}. \tag{5.42}$$

Derivatives w.r.t. Linear Momentum

The force equation is linear on the linear momentum, so the corresponding terms in the Jacobian are easily obtained as:

$$\frac{\partial \mathbf{F}_p}{\partial \mathbf{P}} = -\frac{b}{m}N, \tag{5.43}$$

$$\frac{\partial \mathbf{T}_p}{\partial \mathbf{P}} = (R\mathbf{r})^* \frac{\partial \mathbf{F}_p}{\partial \mathbf{P}}. \tag{5.44}$$

Derivatives w.r.t. Angular Momentum

Both the contact force and torque depend on the angular momentum through the angular velocity term in the force equation. Based on Eq. 5.2, which describes the angular velocity as a linear function of the angular momentum, the Jacobians can be written as:

$$\frac{\partial \mathbf{F}_p}{\partial \mathbf{L}} = bN(R\mathbf{r})^* RM^{-1}R^T, \tag{5.45}$$

$$\frac{\partial \mathbf{T}_p}{\partial \mathbf{L}} = (R\mathbf{r})^* \frac{\partial \mathbf{F}_p}{\partial \mathbf{L}}. \tag{5.46}$$

5.4.4 Texture Rendering

In Sec. 4.3, I defined a force model based on directional penetration depth and its gradient that captures the interaction of detailed surface geometry. Here I reintroduce the force model, following a more convenient notation for the formulation of the force and torque Jacobians. Based on the penalty potential U defined in Eq. 4.4, I compute texture force and torque as:

$$\mathbf{F}_t = -\nabla_{\mathbf{x}} U = -\left(\frac{\partial U}{\partial \mathbf{x}}\right)^T, \tag{5.47}$$

$$\mathbf{T}_t = -\nabla_{\theta} U = -\left(\frac{\partial U}{\partial \theta}\right)^T, \tag{5.48}$$

where $\nabla_{\mathbf{x}}$ and ∇_{θ} represent the gradients w.r.t. the position and orientation of the grasped object in the global reference system. Note that $\frac{\partial U}{\partial \mathbf{x}}$ and $\frac{\partial U}{\partial \theta}$ are row vectors, and need to be transposed in order to express them as gradients. As indicated in Sec. 4.3.2, it is convenient to first compute the gradients of the penalty potential U in a rotated reference system defined by the contact normal $\mathbf{n}$ and located at the center of mass of the grasped object. Force and torque are then transformed to the global reference frame. The same transformation can be obtained by differentiating using the chain rule and applying the differentiation rules for Euler angles described in Sec. A.2.7 in the appendix:

$$\mathbf{F}_t = -\left(\frac{\partial U}{\partial \mathbf{x}}\right)^T = -\left(\frac{\partial U}{\partial \mathbf{x}'}\frac{\partial \mathbf{x}'}{\partial \mathbf{x}}\right)^T = -\left(\frac{\partial U}{\partial \mathbf{x}'}R_t\right)^T = -R_t^T \nabla_{\mathbf{x}'} U, \tag{5.49}$$

$$\mathbf{T}_t = -\left(\frac{\partial U}{\partial \theta}\right)^T = -\left(\frac{\partial U}{\partial \theta'}\frac{\partial \theta'}{\partial \theta}\right)^T = -\left(\frac{\partial U}{\partial \theta'}R_t\right)^T = -R_t^T \nabla_{\theta'} U, \tag{5.50}$$

where $\nabla'_{\mathbf{x}}$ and ∇'_{θ} represent the gradients w.r.t. position and orientation in the rotated reference system $\{\mathbf{u}, \mathbf{v}, \mathbf{n}\}$, as defined in Sec. 4.3.2. $R_t = (\mathbf{u}\ \ \mathbf{v}\ \ \mathbf{n})^T$ is the rotation matrix from the global to the rotated reference system.

The texture force and torque are defined purely based on the position and orientation of the grasped

object. As a result, the Jacobians w.r.t. linear and angular momentum are zero. Next, I formulate the Jacobians w.r.t. the position and orientation of the grasped object.

Derivatives w.r.t. Position

I follow the same procedure as for computing the texture force and torque, by applying the chain rule and defining the derivatives of the force and torque in the rotated reference frame. Substituting Eq. 5.49 into the force Jacobian and applying the definition of the Hessian described in Sec. A.1.5 in the appendix, I obtain the equation:

$$\frac{\partial \mathbf{F}_t}{\partial \mathbf{x}} = \frac{\partial \mathbf{F}_t}{\partial \mathbf{x}'}\frac{\partial \mathbf{x}'}{\partial \mathbf{x}} = -R_t^T \mathscr{H}_{\mathbf{x}'} U R_t, \tag{5.51}$$

where $\mathscr{H}_{\mathbf{x}'}U$ is the Hessian of the penalty potential U w.r.t. the position in the rotated reference frame. Similarly, I formulate the torque Jacobian by substituting Eq. 5.50:

$$\frac{\partial \mathbf{T}_t}{\partial \mathbf{x}} = \frac{\partial \mathbf{T}_t}{\partial \mathbf{x}'}\frac{\partial \mathbf{x}'}{\partial \mathbf{x}} = -R_t^T \frac{\partial \left(\nabla_{\theta'} U\right)}{\partial \mathbf{x}'} R_t. \tag{5.52}$$

Derivatives w.r.t. Quaternion

In order to formulate the Jacobian terms w.r.t. the quaternion, I first define the Jacobians w.r.t. Euler angles in the global reference frame, and then apply the chain rule. Substituting Eq. 5.49, the force Jacobian can be expressed as:

$$\frac{\partial \mathbf{F}_t}{\partial \mathbf{q}} = \frac{\partial \mathbf{F}_t}{\partial \theta'}\frac{\partial \theta'}{\partial \theta}\frac{\partial \theta}{\partial \mathbf{q}} = -R_t^T \frac{\partial \left(\nabla_{\mathbf{x}'} U\right)}{\partial \theta'} R_t \frac{\partial \theta}{\partial \mathbf{q}}. \tag{5.53}$$

The derivation of the Jacobian of Euler angles w.r.t. the quaternion is given in Sec. A.2.8 in the appendix. The torque Jacobian can be obtained similarly, substituting Eq. 5.50 and applying the definition of the Hessian:

$$\frac{\partial \mathbf{T}_t}{\partial \mathbf{q}} = \frac{\partial \mathbf{T}_t}{\partial \theta'}\frac{\partial \theta'}{\partial \theta}\frac{\partial \theta}{\partial \mathbf{q}} = -R_t^T \mathscr{H}_{\theta'} U R_t \frac{\partial \theta}{\partial \mathbf{q}}. \tag{5.54}$$

Hessian of Texture Penalty Potential

The texture force model presented in Sec. 4.3.2 defines force and torque equations based on the gradient of a penalty potential function. As expected, the formulation of the Jacobian of the force and torque requires the computation of the Hessian of the penalty potential. Given a potential function U, the Hessian $\mathscr{H}U$ in the rotated reference frame of the contact, $(\mathbf{x}', \theta')$, is defined as:

$$\mathscr{H}U = \begin{pmatrix} \mathscr{H}_{\mathbf{x}'}U & \frac{\partial(\nabla_{\mathbf{x}'}U)}{\partial\theta'} \\ \\ \frac{\partial\left(\nabla_{\theta'}U\right)}{\partial\mathbf{x}'} & \mathscr{H}_{\theta'}U \end{pmatrix}. \tag{5.55}$$

Given a penalty potential that grows quadratically with the penetration depth δ, as defined in Eq. 4.4, each term of the Hessian is defined as:

$$\frac{\partial^2 U}{\partial x_i \partial x_j} = k\delta\frac{\partial^2\delta}{\partial x_i\partial x_j} + k\frac{\partial\delta}{\partial x_i}\frac{\partial\delta}{\partial x_j}. \tag{5.56}$$

The penetration depth δ is approximated by the directional penetration depth along the contact normal, $\delta_{\mathbf{n}}$. As described in Sec. 4.4, I compute $\delta_{\mathbf{n}}$ following an image-space algorithm, implemented on graphics processors. Due to the discretization of the penetration depth, I propose a discrete approximation of the Hessian based on central differencing. The different terms of the Hessian can be computed as:

$$\frac{\partial^2\delta}{\partial x_i\partial x_j} = \frac{\delta(x_i+\Delta x_i, x_j+\Delta x_j) - \delta(x_i-\Delta x_i, x_j+\Delta x_j) - \delta(x_i+\Delta x_i, x_j-\Delta x_j) + \delta(x_i-\Delta x_i, x_j-\Delta x_j)}{4\Delta x_i\Delta x_j}, \tag{5.57}$$

$$\frac{\partial^2\delta}{\partial x_i^2} = \frac{\delta(x_i+\Delta x_i) - 2\delta(x_i) + \delta(x_i-\Delta x_i)}{\Delta x_i^2}. \tag{5.58}$$

5.5 Experiments and Results

I begin this section with the discussion of implementation details. Then, I present and discuss the results of the experiments I have carried out to test the responsiveness and stability of the 6-DoF haptic rendering system. I have tested the behavior of the system during free-space motion and during contact state, analyzing the influence and the choice of parameters of its different components. Specifically, I focus on 4 experiments: free-space motion of a thin object (i.e., a spoon), contact between relatively simple polygonal models (i.e., a spoon and a cup), contact between complex polygonal models (i.e., virtual jaws), and exploration of a textured surface with a probe.

5.5.1 Implementation Details

The experiments have been performed using a 6-DoF *Phantom*^{TM} haptic device, a dual Pentium-4 2.4GHz processor PC with 2.0 GB of memory and an NVidia GeForce FX5950 graphics card, and Windows2000 OS. I have used the CLAPACK library for solving the linear system of semi-implicit integration (See Sec. 5.2.2). On the PC used for the experiments, the processor spends approximately 55μs per frame formulating and solving the linear system. The haptic thread is executed at a constant frequency of 1kHz, and it employs utilities of GHOST-SDK, the software API of the *Phantom*^{TM} haptic device, to communicate with the device controller. The contact thread is executed asynchronously and is assigned a lower scheduling priority.

In the experiments with the cup and the spoon and the textured plate I have used the libraries SWIFT++ [EL01] and DEEP [KLM02] for collision detection. In the experiment with the virtual jaws I have used my multiresolution collision detection algorithm, *contact levels of detail* (CLODs), which employs routines from SWIFT++ and DEEP for solving distance and penetration queries between pairs of convex primitives. In the experiment of the textured plate, I have incorporated the image-based algorithm for computing directional penetration depth described in Sec. 4.4. For more details on the GPU-based implementation of the penetration depth computation, please refer to Sec. 4.5.2. For contact clustering, I have typically selected $K = 5$ as the number of clusters.

The selection of the contact stiffness value k is associated with each particular experiment, as it depends on parameters such as the virtual mass and inertia of the grasped object, or the maximum

number of contacts. Typically, I have been able to achieve stable behavior of the penalty-based rigid body simulation with contact stiffness values as high as 2kN/m or 5kN/m, with mass values as low as 10g. The maximum steady force of the 6-DoF *Phantom*^{TM} haptic device is 1.4N, and with that force value and the aforementioned stiffness values object interpenetration can remain under 1mm. The selection of the coupling stiffness k_c depends on two factors: the numerical stability of the rigid body simulation and the stability of the force feedback. In practice, the dominant factor has been the stability of the force feedback. Typically, I have used values between 200N/m and 500N/m. For the selection of the rotational coupling stiffness k_θ, however, the dominant factor has been the numerical stability of the simulation, specially with light, thin objects (such as the model of the spoon). I have used values of k_θ between 0.6Nm/rad and 3Nm/rad.

5.5.2 Analysis of Free-Space Motion

As indicated previously in Sec. 1.2.2, responsive free-space interaction is one in which the grasped object closely follows the motion command of the haptic device and the user perceives a low mechanical impedance. Using virtual coupling, the coupling stiffness k_c must be high so that the grasped object closely follows the haptic device. However, a high coupling stiffness imposes stability challenges on the numerical integration of rigid body simulation, specially if the mass of the grasped object is small.

I have designed an experiment to evaluate the performance of implicit integration for rigid body simulation during free-space motion with virtual coupling. In the experiment, the haptic device commands the motion of a 20cm-long spoon (see Fig. 5.7). The spoon is moved freely, without touching other objects. A thin object, such as a spoon, is particularly challenging for numerical integration due to its low inertia around its longitudinal axis.

Fig. 5.6 reflects the coupling deviation, $\|\mathbf{x}_h - \mathbf{x}_c\|$, and the absolute value of coupling force, $\|\mathbf{F}_c\|$, during 2.5sec. of simulation. I have collected the values of coupling deviation and force using different numerical integration methods (i.e., Forward Euler, Runge-Kutta IV, and Backward Euler) and the same pre-recorded trajectory of the haptic device. Coupling force and torque are exerted on the grasped object, following the formulation described in Sec. 5.3, but force-feedback to the user is disabled. Using the Backward Euler implicit integration method, with coupling stiffness $k_c = 200$N/m and $k_\theta = 0.6$Nm/rad, the simulation is stable with a mass as small as 1g. However, using explicit

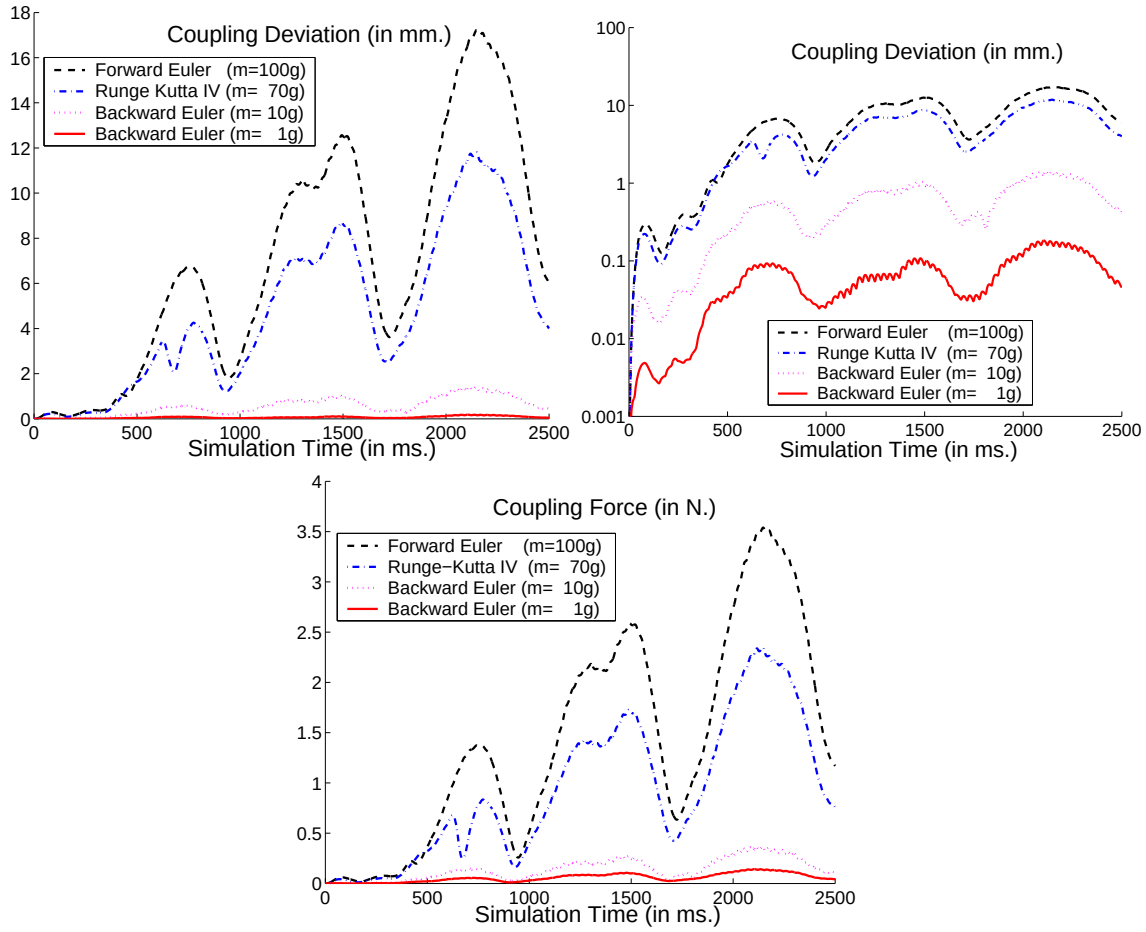

Figure 5.6: **Coupling Deviation and Force During Free-Space Motion.** *Comparison of coupling deviation and force using different numerical integration methods, and varying the mass of the grasped object. Top left: deviation between the position of the haptic device and the position of the coupling point in the spoon; Top right: log plot of the coupling deviation; Bottom: coupling force.*

integration methods, such as Runge-Kutta IV or Forward Euler, the simulation is stable only with masses larger than 70g and 100g respectively.

The top left graph of Fig. 5.6 shows the coupling deviation, which reaches 17mm with a mass of 100g, but it never exceeds 2mm with a mass of 10g. The logarithmic plot in Fig. 5.6 indicates that the coupling deviation is roughly linear w.r.t. the mass of the spoon. The bottom graph of Fig. 5.6 shows the coupling force, which reaches 3.5N with a mass of 100g, but it never exceeds 0.5N with a mass of 10g. The results of the experiment indicate that, with constant coupling stiffness, the coupling deviation is larger when the mass of the grasped object is larger. Similarly, the coupling force is also larger when the mass is larger. From these two observations, and considering that stable mass values

are substantially larger with explicit integration, I conclude that implicit integration enables more responsive free-space interaction with virtual coupling than explicit integration.

5.5.3 Analysis of Contact State

I have also analyzed the behavior of the 6-DoF haptic rendering system during contact state. A scenario with relatively simple models (i.e., the cup and the spoon depicted in Fig. 5.7) has been used to test the performance of the haptic rendering approach, integrating implicit rigid body simulation, penalty-based methods, and virtual coupling. And a scenario with complex models (i.e., the jaws depicted in Figs. 3.12 and 5.9) has been used to test the use of the linearized contact model in the multirate architecture and the integration of multiresolution collision detection using CLODs with the rest of the system.

Contact between a Spoon and a Cup

I have recorded a trajectory of the haptic device while manipulating a virtual spoon (1,344 triangles and 20cm-long) in contact with a virtual cup (4,000 triangles and 8cm-radius). Then, I have played this trajectory using different haptic rendering settings. I have analyzed the stability and responsiveness of the system with different contact stiffness values and with different integration methods. Fig. 5.8 shows graphs of maximum local penetration depth (top left), coupling deviation (top right), contact force (bottom left), and feedback or coupling force (bottom right) during 650ms. of simulation and the following settings: (1) Runge-Kutta IV, $m = 100$g, and $k = 2$kN/m; (2) Backward Euler, $m = 10$g, and $k = 2$kN/m; and (3) Backward Euler, $m = 100$g, and $k = 10$kN/m. The coupling stiffness is 200N/m in all three cases.

As can be inferred from the graph of penetration depth in Fig. 5.8, the spoon moves in free-space for a period of more than 100ms., and then starts penetrating the surface of the cup. The spoon remains in contact with the cup (penetrating slightly) during the rest of the simulation. Penalty forces act on the spoon while contact persists. These forces constrain the motion of the spoon, and the deviation w.r.t. to the command position of the haptic device increases. This deviation produces a coupling force that is fed back to the user, resulting in kinesthetic perception of contact.

Numerical integration of the motion of the spoon with the Runge-Kutta IV method is stable for

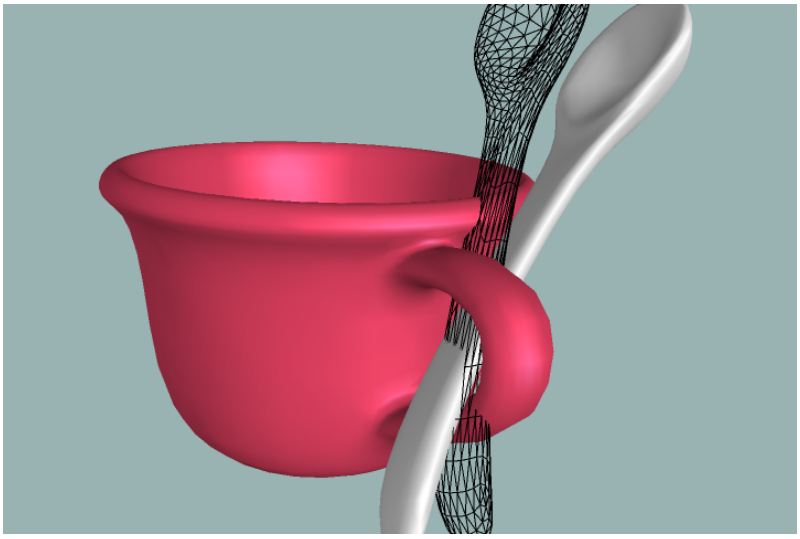

Figure 5.7: **Manipulation of a Spoon in Contact with a Cup Using Virtual Coupling.** *As the spoon is constrained inside the handle of the cup, the contact force and torque are perceived through a virtual coupling. A wireframe image of the spoon represents the actual configuration of the haptic device.*

values of the mass larger than 70g, as concluded from the analysis of free-space motion. This requirement affects the performance during contact state as well. As reflected in the bottom right graph of fig. 5.8, with a mass of 100g the magnitude of feedback force during free-space motion and contact situations is similar. This similarity degrades the kinesthetic perception of contact. Implicit integration is stable for small values of the mass, which enable a clearer distinction in the magnitude of feedback force between free-space motion and contact state.

High contact stiffness minimizes the amount of interpenetration between the spoon and the cup. As shown in the top left graph of Fig. 5.8, the maximum local penetration during the interval of study was smaller than 0.6mm with a contact stiffness of 2kN/m, and smaller than 0.2mm with a contact stiffness of 10kN/m. The maximum coupling force (i.e., the feedback force) during the same interval is 0.8N. Given a saturation value of the coupling force of 4N, the penetration depth could grow up to 3mm if the user pressed the spoon hard against the surface of the cup. A penetration depth of 3mm represents less than 4% of the radius of the cup, and it can be avoided by setting a small tolerance for collision detection and response, as suggested in Sec. 5.4.1. As a conclusion, penalty-based collision response with high contact stiffness enables small visual interpenetrations, which can enhance the perception of hard contact.

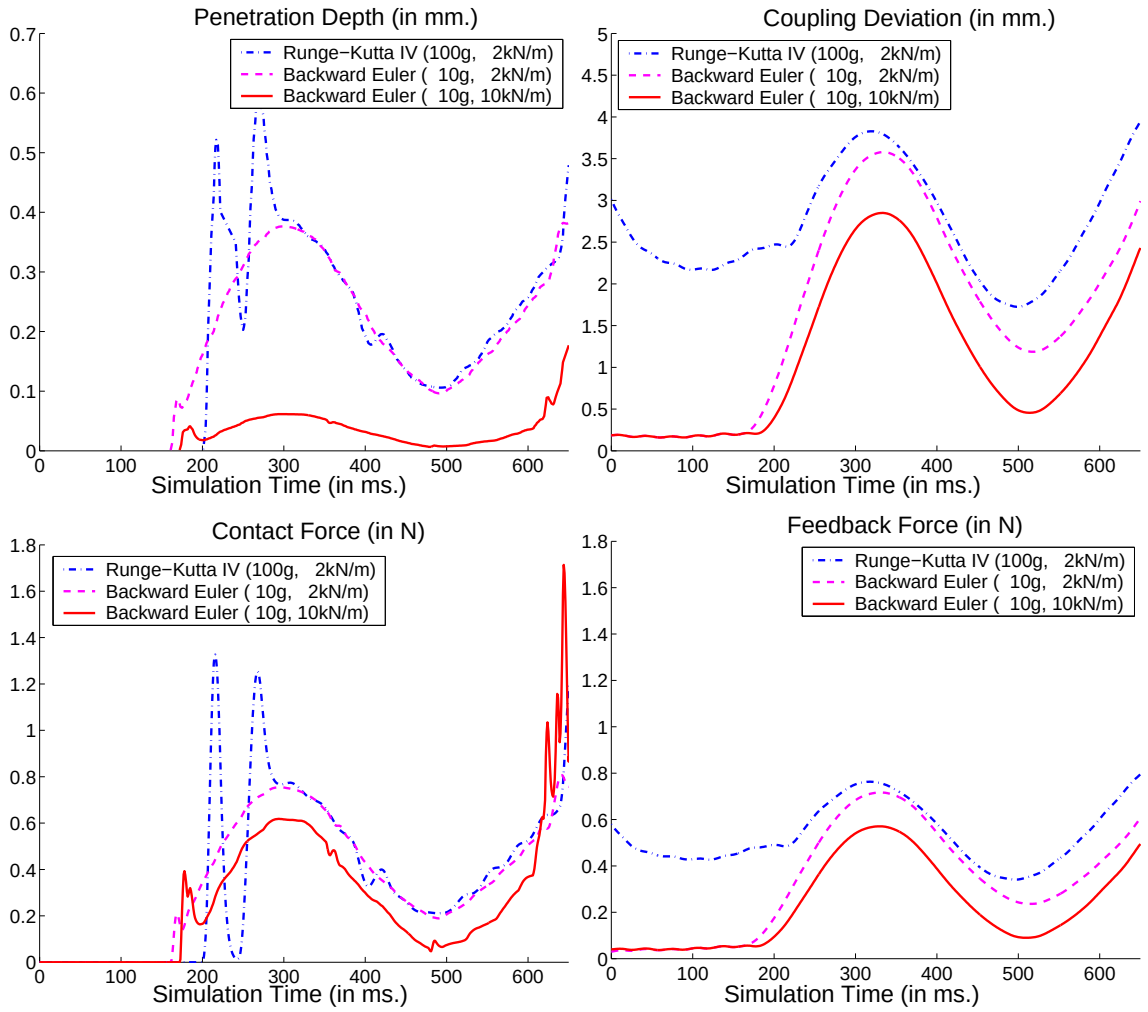

Figure 5.8: **Analysis of Forces and Positions During Contact.** *Comparison of maximum local penetration depth (top left), coupling deviation (top right), contact force (bottom left), and feedback or coupling force (bottom right) using different numerical integration methods and contact stiffness values.*

The numerical integration of the motion of the spoon is susceptible to instability problems with high contact stiffness. Contact clustering alleviates the discontinuities of contact-point positions, but (smaller) discontinuities are still present, and they may induce large oscillations of the contact force and the penetration depth, as shown in the left graphs of Fig. 5.8. Note the existence of oscillations with Runge-Kutta IV and $k = 2$kN/m, and with Backward Euler and $k = 10$kN/m. Out of the interval of study, the oscillations with these settings became more serious, and were also transmitted to the coupling force. However, with Backward Euler and $k = 2$kN/m, the numerical integration of the motion of the spoon remained stable. Moreover, note that, with these settings, the contact force and

the coupling force almost coincide. Implicit integration methods enable stable penalty-based rigid body simulation with (relatively) high contact stiffness and small mass values.

Contact between Virtual Jaws

I have tested the 6-DoF haptic rendering algorithm on a benchmark consisting of complex virtual jaws (See Figs. 3.12 and 5.9). The model of the lower jaw is composed of $40,180$ triangles, while the upper jaw contains $47,339$ triangles (See Table 3.2 for more statistics of the models). Interactive 6-DoF haptic rendering of such complex models is possible only by using CLODs, as demonstrated in Sec. 3.5. This benchmark has been used to validate the multirate architecture described in Sec. 5.1.2 and the use of a linearized contact model.

I have recorded a trajectory of the upper jaw while rendering the interaction with the lower jaw and using CLODs with an error threshold of 2.5% of the radius of the jaws. Then, I have played this same trajectory with smaller error thresholds of 1% and 0.4%, thereby increasing the cost of collision detection and decreasing the update rate of the contact thread. I have run the experiments with and without the use of the linearized contact model. By using the linearized contact model, contact forces are approximated every frame of the haptic thread based on their Jacobian. Without the linearized contact model, the update rate of contact forces is limited by the cost of collision detection. In the experiment without linearized contact model and with an error threshold of 0.4%, the simulation soon becomes unstable to the point that the state of the upper jaw diverges to infinity. For clarity of the graphs, I have not included the data of this experiment.

Fig. 5.10 shows graphs of maximum local penetration depth (top left), frame rate of the contact thread (top right), coupling deviation (center), and feedback or coupling force (bottom) during 900ms. of simulation, using different error tolerances for CLODs, with and without (w/o) linearized contact model. The models of both jaws can be bounded by spheres of 6cm-radius. I have scaled the workspace of the haptic device by a factor of 0.4, therefore the forces plotted in the graphs are scaled by a factor of 2.5 before being fed back to the user. All the experiments have been executed using Backward Euler semi-implicit integration as described in Sec. 5.2.2, a mass $m = 10$g for the upper jaw, coupling stiffness $k_c = 500$N/m, and contact stiffness $k = 5$kN/m. I have applied a low-pass filter with a cut-off frequency of 300Hz to the data of the period of the contact thread, in order to remove

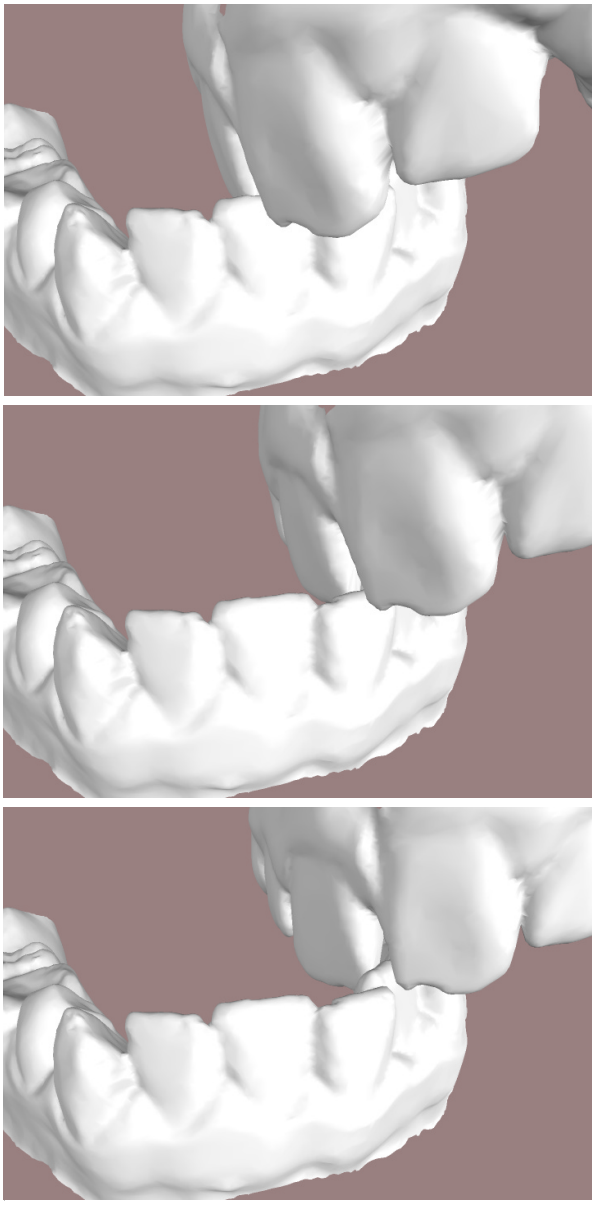

Figure 5.9: **Dexterous Interaction of Virtual Jaws.** *Three snapshots of an upper jaw being moved over a lower jaw, with intricate teeth interaction.*

very high frequency noise caused by thread scheduling.

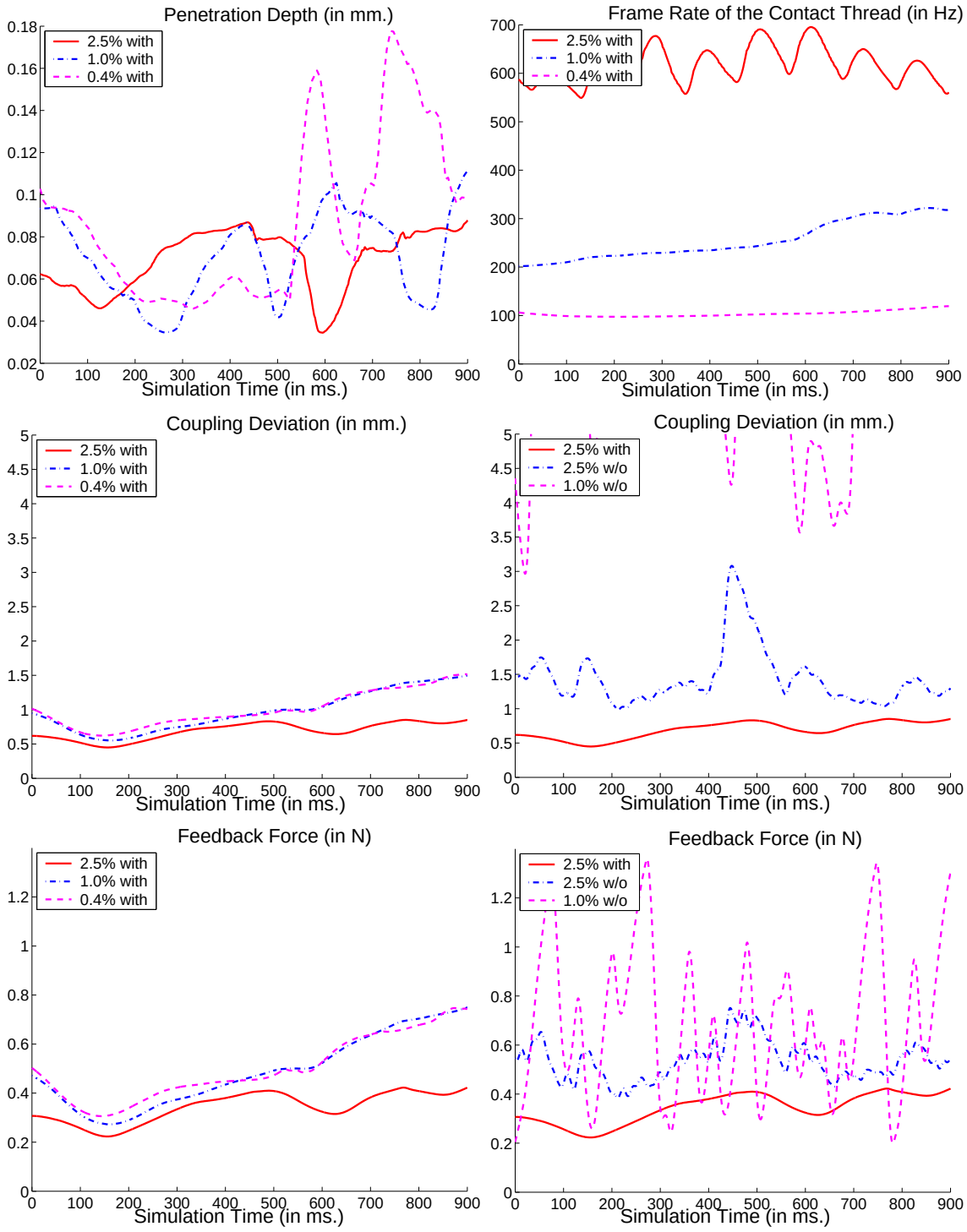

Figure 5.10: **Analysis of the Linearized Contact Model.** *Comparison of maximum local penetration depth (top left), frame rate of the contact thread (top right), coupling deviation (center), and feedback or coupling force (bottom) using different error tolerances for CLODs, with and without (w/o) linearized contact model.*

The plots demonstrate that, with the linearized contact model and an error threshold of 2.5% the behavior of the system becomes stable and responsive. For example, the maximum local penetration depth never exceeds 0.1mm, thanks to high stability with a contact stiffness as high as 5kN/m. With the linearized contact model but reducing the error threshold, the behavior degrades slightly, but remains considerably stable. With an error threshold of 0.4% the frame rate of the contact thread goes down to 100Hz. Even in such a challenging situation, the computation of approximate contact forces with the linearized contact model maintains high stability.

On the other hand, without the linearized contact model, the performance degrades rapidly. Even with an error threshold of 2.5%, which keeps the frame rate of the contact thread over 500Hz, the feedback force becomes clearly unstable. The comparison of simulation data with and without the linearized contact model clearly indicates the influence of the linearized contact model on the stability of the system when the update rate of the contact thread decays. This observation demonstrates that the linearized contact model is a key factor for the success of 6-DoF haptic rendering of complex models.

5.5.4 Haptic Rendering of Textures

In Sec. 5.4.4, I have described the integration of the haptic texture rendering algorithm described in Chapter 4 with the complete system for 6-DoF haptic rendering. As will be discussed later in Sec. 5.6.2, the discrete computation of derivatives imposes serious limitations on the application of implicit integration to texture forces. Nevertheless, I have successfully tested the complete system on moderately complex textured models.

Fig. 5.11 shows a 10cm-long cylindrical probe with a spherical tip of 1cm-radius exploring a 35cm$\times$35cm plate with 8mm-high sinusoidal ridges. The low-resolution models used for collision detection are shown on the left, and the high-resolution models represented with *haptic textures* are shown on the right. I have rendered the haptic interaction with contact stiffness $k = 2$kN/m, linear coupling stiffness $k_c = 200$N/m, angular coupling stiffness $k_\theta = 3$Nm/rad, and a mass $m = 10$g for the probe.

I have compared the feedback forces produced by the interaction with the textured plate and with the flat, low-resolution plate. Fig. 5.12 shows the plots of penetration depth and feedback force for

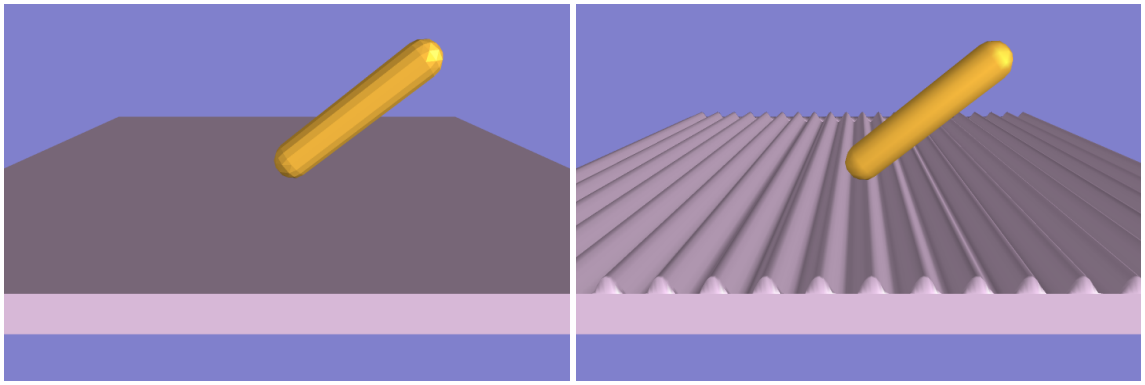

Figure 5.11: **Haptic Exploration of a Plate with a Probe.** *Left: exploration of a flat, low-resolution plate; Right: exploration of a textured plate, modeled as a low-resolution plate with a haptic texture.*

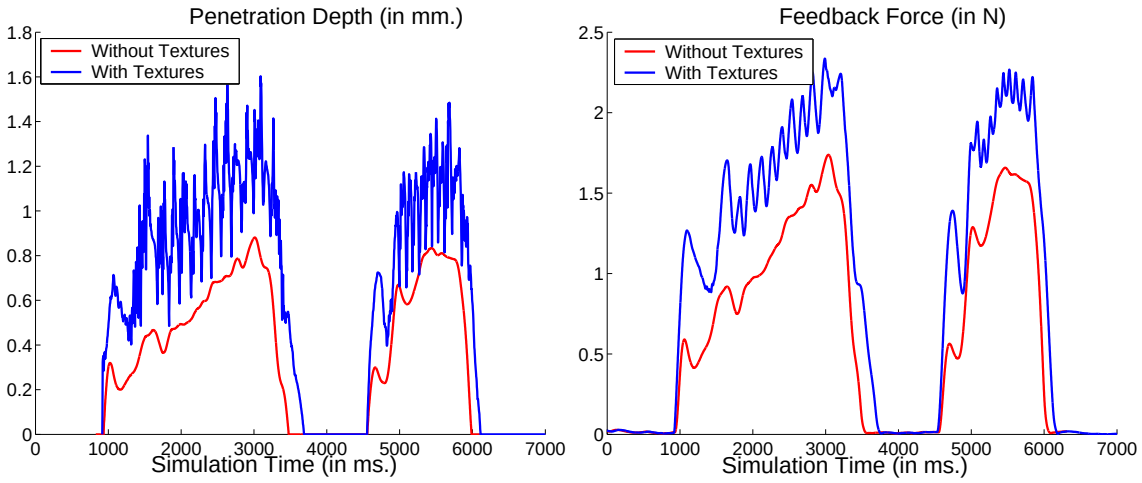

Figure 5.12: **Analysis of Texture Forces.** *Comparison of penetration depth (left) and feedback force (right) between haptic exploration of a flat plate (in red) and a textured plate (in blue). The textured plate is represented by the flat plate and a haptic texture for the computation of contact information.*

textured and flat plates during 7sec. of interaction. The trajectory of the probe was the same for both plates. The probe was first moved from left to right, and then from right to left at a higher speed. Both plots of penetration depth and feedback force clearly reflect oscillations induced by the ridges of the textured plate, which are not present in the case of the flat plate. These oscillations produce vibratory motion of the user's hand, that leads to perception of texture. The frame rate of the contact thread was only slightly higher than 400Hz for the interaction with the textured plate, but the graph of feedback force denotes a stable interaction.

The graph of penetration depth for the textured plate presents noticeable noise, caused by resolu-

tion limitations associated with the image-based computation of penetration depth. These limitations are discussed in more detail in Sec. 5.6.2.

5.6 Summary and Limitations

In this chapter I have presented a stable and responsive algorithm for 6-DoF haptic rendering using implicit integration methods. The key ideas for the success of the algorithm are: (1) decoupling the computation into several components, (2) the use of fast but accurate approximations, and (3) high update rates for each component.

Six-DoF haptic rendering comprises three main problems: synthesis of force feedback, computation of the motion of the grasped object, and computation of collision response. I have designed a system that solves each problem separately. I use the well-known technique of virtual coupling [CSB95, AH98a] for synthesizing bidirectional interaction between the user and the grasped object; I use implicit integration to perform rigid body simulation; and I use a novel multiresolution collision detection algorithm, CLODs, and a novel haptic texture rendering algorithm in conjunction with penalty-based methods for collision response.

Implicit integration facilitates the use of high stiffness and low mass values in the simulation, thereby enabling high responsiveness and stability, as demonstrated in several experiments described in this chapter. Implicit integration methods achieve high stability by estimating the derivatives of state variables. In this chapter I have presented the mathematical formulation for incorporating penalty-based contact forces, coupling forces, and texture-induced forces into implicit integration of rigid body simulation. Moreover, I have proposed a linearized contact model that approximates the values of contact forces using the same Jacobians w.r.t. state variables that are used for implicit integration. I have performed experiments that demonstrate the benefits of the linearized contact model for achieving stable and responsive interaction with complex models.

High force update rates enable high coupling stiffness, and thereby very responsive interaction. I have presented a multirate architecture that ensures a fast update of the force-and-torque feedback values, as well as a fast update of the motion of the grasped object. With the haptic device selected for the experiments, this update takes place at 1kHz. The effectiveness of the linearized contact model

is based on the assumption that contact forces can be well-approximated by linear functions. This assumption holds only if contact forces are updated fast enough. Multiresolution collision detection using CLODs provides a fast, perceptually-driven update of contact information for complex polygonal models.

The 6-DoF haptic rendering algorithm presented in this chapter enables stable and responsive dexterous interaction with complex models, as demonstrated in the experiments, but it also presents several limitations. Next I discuss limitations associated with penalty-based collision response and limitations for haptic texture rendering.

5.6.1 Limitations of Penalty-Based Methods

The use of penalty-based collision response involves two major problems:

- **Passing through objects.** If the grasped object penetrates deeply inside an object in the scene, the penetration direction may suddenly change. This change produces a penalty force that pushes the grasped object in a direction different from the one in which it penetrated. The implementation of CLODs based on surface convex decomposition uses the publicly available library DEEP [KLM02] for computing penetration depth between convex portions of the objects. With this implementation, deep interpenetrations are not always correctly quantified, thus increasing the chances of passing through objects in the scene. As described in Sec. 2.4, ideally one would impose non-penetration constraints explicitly, and solve analytically for the constraint forces. To the best of my knowledge, there is no practical approach that integrates the advantages of constraint-based and penalty-based methods for enforcing non-penetration constraints with fixed-time-step integration.

- **Geometry-driven instabilities.** As described in Sec. 5.2.2, semi-implicit integration employs linear approximations of the time-derivatives of state variables. Linear functions, however, do not approximate these derivatives well when contact discontinuities take place, resulting in numerical instabilities, as demonstrated in the experiments. I have proposed a contact clustering approach in order to alleviate the discontinuities of contact information, and it has proved to improve stability, but discontinuities are still present. As presented in Sec. 5.4.1, I define

penalty collision response based on point-on-plane contact. A possible solution to the problem of geometry-driven instabilities would be to generalize the definition of contacts, capturing situations such as plane-on-plane contact. However, this solution could add a substantial cost to collision detection, in order to find faces that were close to parallel.

5.6.2 Limitations for Haptic Texture Rendering

During the interaction between textured surfaces, the implicit integration of the motion of the grasped object and the formulation of the linearized contact model require the computation of derivatives of texture forces w.r.t. the state variables. Texture force and torque are themselves defined as derivatives of penetration depth, therefore, as described in Sec. 5.4.4, the derivatives of texture forces are based on the Hessian of penetration depth. In Chapter 4, I proposed an image-based algorithm for computing penetration depth and its gradient. The gradient is approximated by computing each partial derivative using central differences. As discussed in Sec. 4.6.2, discrete approximation of the derivatives is subject to aliasing problems, but the experiments described in Chapter 4 proved that a 50×50 grid was fine enough for providing accurate results at interactive rates. However, in general the same grid resolution was insufficient for obtaining stable behavior with the formulation presented in this chapter, specially for the complex benchmarks discussed in Sec. 4.5.2.

I presume the existence of at least three reasons why the selected resolution was sufficient in the experiments of Chapter 4, but insufficient for integrating the haptic texture rendering algorithm with the implicit integration of rigid body simulation. The first reason is that the computation of second derivatives amplifies high frequencies even more than the computation of first derivatives, and it is thus more susceptible to aliasing problems. The second reason is that discrete approximation of the Hessian of a function in $\mathbb{R}^n$ requires $O(n^2)$ evaluations of the function, while the gradient requires only $O(n)$ evaluations. This difference reduces considerably the update rate of the contact thread. The third reason is that the contact stiffness was under 100N/m in the experiments described in Chapter 4, considerably lower than the values of several kN/m used in the experiments described in this chapter. Higher contact stiffness has the advantage of increasing responsiveness in contact tasks, but it has the disadvantage of increasing the sensitivity to inaccurate penetration depth values as well.

The processing capability of GPUs grows at rates higher than Moore's Law. Consequently, the

complexity of the textured surfaces that can be handled in a stable manner will increase rapidly as well. Nevertheless, I also consider other directions for solving the limitations of haptic texture rendering, such as the application of concepts from differential geometry to the formulation of the derivatives of penetration depth. Surface tangents, normals, and curvature information can be stored directly in texture images, and this approach would not require discrete approximations of the derivatives of penetration depth.

Chapter 6

Conclusion

In computer graphics, many researchers have investigated the interactive visual rendering of increasingly complex objects and scenes over the years. Nowadays virtual environments present synthetic images of objects with complex shapes, highly textured surfaces and rich lighting effects. Users of virtual environments would probably like to be able to touch the virtual objects and interact with them, but the interfaces and computational techniques that can make it possible are still fairly rudimentary. Haptic rendering techniques have targeted mostly the problem of tracing objects with a point (i.e., 3-DoF haptic rendering), and important advances over the last 10 years have enabled interactive 3-DoF haptic display of fairly complex surfaces [WS03]. However, few researchers have tackled the problem of synthesizing force and torque feedback resulting from object-object interaction (i.e., 6-DoF haptic rendering). Earlier techniques for 6-DoF haptic rendering [MPT99, NJC99, GME+00, KOLM03, JW03, WM03, JW04] were limited to fairly simple models or contact configurations, mostly due to their dependency on the sampling of the models and the inherent cost of contact determination.

In this dissertation I have presented techniques that attempt to overcome the high cost of contact determination between complex models and the high performance constraints of force feedback by exploiting multiresolution representations, perceptually-driven simplifications, and fast and stable approximations. The integration of these novel techniques has successfully been demonstrated by achieving stable and responsive 6-DoF haptic rendering of complex polygonal models.

In spite of the advances I have presented in this dissertation, there are still many open problems in 6-DoF haptic rendering. The techniques I have developed are based on assumptions about the nature and the behavior of the objects that do not always hold. Also, these techniques have some performance

limitations.

In this chapter I summarize the main results of my dissertation. I also discuss possible future research directions for overcoming the limitations of my current approach, and for extending the techniques I have developed to new research problems and applications.

6.1 Summary of Results

I have adapted two techniques that have proved great success in computer graphics for the rendering of complex models, levels of detail and texture mapping, to 6-DoF haptic rendering. The synthesis of images requires the simulation of the interaction between light and objects. However, the synthesis of force and torque feedback requires the simulation of the interaction between objects. Due to this inherent difference, it is not trivial to adapt level-of-detail techniques or texture mapping to the algorithms for haptic rendering.

In this dissertation I have presented *contact levels of detail* (CLODs), a multiresolution collision detection algorithm that integrates level-of-detail techniques with efficient data structures for hierarchical collision culling. And I have also presented a 6-DoF haptic texture rendering algorithm in which objects are described as low-resolution geometric representations with *haptic textures* that encode fine geometric detail. CLODs incorporate level-of-detail techniques into collision detection algorithms, and they enable an adaptive selection of object resolution at each contact independently. My haptic texture rendering algorithm first computes approximate contact information between low-resolution models, and then refines this information using the geometric detail stored in haptic textures.

The use of multiresolution representations implies the need for error metrics in order to adaptively select appropriate object resolutions. Researchers in psychophysics have investigated the factors involved in the haptic perception of surface features and roughness [KL95, OC01, KL02], and I have built on their observations to design fast, yet perceptually accurate force models and geometric approximations. Feature detection is influenced by the relationship between contact area, object resolution, and size of surface features, and I have exploited this relationship in order to guide the creation and run-time selection of CLODs. Roughness perception is influenced by the vibratory motion induced by geometric interaction, applied force, and exploratory speed, and I have accounted for these factors

in the design of a texture force model based on the gradient of local penetration depth.

I have incorporated my collision detection and response techniques into a 6-DoF haptic rendering algorithm based on implicit integration for rigid body dynamic simulation. I achieve high stability and responsiveness by decomposing the haptic rendering pipeline into modular components and maximizing the update rate of each component. The experiments I have performed on complex polygonal models demonstrate that, as stated in my thesis, the combination of efficient multiresolution data structures and collision detection algorithms with perceptually inspired force models and simplification techniques enables stable and responsive 6-DoF haptic rendering.

Due to the rapid increase of the computational power of commodity processors, 6-DoF haptic rendering of *complex* models will eventually be possible using constraint-based rigid body simulation techniques and full-resolution models. However, multiresolution techniques, such as the ones I have proposed in my dissertation, will still be applicable. Visual level-of-detail rendering techniques became popular in the early ’90s, but more than 10 years later they are still the focus of important research in computer graphics. The reason is that the complexity of the models and the scenes that people want to display has not stopped increasing. I believe that the same trend will hold in haptic rendering. Texture mapping and multiresolution or level-of-detail techniques will persist as enabling tools for haptic rendering of complex scenarios.

The techniques I have presented in this dissertation have been applied almost exclusively to the problem of 6-DoF haptic rendering, and only on a small set of benchmarks. I believe that many of the advances presented in this dissertation can be applied in other areas of interactive simulation and on models with diverse behaviors and representations. In the next section I describe some possible extensions to my work.

6.2 Future Work

The results I have presented in this dissertation suggest many exciting research directions. The limitations of my approach are partly due to assumptions that do not always hold and partly due to implementations I have selected. Among the assumptions, here I focus the discussion on the ones that established the scope of my dissertation, concerning the description and behavior of the models

involved in the simulation. Further research along the lines of my techniques will benefit greatly from their application in practical problems and from user studies. Nevertheless, I believe that some of the results I presented in this dissertation are not restricted to haptic rendering, and they can have an impact on the more general context of simulation of object-object interaction.

6.2.1 Limitations of the Current Techniques

In previous chapters, I have already discussed several limitations of the techniques I have developed (See Secs. 3.6, 4.6, and 5.6). Here I list those limitations again:

- A CLOD at a certain level in the bounding volume hierarchy (BVH) may not bound higher-resolution CLODs or the full-resolution surface.
- The highest levels of the BVH of CLODs are not obtained through simplification operations, and cannot be considered part of the multiresolution representation.
- CLODs are static LODs, and contact information may not vary smoothly across levels.
- For haptic texture rendering, the surface of an object at full-resolution may not constitute a height field in the contact patch.
- Interactive haptic texture rendering of complex textured surfaces is not yet possible, due to the high resolution required in the discrete approximation of the derivatives of penetration depth.
- A high gradient of penetration depth produces high contact stiffness that can induce instabilities.
- Deep interpenetrations, and even passing with the grasped object through other objects, may occur due to the lack of non-penetration constraints with penalty-based methods.
- The output of collision queries may exhibit geometric discontinuities in the contacts.
- The formulation of contact forces ignores friction effects at the moment.

In this list I have not included intrinsic approximation errors associated with the algorithms I have designed. Some of these approximation errors are the surface deviation introduced by CLODs, the distortion introduced by texture mapping, the errors induced by image-based computation of penetration depth, and the error induced by semi-implicit integration techniques.

6.2.2 Relaxation of Initial Assumptions

In the introduction of my dissertation I posed several assumptions concerning the nature and behavior of the objects involved in the simulation. The techniques that I have developed currently enable the interaction with rigid, static objects described by triangle meshes and, in general, they cannot be applied directly to objects described with different representations, to deformable objects, or to multiple moving objects. A complete 6-DoF haptic rendering algorithm should enable the interaction with moving and deformable objects. Next, I briefly discuss possible extensions of my techniques to more general 6-DoF haptic rendering.

Representation of the Models

The data structure for CLODs assumes that models are described as triangle meshes. However, many of the techniques I have developed do not rely on this assumption, and they can be applied to different model representations provided that the appropriate data structures are defined. For instance, the perceptual observations that drive CLODs and the haptic texture rendering algorithm are independent of the model representation.

Deformable Models

Many real-world objects undergo deformations and topological changes. In this dissertation I have proved that multiresolution representations and perceptually based simplification techniques can enable 6-DoF haptic rendering of rigid bodies. Before applying such techniques to 6-DoF haptic rendering of deformable bodies, one must revisit many of the driving observations I have made. For example, the psychophysics studies that set the foundations for the error metrics of CLODs and for the force model for haptic texture rendering assume that the objects involved in the simulation are rigid. The development of error metrics and force models for deformable bodies must be based on different psychophysics studies.

The collision detection techniques that I have developed do not seem to be directly applicable to deformable bodies either. CLODs cannot offer the same performance gains with deformable bodies as with rigid bodies, due to the cost of updating the BVHs. The effectiveness of haptic textures for

refining penetration depth information is also unclear, as the geometric detail at texture level may also deform. To sum up, 6-DoF haptic rendering of complex deformable models will probably require the design of novel techniques.

Multiple Dynamic Objects

A possible way of simulating the dynamics of multiple objects is to apply the same methods I have proposed for the grasped object: implicit integration for rigid body dynamic simulation with penalty-based collisions. However, this approach may increase notably the cost of collision detection and the cost of the implicit integration of rigid body dynamics. Another possible approach is to decouple the simulation of the grasped object from the simulation of the rest of the objects in the scene. This approach exploits the fact that the only object whose motion must be updated at force update rates is the grasped object.

In this dissertation I have addressed the integration of CLODs in rigid body simulation. I have also defined error metrics for determining the contact resolution in collisions between dynamic objects, but I have not studied the problem of touching a dynamic object with the grasped object. The use of multiresolution collision detection algorithms for the interaction with dynamic objects would benefit from perceptual studies on feature detection during dynamic collisions.

6.2.3 Applications and Further Analysis

The effectiveness of the different techniques I have developed can be further analyzed from the perspective of human factors. For instance, it would be interesting to analyze from a perceptual perspective the stability and responsiveness of the system, the influence on task performance of the different techniques that compose the system, the conveyance of roughness with the haptic texture rendering algorithm, and the effects of visual and haptic discrepancies. Further investigation on human kinesthetic perception will undoubtedly be beneficial for overall research in haptic rendering. As discussed earlier, the development of multiresolution techniques for computing the interaction with dynamic and/or deformable bodies requires studies on human perception that will guide the design of error metrics and force models.

A natural way of assessing the effectiveness of the techniques presented in this dissertation will

be to incorporate them into practical applications of 6-DoF haptic rendering. One possibility is the development of a training simulator for sinus surgery. During sinus surgical procedures, surgeons receive visual feedback from endoscopic cameras that are not aligned with their view direction. The images provided by the cameras are difficult to interpret by surgeons, and they often conflict with the haptic cues provided by the interaction of the tools and the sinus cavities. Training on a sinus surgery simulator would reduce the learning curve once the surgeons operate on actual patients. The development of a practical system such as a sinus surgery simulator will require building appropriate software and hardware interfaces, but also thorough testing of the haptic rendering system and fine tuning of parameters for maximizing stability.

The development of a haptic device for sinus surgery simulation introduces an additional challenge: some of the tools employed in sinus surgery present seven DoFs. A small articulation in the tip of the tool adds one DoF to the intrinsic six DoFs of a rigid body. This difference in the number of DoFs poses concerns regarding the applicability of the techniques that I have presented, focused on 6-DoF haptic rendering. Fortunately, the use of virtual coupling solves most of the concerns. The grasped object will no longer be a rigid body, but two rigid bodies with a 1-DoF articulation. The techniques that I have presented for computing collision response remain effective, but the rigid body simulation must be modified to handle articulated bodies. The virtual coupling must also be modified. The interaction paradigm of grasping the virtual object from one point will no longer be valid, and the sinus surgery simulator will require the design of a specialized interaction paradigm. But, conceptually, the function of virtual coupling will be the same, acting as an interface for bidirectional interaction. As a conclusion, the advances that I have developed are not restricted to 6-DoF haptic interaction. I believe that they are applicable to more general devices for kinesthetic feedback, although maybe not to devices for cutaneous feedback, as several perceptual observations that form the basis for my techniques do not apply.

In 1965, Ivan Sutherland suggested an *ultimate display* with force feedback [Sut65]. Almost 40 years later, haptic displays have not yet reached the mass public. Further research is necessary, in both computational techniques and hardware devices. The resources devoted to further research will depend on the successful application of haptic rendering to practical problems. In this dissertation I have presented techniques that enhance the applicability of haptic rendering by enabling 6-DoF haptic

rendering of complex models. As discussed in this chapter, there are still many open research areas, and eventually they will be addressed too, and the dream of the ultimate display will someday come true.

Appendix A

Differentiation Rules

Implicit haptic rendering, as described in Chapter 5, requires the computation of several Jacobians. In this appendix I review some useful differentiation rules for the computation of Jacobians, and I derive terms necessary in the implementation of implicit integration for rigid body dynamic simulation with haptic interaction.

A.1 Vector Differentiation Rules

A.1.1 Jacobian

Given a system of equations expressed in vector form as $\mathbf{y} = \mathbf{f}(\mathbf{x})$, the Jacobian matrix can be written as:

$$J = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_m} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial y_n}{\partial x_1} & \frac{\partial y_n}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_m} \end{pmatrix}. \tag{A.1}$$

Note that, according to the above definition of the Jacobian, the derivative of each of the equations, $\frac{\partial y_i}{\partial \mathbf{x}}$, is represented as a row vector.

A.1.2 Derivative of a dot product

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T \mathbf{v} \tag{A.2}$$

$$\frac{\partial (\mathbf{u} \cdot \mathbf{v})}{\partial \mathbf{w}} = \mathbf{u}^T \frac{\partial \mathbf{v}}{\partial \mathbf{w}} + \mathbf{v}^T \frac{\partial \mathbf{u}}{\partial \mathbf{w}}. \tag{A.3}$$

A.1.3 Derivative of a cross product

A cross product $\mathbf{u} \times \mathbf{v}$ can be regarded as a linear transformation on $\mathbf{v}$:

$$\mathbf{u} \times \mathbf{v} = \mathbf{u}^* \mathbf{v}, \tag{A.4}$$

where $\mathbf{u}^*$ is a matrix defined as:

$$\mathbf{u}^* = \begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix}. \tag{A.5}$$

Note some properties of cross products:

$$\mathbf{u} \times \mathbf{v} = -\mathbf{v} \times \mathbf{u}, \tag{A.6}$$

$$\mathbf{u}^* \mathbf{v} = -\mathbf{v}^* \mathbf{u}. \tag{A.7}$$

From these, one can deduce that:

$$\frac{\partial (\mathbf{u} \times \mathbf{v})}{\partial \mathbf{w}} = \mathbf{u}^* \frac{\partial \mathbf{v}}{\partial \mathbf{w}} - \mathbf{v}^* \frac{\partial \mathbf{u}}{\partial \mathbf{w}}. \tag{A.8}$$

A.1.4 Gradient

The gradient can be regarded as the computation of the derivative of a scalar function w.r.t. a vector. Using the notation of Jacobians introduced earlier, the derivative of each scalar function is represented as a row vector. Considering the gradient as a column vector leaves the following relation:

$$\nabla_{\mathbf{w}} f(\mathbf{w}) = \left(\frac{\partial (f(\mathbf{w}))}{\partial \mathbf{w}} \right)^T. \tag{A.9}$$

A.1.5 Hessian

The Hessian matrix is the Jacobian of the gradient of a scalar function. Therefore, it can be represented as:

$$\mathscr{H}_{\mathbf{w}} f(\mathbf{w}) = \frac{\partial \left(\nabla_{\mathbf{w}} f(\mathbf{w})\right)}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial^2 f}{\partial \mathbf{w}_1{}^2} & \frac{\partial^2 f}{\partial \mathbf{w}_1 \partial \mathbf{w}_2} & \cdots & \frac{\partial^2 f}{\partial \mathbf{w}_1 \partial \mathbf{w}_n} \\ \frac{\partial^2 f}{\partial \mathbf{w}_2 \partial \mathbf{w}_1} & \frac{\partial^2 f}{\partial \mathbf{w}_2{}^2} & \cdots & \frac{\partial^2 f}{\partial \mathbf{w}_2 \partial \mathbf{w}_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial \mathbf{w}_n \partial \mathbf{w}_1} & \frac{\partial^2 f}{\partial \mathbf{w}_n \partial \mathbf{w}_2} & \cdots & \frac{\partial^2 f}{\partial \mathbf{w}_n{}^2} \end{pmatrix}. \tag{A.10}$$

A.1.6 Derivative of a vector multiplied by a scalar function

$$\frac{\partial \left(f(\mathbf{w})\mathbf{u}\right)}{\partial \mathbf{w}} = f(\mathbf{w})\frac{\partial \mathbf{u}}{\partial \mathbf{w}} + \mathbf{u}\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}}. \tag{A.11}$$

A.1.7 Derivative of a vector multiplied by a matrix

In this case it is more convenient to express the derivative w.r.t. each of the components of the vector separately:

$$\frac{\partial \left(M\mathbf{u}\right)}{\partial w_i} = M\frac{\partial \mathbf{u}}{\partial w_i} + \frac{\partial M}{\partial w_i}\mathbf{u}. \tag{A.12}$$

A.2 Rotations

A.2.1 Quaternions

Let us define a unit quaternion $\mathbf{q} = (\mathbf{u}, s)$, where $\mathbf{u} = (x, y, z)$ is the vector part, and s is the scalar part. The inverse of $\mathbf{q}$, $\mathbf{q}^{-1}$, is defined as:

$$\mathbf{q}^{-1} = (-\mathbf{u}, s) = (-x, -y, -z, s). \tag{A.13}$$

A.2.2 Product of Quaternions

The product of two quaternions **ab** is defined as:

$$\mathbf{ab} = (a_s b_u + b_s a_u + a_u \times b_u, a_s b_s - a_u \cdot b_u). \tag{A.14}$$

It can also be regarded as a linear transformation on **b**, and represented as a matrix-vector product:

$$\mathbf{ab} = A\mathbf{b},$$
$$A = \begin{pmatrix} a_u^* + a_s I & a_u \\ -a_u^T & a_s \end{pmatrix} = \begin{pmatrix} a_s & -a_z & a_y & a_x \\ a_z & a_s & -a_x & a_y \\ -a_y & a_x & a_s & a_z \\ -a_x & -a_y & -a_z & a_s \end{pmatrix}. \tag{A.15}$$

Similarly, it can be regarded as a linear transformation on **a**:

$$\mathbf{ab} = B\mathbf{a},$$
$$B = \begin{pmatrix} -b_u^* + b_s I & b_u \\ -b_u^T & b_s \end{pmatrix} = \begin{pmatrix} b_s & b_z & -b_y & b_x \\ -b_z & b_s & b_x & b_y \\ b_y & -b_x & b_s & b_z \\ -b_x & -b_y & -b_z & b_s \end{pmatrix}. \tag{A.16}$$

Note that there are some differences between the matrix A defined in Eq. A.15 and the matrix B defined in Eq. A.16, because quaternion product is not commutative.

A.2.3 Derivative of a Product of Quaternions

It is convenient to regard the product **ab** as a linear transformation on **b**, and express it as $A\mathbf{b}$. Then, following Eq. A.12, the derivative w.r.t. each component of a vector **w** is:

$$\frac{\partial(\mathbf{ab})}{\partial w_i} = \frac{\partial(A\mathbf{b})}{\partial w_i} = A\frac{\partial \mathbf{b}}{\partial w_i} + \frac{\partial A}{\partial w_i}\mathbf{b}. \tag{A.17}$$

It is also interesting to study the derivative w.r.t. one of the quaternions involved in the product.

For example, the computation of $\frac{\partial(\mathbf{ab})}{\partial \mathbf{a}}$ requires the following matrices:

$$\frac{\partial A}{\partial a_x} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}, \qquad \frac{\partial A}{\partial a_y} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix},$$

$$\frac{\partial A}{\partial a_z} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix}, \qquad \frac{\partial A}{\partial a_s} = I.$$

A.2.4 Quaternions and Rotations

3D rotations can be represented using unit quaternions. A rotation θ around a unit vector $\mathbf{u}$ is represented by a quaternion $\mathbf{q}$, where:

$$\mathbf{q} = \left(\sin\left(\frac{\theta}{2}\right)\mathbf{u}, \cos\left(\frac{\theta}{2}\right)\right). \tag{A.18}$$

Then, $\mathbf{q}$ can be used to rotate a vector $\mathbf{v}$, applying two quaternion products:

$$\mathbf{v}_{q,rot} = \mathbf{q}\mathbf{v}_q\mathbf{q}^{-1}, \tag{A.19}$$

where $\mathbf{v}_q$ is a quaternion constructed as $\mathbf{v}_q = (\mathbf{v}, 0)$, and the resulting rotated vector $\mathbf{v}_{rot}$ is the vector part of $\mathbf{v}_{q,rot}$.

The relation between a quaternion and a rotation matrix can be obtained by expressing $\mathbf{v}_{rot}$ as a linear transformation of $\mathbf{v}$:

$$\mathbf{v}_{rot} = R\mathbf{v}. \tag{A.20}$$

Given $\mathbf{q} = (x, y, z, s)$, one can deduce that:

$$R = \begin{pmatrix} x^2 - y^2 - z^2 + s^2 & 2yx - 2zs & 2zx + 2ys \\ 2xy + 2zs & -x^2 + y^2 - z^2 + s^2 & 2zy - 2xs \\ 2xz - 2ys & 2yz + 2xs & -x^2 - y^2 + z^2 + s^2 \end{pmatrix}. \tag{A.21}$$

A.2.5 Derivative of a Rotation w.r.t. the Quaternion

In order to differentiate a rotation $\mathbf{q}\mathbf{v}_q\mathbf{q}^{-1}$ w.r.t. $\mathbf{q}$ itself, it is convenient to represent the rotation using the rotation matrix R defined by $\mathbf{q}$. Then, the derivative is simply a special case of the expression defined in Eq. A.12. The derivative w.r.t. each component of $\mathbf{q}$ is:

$$\frac{\partial \mathbf{v}_{rot}}{\partial q_i} = \frac{\partial (R\mathbf{v})}{\partial q_i} = R\frac{\partial \mathbf{v}}{\partial q_i} + \frac{\partial R}{\partial q_i}\mathbf{v}. \tag{A.22}$$

Given $\mathbf{q} = (x, y, z, s)$, the partial derivatives of the rotation matrix R are:

$$\frac{\partial R}{\partial x} = 2\begin{pmatrix} x & y & z \\ y & -x & -s \\ z & s & -x \end{pmatrix}, \qquad \frac{\partial R}{\partial y} = 2\begin{pmatrix} -y & x & s \\ x & y & z \\ -s & z & -y \end{pmatrix},$$
$$\frac{\partial R}{\partial z} = 2\begin{pmatrix} -z & -s & x \\ s & -z & y \\ x & y & z \end{pmatrix}, \qquad \frac{\partial R}{\partial s} = 2\begin{pmatrix} s & -z & y \\ z & s & -x \\ -y & x & s \end{pmatrix}. \tag{A.23}$$

A.2.6 Time Derivative of a Rigid Body's Quaternion

The orientation of a rigid body can be described by a quaternion $\mathbf{q}$. Mirtich [Mir96] describes the time derivative of this quaternion based on the angular velocity ω as:

$$\dot{\mathbf{q}} = \frac{1}{2}\omega_q\mathbf{q}. \tag{A.24}$$

As indicated by Eq. A.16, this formula can be expressed as a linear transformation on ω_q. Given $\mathbf{q} = (\mathbf{u}, s) = (x, y, z, s)$, and knowing that $\omega_q = (\omega, 0)$, the linear transformation is:

$$\dot{\mathbf{q}} = Q\omega,$$

$$Q = \frac{1}{2}\begin{pmatrix} -\mathbf{u}^* + sI \\ -\mathbf{u}^T \end{pmatrix} = \frac{1}{2}\begin{pmatrix} s & z & -y \\ -z & s & x \\ y & -x & s \\ -x & -y & -z \end{pmatrix}. \tag{A.25}$$

The Jacobian of $\dot{\mathbf{q}}$ w.r.t. $\mathbf{q}$, following Eq. A.12, requires the partial derivatives of Q w.r.t. each of the components of $\mathbf{q}$.

$$\frac{\partial Q}{\partial x} = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \qquad \frac{\partial Q}{\partial y} = \frac{1}{2}\begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix},$$

$$\frac{\partial Q}{\partial z} = \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \qquad \frac{\partial Q}{\partial s} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

A.2.7 Transformation between Euler Angles

Euler angles describe arbitrary rotations as 3 successive rotations around the coordinate axes. Assuming XYZ Euler angles in global coordinates, the resulting rotation matrix is:

$$R = Rot(z,\theta_z)\ Rot(y,\theta_y)\ Rot(x,\theta_x) = \begin{pmatrix} c\theta_y c\theta_z & s\theta_x s\theta_y c\theta_z + c\theta_x s\theta_z & -c\theta_x s\theta_y c\theta_z + s\theta_x s\theta_z \\ -c\theta_y s\theta_z & -s\theta_x s\theta_y s\theta_z + c\theta_x c\theta_z & c\theta_x s\theta_y s\theta_z + s\theta_x c\theta_z \\ s\theta_y & -s\theta_x c\theta_y & c\theta_x c\theta_y \end{pmatrix}. \tag{A.26}$$

For compactness, the sine function is represented as *s*, and the cosine function as *c*.

Given two reference systems, a global reference system $\{X,Y,Z\}$ and a local reference system $\{\mathbf{u},\mathbf{v},\mathbf{n}\}$, the transformation from the global to the local reference system is $R = (\mathbf{u}\,\mathbf{v}\,\mathbf{n})^T$. A rotation with Euler angles θ in the global reference system can also be expressed as a rotation with Euler angles θ' in the local reference system. The relation between global and local Euler angles is:

$$Rot(\theta')\,R = R\,Rot(\theta). \tag{A.27}$$

The Jacobian $\frac{\partial\theta'}{\partial\theta}$ describes the differential change of local Euler angles as a result of a differential change of global Euler angles. Assuming infinitesimal rotations, the 3 angles are decoupled [GPS02] and their order does not matter (This observation also implies that, for infinitesimal rotations, the overall rotation is independent of the particular definition of Euler angles). Then, the above expression can be rewritten as:

$$\left(I - \theta'^*\right) R = R\left(I - \theta^*\right). \tag{A.28}$$

This expression can be simplified to:

$$\theta'^* = R\theta^* R^T. \tag{A.29}$$

Substituting the definition of R yields the following relation:

$$\theta'^* = \begin{pmatrix} \mathbf{u}^T\theta^*\mathbf{u} & \mathbf{u}^T\theta^*\mathbf{v} & \mathbf{u}^T\theta^*\mathbf{n} \\ \mathbf{v}^T\theta^*\mathbf{u} & \mathbf{v}^T\theta^*\mathbf{v} & \mathbf{v}^T\theta^*\mathbf{n} \\ \mathbf{n}^T\theta^*\mathbf{u} & \mathbf{n}^T\theta^*\mathbf{v} & \mathbf{n}^T\theta^*\mathbf{n} \end{pmatrix} = \begin{pmatrix} 0 & -\theta\cdot\mathbf{n} & \theta\cdot\mathbf{v} \\ \theta\cdot\mathbf{n} & 0 & -\theta\cdot\mathbf{u} \\ -\theta\cdot\mathbf{v} & \theta\cdot\mathbf{u} & 0 \end{pmatrix}. \tag{A.30}$$

From this relation, the Jacobian can be computed as:

$$\frac{\partial\theta'}{\partial\theta} = R = (\mathbf{u}\,\mathbf{v}\,\mathbf{n})^T. \tag{A.31}$$

A.2.8 Derivative of Euler Angles w.r.t. the Quaternion

The combination of Eqs. A.21 and A.26 sets the relation between XYZ Euler angles and a quaternion. From this relation, the Euler angles can be expressed in terms of the components of the quaternion as:

$$\begin{aligned}
\theta_x &= \tan^{-1}\left(\frac{q_y q_z + q_x q_s}{q_x^2 + q_y^2 - q_z^2 - q_s^2}\right) = \tan^{-1}\left(\frac{A}{B}\right), \\
\theta_y &= \sin^{-1}\left(q_x q_z - q_y q_s\right) = \sin^{-1}\left(C\right), \\
\theta_z &= \tan^{-1}\left(\frac{q_x q_y + q_z q_s}{q_y^2 + q_z^2 - q_x^2 - q_s^2}\right) = \tan^{-1}\left(\frac{D}{E}\right).
\end{aligned} \tag{A.32}$$

The Jacobian can be written as:

$$\frac{\partial \theta}{\partial \mathbf{q}} = \begin{pmatrix}
\frac{q_s B - 2q_x A}{A^2+B^2} & \frac{q_z B - 2q_y A}{A^2+B^2} & \frac{q_y B + 2q_z A}{A^2+B^2} & \frac{q_x B + 2q_s A}{A^2+B^2} \\
\frac{q_z}{\sqrt{1-C^2}} & \frac{-q_s}{\sqrt{1-C^2}} & \frac{q_x}{\sqrt{1-C^2}} & \frac{-q_y}{\sqrt{1-C^2}} \\
\frac{q_y E + 2q_x D}{D^2+E^2} & \frac{q_x E - 2q_y D}{D^2+E^2} & \frac{q_s E - 2q_z D}{D^2+E^2} & \frac{q_z E + 2q_s D}{D^2+E^2}
\end{pmatrix}. \tag{A.33}$$

BIBLIOGRAPHY

[AGHP+00] P. Agarwal, L. Guibas, S. Har-Peled, A. Rabinovitch, and M. Sharir. Penetration depth of two convex polytopes in 3d. *Nordic J. Computing*, 7:227–240, 2000.

[AH98a] R. J. Adams and B. Hannaford. A two-port framework for the design of unconditionally stable haptic interfaces. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[AH98b] O. R. Astley and V. Hayward. Multirate haptic simulation achieved by coupling finite element meshes through norton equivalents. *Proc. of IEEE International Conference on Robotics and Automation*, 1998.

[AKO95] Y. Adachi, T. Kumano, and K. Ogino. Intermediate representation for stiff virtual objects. *Virtual Reality Annual International Symposium*, pages 203–210, 1995.

[And02] C. Andriot. Advances in virtual prototyping. *Clefs CEA*, Vol. 47, Research and Simulation, 2002.

[AS96] R. S. Avila and L. M. Sobierajski. A haptic interaction method for volume visualization. In *Proc. of IEEE Visualization Conference*, 1996.

[Bar89] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Computer Graphics (Proc. of ACM SIGGRAPH)*, volume 23, pages 223–232, 1989.

[Bar91] D. Baraff. Coping with friction for non-penetrating rigid body simulation. In *Computer Graphics (Proc. of ACM SIGGRAPH))*, volume 25, pages 31–40, 1991.

[Bar92] D. Baraff.s *Dynamic simulation of non-penetrating rigid body simulation*. PhD thesis, Cornell University, 1992.

[Bar94] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proc. of ACM SIGGRAPH*, pages 23–34, 1994.

[Ber99] P. J. Berkelman. *Tool-Based Haptic Interaction with Dynamic Physical Simulations Using Lorentz Magnetic Levitation*. PhD thesis, Carnegie Mellon University, 1999.

[BH95] P. J. Berkelman and R. L. Hollis. Interacting with virtual environments using a magnetic levitation haptic interface. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995.

[BHS97] C. Basdogan, C.-H. Ho, and M. A. Srinivasan. A ray-based haptic rendering technique for displaying shape and texture of 3d objects in virtual environments. *Proc. of ASME Dynamic Systems and Control Division*, 61, pages 77–84, 1997.

[BKSS90] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. *Proc. of ACM SIGMOD*, pages 322–331, 1990.

[Bli78] J. F. Blinn. Simulation of wrinkled surfaces. In *Computer Graphics (Proc. of ACM SIGGRAPH)*, pages 286–292, 1978.

[BOYBK90] F. P. Brooks, Jr., M. Ouh-Young, J. J. Batter, and P. J. Kilpatrick. Project GROPE — Haptic displays for scientific visualization. In *Computer Graphics (Proc. of ACM SIGGRAPH)*, volume 24, pages 177–185, 1990.

[BS80] A. Bejczy and J. K. Salisbury. Kinematic coupling between operator and remote manipulator. *Advances in Computer Technology*, volume 1, 197–211, 1980.

[Bur96] G. Burdea. *Force and Touch Feedback for Virtual Reality*. John Wiley and Sons, 1996.

[BW98] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proc. of ACM SIGGRAPH*, pages 43–54, 1998.

[BW01] D. Baraff and A. Witkin. *Physically-Based Modeling*. ACM SIGGRAPH Course Notes, 2001.

[Cam97] S. Cameron. Enhancing GJK: Computing minimum and penetration distance between convex polyhedra. *Proc. of IEEE International Conference on Robotics and Automation*, pages 3112–3117, 1997.

[Cat74] E. E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, University of Utah, 1974.

[CB94] J. E. Colgate and J. M. Brown. Factors affecting the z-width of a haptic display. *Proc. of IEEE International Conference on Robotics and Automation*, pages 3205–3210, 1994.

[CC86] S. Cameron and R. K. Culley. Determining the minimum translational distance between two convex polyhedra. *Proc. of International Conference on Robotics and Automation*, pages 591–596, 1986.

[CC97] B. Chang and J. E. Colgate. Real-time impulse-based simulation of rigid body systems for haptic display. *Proc. of ASME Dynamic Systems and Control Division*, 1997.

[CD68] R. W. Cottle and G. B. Dantzig. Complementarity pivot theory of mathematical programming. *Linear Algebra and its Applications*, pages 103–125, 1968.

[CDST97] B. Chazelle, D. Dobkin, N. Shouraboura, and A. Tal. Strategies for polyhedral surface decomposition: An experimental study. *Computational Geometry: Theory and Applications*, 7:327–342, 1997.

[CGSS93] J. E. Colgate, P. E. Grafing, M. C. Stanley, and G. Schenkel. Implementation of stiff virtual walls in force-reflecting interfaces. *Virtual Reality Annual International Symposium*, pages 202–207, 1993.

[Che99] E. Chen. Six degree-of-freedom haptic system for desktop virtual prototyping applications. In *Proceedings of the First International Workshop on Virtual Reality and Prototyping*, pages 97–106, 1999.

[CJ92] C. E. Connor and K. O. Johnson. Neural coding of tactile texture: Comparison of spatial and temporal mechanisms for roughness perception. *Journal of Neuroscience*, 12:pages 3414–3426, 1992.

[COM98] J. Cohen, M. Olano, and D. Manocha. Appearance preserving simplification. In *Proc. of ACM SIGGRAPH*, pages 115–122, 1998.

[Cos00] M. A. Costa. *Fractal Description of Rough Surfaces for Haptic Display*. PhD thesis, Stanford University, 2000.

[CpS92] R. W. Cottle, J. S. pang, and R. E. Stone. The linear complementarity problem. *Academic-Press, Inc.*, 1992.

[CS94] J. E. Colgate and G. G. Schenkel. Passivity of a class of sampled-data systems: Application to haptic interfaces. *Proc. of American Control Conference*, 1994.

[CSB95] J. E. Colgate, M. C. Stanley, and J. M. Brown. Issues in the haptic display of tool use. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 140–145, 1995.

[CSC04] D. Constantinescu, S. E. Salcudean, and E. A. Croft. Impulsive forces for haptic rendering of rigid contact. *Proc. of International Symposium on Robotics*, pages 1–6, 2004.

[ÇT00] M. C. Çavuşoğlu and F. Tendick. Multirate simulation for high fidelity haptic interaction with deformable objects in virtual environments. *Proc. of IEEE International Conference on Robotics and Automation*, pages 2458–2465, 2000.

[CT03a] S. Choi and H. Z. Tan. Aliveness: Perceived instability from a passive haptic texture rendering system. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[CT03b] S. Choi and H. Z. Tan. An experimental study of perceived instability during haptic texture rendering: Effects of collision detection algorithm. In *Proc. of Haptics Symposium*, pages 197–204, 2003.

[ÇTS02] M. C. Çavuşoğlu, F. Tendick, and S. S. Sastry. Haptic interfaces to real and virtual surgical environments. In M. L. McLaughlin, J. P. Hespanha, and G. S. Sukhatme, editors, *Touch in Virtual Environments*, chapter 13, pages 217–237. Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[DAK04] C. Duriez, C. Andriot, and A. Kheddar. A multi-threaded approach for deformable/rigid contacts with haptic feedback. In *Proc. of Haptics Symposium*, 2004.

[DDCB01] G. Debunne, M. Desbrun, M. P. Cani, and A. H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Proc. of ACM SIGGRAPH*, 2001.

[DHKS93] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri. Computing the intersection-depth of polyhedra. *Algorithmica*, 9:518–533, 1993.

[DK90] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra – a unified approach. In *Proc. of 17th International Colloquium on Automata Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 400–413, Springer-Verlag, 1990.

[DQ+99] F. Dachille, H. Qin, , A. Kaufman, and J. El-Sana. Haptic sculpting of dynamic surfaces. In *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 103–110, 1999.

[EB01] M. O. Ernst and M. S. Banks. Does vision always dominate haptics? *Touch in Virtual Environments Conference*, 2001.

[EDD+95] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proc. of ACM SIGGRAPH*, pages 173–182, 1995

[EHS+97] C. Edmond, D. Heskamp, D. Sluis, D. Stredney, G. Wiet, R. Yagel, S. Weghorst, P. Oppenheimer, J. Miller, M. Levin, and L. Rosenberg. ENT endoscopic surgical simulator. *Proc. of Medicine Meets VR*, pages 518–528, 1997.

[EL00] S. Ehmann and M. C. Lin. Accelerated proximity queries between convex polyhedra using multi-level voronoi marching. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2101–2106, 2000.

[EL01] S. Ehmann and M. C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In *Proc. of Eurographics*, pages 500–510, 2001.

[ESJ97] R. E. Ellis, N. Sarkar, and M. A. Jenkins. Numerical methods for the force reflection of contact. *ASME Transactions on Dynamic Systems, Modeling and Control*, 119:768–774, 1997.

[ESV00] J. El-Sana and A. Varshney. Continuously-adaptive haptic rendering. In *Proc. of Virtual Environments Conference*, pages 135–144, 2000.

[FFM+04] B. Fisher, S. Fels, K. MacLean, T. Munzner, and R. Rensink. *Seeing, hearing and touching: Putting it all together*. ACM SIGGRAPH course notes, 2004.

[FL01] S. Fisher and M. C. Lin. Fast penetration depth estimation for elastic bodies using deformed distance fields. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[FOL02] M. Foskey, M. A. Otaduy, and M. C. Lin. ArtNova: Touch-enabled 3D model design. *Proc. of IEEE Virtual Reality Conference*, 2002.

[GBF03] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)*, volume 22, pages 871–878, 2003.

[GCH+01] S. Grange, F. Conti, P. Helmer, P. Rouiller, and C. Baur. Overview of the delta haptic device. In *Proc. of Eurohaptics Conference*, 2001.

[GEL00] A. Gregory, S. Ehmann, and M. C. Lin. *inTouch*: Interactive multiresolution modeling and 3d painting with a haptic interface. In *Proc. of IEEE Virtual Reality Conference*, 2000.

[GFL04] P. Garrec, J.-P. Friconneau, and F. Louveau. Virtuose 6d: A new industrial master arm using innovative ball-screw actuators. *Proc. of International Symposium on Robotics*, 2004.

[GH97] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proc. of ACM SIGGRAPH*, pages 209–216, 1997.

[GHZ99] L. Guibas, D. Hsu, and L. Zhang. *H-Walk*: Hierarchical distance computation for moving convex bodies. *Proc. of ACM Symposium on Computational Geometry*, 1999.

[GJK88] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4:193–203, 1988.

[GLGT99] A. Gregory, M. C. Lin, S. Gottschalk, and R. M. Taylor II. H-COLLIDE: A framework for fast and accurate collision detection for haptic interaction. In *Proc. of Virtual Reality Conference*, pages 38–45, 1999.

[GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In *Proc. of ACM SIGGRAPH*, pages 171–180, 1996.

[GLW+04] N. Govindaraju, B. Lloyd, W. Wang, M. C. Lin, and D. Manocha. Fast computation of database operations using graphics processors. *Proc. of ACM SIGMOD*, 2004.

[GME+00] A. Gregory, A. Mascarenhas, S. Ehmann, M. C. Lin, and D. Manocha. 6-DoF haptic display of polygonal models. *Proc. of IEEE Visualization Conference*, 2000.

[Got00] S. Gottschalk. *Collision Queries using Oriented Bounding Boxes*. PhD thesis, University of North Carolina at Chapel Hill, 2000.

[GPS02] H. Goldstein, C. Poole, and J. Safko. *Classical Mechanics (3rd Ed.)*. Addison-Wesley, Reading, MA, 2002.

[GRLM03] N. Govindaraju, S. Redon, M. C. Lin, and D. Manocha. CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware. In *Proc. of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pages 25–32, 2003.

[GSM+97] S. Gibson, J. Samosky, A. Mor, C. Fyock, E. Grimson, and T. Kanade. Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback. *First Joint Conference on Computer Vision, Virtual Reality and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery (CVMed-MRCAS)*, pages 368–378, 1997.

[GSS99] I. Guskov, W. Sweldens, and P. Schroder. Multiresolution signal processing for meshes. In *Proc. of ACM SIGGRAPH*, pages 325 – 334, 1999.

[GT54] R. Goertz and R. Thompson. Electronically controlled manipulator. *Nucleonics*, pages 46–47, 1954.

[HA00] V. Hayward and B Armstrong. A new computational model of friction applied to haptic rendering. *Experimental Robotics VI*, 2000.

[Hay01] V. Hayward. Survey of haptic interface research at mcgill university. *Workshop in Interactive Multimodal Telepresence Systems*, pages 91–98, 2001.

[HBS99] C.-H. Ho, C. Basdogan, and M. A. Srinivasan. Efficient point-based rendering techniques for haptic display of virtual objects. *Presence*, volume 8(5), pages 477–491, 1999.

[HC02] C. J. Hasser and M. R. Cutkosky. System identification of the human hand grasping a haptic knob. In *Proc. of Haptics Symposium*, pages 180–189, 2002.

[HCGB99] M. A. Heller, J. A. Calcaterra, S. L. Green, and L. Brown. Intersensory conflict between vision and touch: The response modality dominates when precise, attention-riveting judgements are required. *Perception and Psychophysics*, volume 61, pages 1384–1398, 1999.

[HGA+98] V. Hayward, P. Gregorio, O. Astley, S. Greenish, and M. Doyon. Freedom-7: A high fidelity seven axis haptic device with applications to surgical training. *Experimental Robotics*, pages 445–456, 1998.

[Hog85] N. Hogan. Impedance control: An approach to manipulation, part i - theory, part ii - implementation, part iii - applications. *Journal of Dynamic Systems, Measurement and Control*, volume 107, pages1–24, 1985.

[Hog86] N. Hogan. Multivariable mechanics of the neuromuscular system. *IEEE Annual Conference of the Engineering in Medicine and Biology Society*, pages 594–598, 1986.

[Hop96] H. Hoppe. Progressive meshes. In *Proc. of ACM SIGGRAPH*, pages 99–108, 1996.

[Hop97] H. Hoppe. View dependent refinement of progressive meshes. In *Proc. of ACM SIGGRAPH*, pages 189–198, 1997.

[HR00] M. Hollins and S. R. Risner. Evidence for the duplex theory of tactile texture perception. *Perception and Psychophysics*, volume 62, pages 695–705, 2000.

[HRK02] B. Hannaford, J.-H. Ryu, and Y. S. Kim. Stable control of haptics. In M. L. McLaughlin, J. P. Hespanha, and G. S. Sukhatme, editors, *Touch in Virtual Environments*, chapter 3, pages 47–70. Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[HS77] J. W. Hill and J. K. Salisbury. Two measures of performance in a peg-in-hole manipulation task with force feedback. *Thirteenth Annual Conference on Manual Control*, 1977.

[HS04] S. Hasegawa and M. Sato. Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects. In *Proc. of Eurographics*, 2004.

[Hub94] P. Hubbard. *Collision Detection for Interactive Graphics Applications*. PhD thesis, Brown University, 1994.

[HZLM01] K. Hoff, A. Zaferakis, M. C. Lin, and D. Manocha. Fast and simple 2d geometric proximity queries using graphics hardware. In *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 145–148, 2001.

[IMWB01] B. Insko, M. Meehan, M. Whitton, and F. Brooks. Passive haptics significantly enhances virtual environments. Technical Report 01-010, Department of Computer Science, University of North Carolina at Chapel Hill, 2001.

[Ins01] B. Insko. *Passive Haptics Significantly Enhance Virtual Environments*. PhD thesis, University of North Carolina at Chapel Hill, 2001.

[JC01] D. E. Johnson and E. Cohen. Spatialized normal cone hierarchies. In *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 129–134, 2001.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[JP04] D. L. James and D. K. Pai. Bd-tree: Output-sensitive collision detection for reduced deformable models. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)*, 2004.

[JTK+99] D. E. Johnson, T. V. Thompson II, M. Kaplan, D. Nelson, and E. Cohen. Painting textures with a haptic interface. In *Proc. of IEEE Virtual Reality Conference*, 1999.

[JW03] D. E. Johnson and P. Willemsen. Six degree of freedom haptic rendering of complex polygonal models. In *Proc. of Haptics Symposium*, 2003.

[JW04] D. E. Johnson and P. Willemsen. Accelerated haptic rendering of polygonal models through local descent. In *Proc. of Haptics Symposium*, 2004.

[Kar85] D. Karnopp. Computer simulation of stick slip friction in mechanical dynamic systems. *Trans. ASME, Journal of Dynamic Systems, Measurement, and Control*, 1985.

[Kat89] D. Katz. *The World of Touch*. Erlbaum, Hillsdale, NJ, 1989. L. Krueger, Trans. (Original work published 1925).

[KB91] W. Kim and A. Bejczy. Graphical displays for operator aid in telemanipulation. *IEEE International Conference on Systems, Man and Cybernetics*, 1991.

[KHM+98] J. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Trans. on Visualization and Computer Graphics*, 4(1):21–37, 1998.

[Kil76] P. J. Kilpatrick. *The use of a kinesthetic supplement in an interactive graphics system*. PhD thesis, University of North Carolina at Chapel Hill, 1976.

[KKH+97] U. G. Kuhnapfel, C. Kuhn, M. Hubner, H.-G. Krumm, H. Maass, and B. Neisius. The karlsruhe endoscopic surgery trainer as an example for virtual reality in medical education. *Minimally Invasive Therapy and Allied Technologies*, Vol. 6:122–125, 1997.

[KL95] R. L. Klatzky and S. J. Lederman. Identifying objects from a haptic glance. *Perception and Psychophysics*, volume 57, pages 1111–1123, 1995.

[KL99] R. L. Klatzky and S. J. Lederman. Tactile roughness perception with a rigid link interposed between skin and surface. *Perception and Psychophysics*, volume 61, pages 591–607, 1999.

[KL02] R. L. Klatzky and S. J. Lederman. Perceiving texture through a probe. In M. L. McLaughlin, J. P. Hespanha, and G. S. Sukhatme, editors, *Touch in Virtual Environments*, chapter 10, pages 180–193. Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[KL03] R. L. Klatzky and S. J. Lederman. Touch. In A. F. Healy and R. W. Proctor, editors, *Experimental Psychology*, volume 4, pages 147–176. In I.B. Weiner (Editor-in-Chief), *Handbook of Psychology*, 2003.

[KLH+03] R. L. Klatzky, S. J. Lederman, C. Hamilton, M. Grindley, and R. H. Swendsen. Feeling textures through a probe: Effects of probe and surface geometry and exploratory factors. *Perception and Psychophysics*, volume 65(4), pages 613–631, 2003.

[KLM02] Y. J. Kim, M. C. Lin, and D. Manocha. DEEP: an incremental algorithm for penetration depth computation between convex polytopes. *Proc. of IEEE International Conference on Robotics and Automation*, pages 921–926, 2002.

[KOLM02] Y. J. Kim, M. A. Otaduy, M. C. Lin, and D. Manocha. Fast Penetration Depth Computation for Physically-based Animation. In *Proc. of ACM Symposium on Computer Animation*, 2002.

[KOLM03] Y. J. Kim, M. A. Otaduy, M. C. Lin, and D. Manocha. Six-degree-of-freedom haptic rendering using incremental and localized computations. *Presence*, volume 12(3), pages 277–295, 2003.

[Lar01] E. Larsen. A robot soccer simulator: A case study for rigid body contact. *Game Developers Conference*, 2001.

[LC91] M. C. Lin and J. F. Canny. Efficient algorithms for incremental distance computation. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 2, pages 1008–1014, 1991.

[LCN99] J. C. Lombardo, M.-P. Cani, and F. Neyret. Real-time collision detection for virtual surgery. *Proc. of Computer Animation*, 1999.

[LDW97] M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. In *ACM Trans. on Graphics*, volume 16(1), pages 34–73, 1997.

[LE97] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygon environments. In *Proc. of ACM SIGGRAPH*, 1997.

[Led74] S. J. Lederman. Tactile roughness of grooved surfaces: The touching process and the effects of macro- and microsurface structure. In *Perception and Psychophysics*, volume 16, pages 385–395, 1974.

[Lem65] C. E. Lemke. Bimatrix equilibrium points and mathematical programming. *Management Science*, 11:681–689, 1965.

[LG98] M. C. Lin and S. Gottschalk. Collision detection between geometric models: A survey. *Proc. of IMA Conference on Mathematics of Surfaces*, 1998.

[LGLM00] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Distance queries with rectangular swept sphere volumes. *Proc. of IEEE International Conference on Robotics and Automation*, 2000.

[Lin93] M. C. Lin. *Efficient Collision Detection for Animation and Robotics*. PhD thesis, University of California, Berkeley, 1993.

[LKHG00] S. J. Lederman, R. L. Klatzky, C. Hamilton, and M. Grindley. Perceiving surface roughness through a probe: Effects of applied force and probe diameter. *Proc. of ASME Dynamic Systems and Control Division*, 2000.

[LKHR99] S. J. Lederman, R. L. Klatzky, C. Hamilton, and G. I. Ramsay. Perceiving roughness via a rigid stylus: Psychophysical effects of exploration speed and mode of touch. *Haptics-e*, 1999.

[Llo57] S. P. Lloyd. Least squares quantization in PCM'S. *Bell Telephone Labs Memo*, 1957.

[LM04] M. C. Lin and D. Manocha. Collision and proximity queries. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, chapter 35, pages 787–807. CRC Press LLC, Boca Raton, FL, 2004.

[Löt84] P. Lötstedt. Numerical simulation of time-dependent contact friction problems in rigid body mechanics. *SIAM Journal of Scientific Statistical Computing*, 5:370–393, 1984.

[LPD+98] D. A. Lawrence, L. Y. Pao, A. M. Dougherty, Y. Pavlou, S. W. Brown, and S. A. Wallace. Human perceptual thresholds of friction in haptic interfaces. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 287–294, 1998.

[LRC+02] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan-Kaufmann, 2002.

[LS91] R. H. LaMotte and M. A. Srinivasan. Surface microgeometry: Tactile perception and neural encoding. In O. Franzen and J. Westman, editors, *Information Processing in the Somatosensory System*, pages 49–58. Macmillan Press, London, 1991.

[LT98] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. *Proc. of IEEE Visualization Conference*, pages 279–286, 1998.

[MC95] B. Mirtich and J. Canny. Impulse-based simulation of rigid bodies. In *Proc. of ACM Symposium on Interactive 3D Graphics*, 1995.

[MCF90] B. E. Miller, J. E. Colgate, and R. A. Freeman. Guaranteed stability of haptic systems with nonlinear virtual environments. *IEEE Trans. on Robotics and Automation*, 16(6):712–719, 1990.

[MHS02] M. L. McLaughlin, J. P. Hespanha, and G. S. Sukhatme. *Touch in Virtual Environments*. Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[Min95] M. Minsky. *Computational Haptics: The Sandpaper System for Synthesizing Texture for a Force-Feedback Display*. PhD thesis, Massachusetts Institute of Technology, 1995.

[Mir96] B. V. Mirtich. *Impulse-based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California, Berkeley, 1996.

[Mir98a] B. V. Mirtich. Rigid body contact: Collision detection to force computation. Technical Report TR98-01, Mitsubishi Electric Research Laboratory, 1998.

[Mir98b] B. V. Mirtich. V-Clip: Fast and robust polyhedral collision detection. In *ACM Trans. on Graphics*, volume 17(3), pages 177–208, 1998.

[Mir00] B. V. Mirtich. Timewarp rigid body simulation. In *Proc. of ACM SIGGRAPH*, pages 193–200, 2000.

[MOYS+90] M. Minsky, M. Ouh-Young, O. Steele, F. P. Brooks, Jr., and M. Behensky. Feeling and seeing: Issues in force display. In *Computer Graphics (Proc. of ACM Symposium on Interactive 3D Graphics)*, volume 24, pages 235–243, 1990.

[MPT99] W. McNeely, K. Puterbaugh, and J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *Proc. of ACM SIGGRAPH*, pages 401–408, 1999.

[MQW01] K. McDonnell, H. Qin, and R. Wlodarczyk. Virtual clay: A real-time sculpting system with haptic interface. In *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 179–190, 2001.

[MRF+96] W. Mark, S. Randolph, M. Finch, J. Van Verth, and R. M. Taylor II. Adding force feedback to graphics systems: Issues and solutions. In *Proc. of ACM SIGGRAPH*, pages 447–452, 1996.

[MS94] T. M. Massie and J. K. Salisbury. The phantom haptic interface: A device for probing virtual objects. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 1:295–301, 1994.

[MS01] V. J. Milenkovic and H. Schmidl. Optimization-based animation. In *Proc. of ACM SIGGRAPH*, pages 37–46, 2001.

[MW88] M. Moore and J. Wilhelms. Collision detection and response for computer animation. In *Computer Graphics (Proc. of ACM SIGGRAPH)*, volume 22, pages 289–298, 1988.

[NJC99] D. D. Nelson, D. E. Johnson, and E. Cohen. Haptic rendering of surface-to-surface sculpted model interaction. *Proc. of ASME Dynamic Systems and Control Division*, 1999.

[OC99] A. M. Okamura and M. R. Cutkosky. Haptic exploration of fine surface features. *Proc. of IEEE International Conference on Robotics and Automation*, pages 2930–2936, 1999.

[OC01] A. M. Okamura and M. R. Cutkosky. Feature detection for haptic exploration with robotic fingers. *International Journal of Robotics Research*, 20(12):925–938, 2001.

[OD01] C. O'Sullivan and J. Dingliana. Collisions and perception. In *ACM Trans. on Graphics*, volume 20(3), pages 151–168, 2001.

[ODGK03] C. O'Sullivan, J. Dingliana, T. Giang, and M. K. Kaiser. Evaluating the visual fidelity of physically based animations. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)*, volume 22, pages 527–536, 2003.

[OJSL04] M. A. Otaduy, N. Jain, A. Sud, and M. C. Lin. Haptic display of interaction between textured models. *Proc. of IEEE Visualization Conference*, pages 297–304, 2004.

[OL03a] M. A. Otaduy and M. C. Lin. CLODs: Dual hierarchies for multiresolution collision detection. In *Proc. of Eurographics Symposium on Geometry Processing*, pages 94–101, 2003.

[OL03b] M. A. Otaduy and M. C. Lin. Sensation preserving simplification for haptic rendering. In *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)*, volume 22, pages 543–553, 2003.

[OL04] M. A. Otaduy and M. C. Lin. A perceptually-inspired force model for haptic texture rendering. In *Proc. of Symposium on Applied Perception in Graphics and Visualization*, pages 123-126, 2004.

[ORC99] C. O'Sullivan, R. Radach, and S. Collins. A model of collision perception for real-time animation. *Computer Animation and Simulation*, pages 67–76, 1999.

[OY90] M. Ouh-Young. *Force Display in Molecular Docking*. PhD thesis, University of North Carolina at Chapel Hill, 1990.

[PC99] M. Peshkin and J. E. Colgate. Cobots. *Industrial Robot*, 26(5):335–341, 1999.

[PR97] D. K. Pai and L. M. Reissel. Haptic interaction with multiresolution image curves. *Computer and Graphics*, 21:405–411, 1997.

[PvdDJ+01] D. K. Pai, K. van den Doel, D. L. James, J. Lang, J. E. Lloyd, J. L. Richmond, and S. H. Yau. Scanning physical interaction behavior of 3d objects. In *Proc. of ACM SIGGRAPH*, 2001.

[Qui94] S. Quinlan. Efficient distance computation between non-convex objects. In *Proc. of International Conference on Robotics and Automation*, pages 3324–3329, 1994.

[RK00] D. Ruspini and O. Khatib. A framework for multi-contact multi-body dynamic simulation and haptic display. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.

[RKC02] S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies. In *Proc. of Eurographics*, 2002.

[RKK97] D. C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. In *Proc. of ACM SIGGRAPH*, pages 345–352, 1997.

[RPP+01] M. Renz, C. Preusche, M. Pötke, H.-P. Kriegel, and G. Hirzinger. Stable haptic interaction with virtual environments using an adapted voxmap-pointshell algorithm. *Eurohaptics Conference*, 2001.

[RV64] I. Rock and J. Victor. Vision and touch: An experimentally created conflict between the two senses. *Science*, 143:pp. 594–596, 1964.

[Sal99] J. K. Salisbury. Making graphics physically tangible. *Communications of the ACM*, 42(8), 1999.

[SBM+95] J. K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Zilles. Haptic rendering: Programming touch interaction with virtual objects. In *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 123–130, 1995.

[SDS96] E. Stollnitz, T. DeRose, and D. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan-Kaufmann, 1996.

[Sei90] R. Seidel. Linear programming and convex hulls made easy. In *Proc. 6th Ann. ACM Conf. on Computational Geometry*, pages 211–215, 1990.

[SGG+00] P. V. Sander, X. Gu, S. J. Gortler, H. Hoppe, and J. Snyder. Silhouette clipping. In *Proc. of ACM SIGGRAPH*, pages 327–334, 2000.

[Shi92] K. Shimoga. Finger force and touch feedback issues in dextrous manipulation. *NASA-CIRSSE International Conference on Inetelligent Robotic Systems for Space Exploration*, 1992.

[SP96] J. Siira and D. K. Pai. Haptic textures – a stochastic approach. *Proc. of IEEE International Conference on Robotics and Automation*, pages 557–562, 1996.

[SP97] S. E. Salcudean and N. R. Parker. 6-dof desk-top voice-coil joystick. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 131–138, 1997.

[SPD00] C. Spence, F. Pavani, and J. Driver. Crossmodal links between vision and touch in covert endogenous spatial attention. *Journal of Experimental Psychology: Human Perception and Performance*, volume 26, pages 1298–1319, 2000.

[SSGH01] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Proc. of ACM SIGGRAPH*, pages 409–416, 2001.

[ST96] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal of Numerical Methods in Engineering*, 39:2673–2691, 1996.

[ST00] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. *Proc. of IEEE International Conference on Robotics and Automation*, pages 162–169, 2000.

[SU93] M. Slater and M. Usoh. An experimental exploration of presence in virtual environments. Technical Report 689, Department of Computer Science, University College London, 1993.

[Sut65] I. Sutherland. The ultimate display. *Proc. of IFIP*, pages 506–508, 1965.

[SV94] S. E. Salcudean and T. D. Vlaar. On the emulation of stiff walls and static friction with a magnetically levitated input/output device. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 1994.

[Tau95] G. Taubin. A signal processing approach to fair surface design. In *Proc. of ACM SIGGRAPH*, pages 351–358, 1995.

[TJC97] T. V. Thompson, D. E. Johnson, and E. Cohen. Direct haptic rendering of sculptured models. In *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 167–176, 1997.

[TRC+93] R. M. Taylor II, W. Robinett, V. L. Chi, F. P. Brooks, Jr., and W. Wright. The nanomanipulator: A virtual-reality interface for a scanning tunneling microscope. In *Proc. of ACM SIGGRAPH*, pages 127–134, 1993.

[TSEC94] H. Z. Tan, M. A. Srinivasan, B. Eberman, and B. Cheng. Human factors for the design of force-reflecting haptic interfaces. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 1:353–360, 1994.

[UNB+02] B. J. Unger, A. Nicolaidis, P. J. Berkelman, A. Thompson, S. J. Lederman, R. L. Klatzky, and R. L. Hollis. Virtual peg-in-hole performance using a 6-dof magnetic levitation haptic device: Comparison with real forces and with visual guidance alone. In *Proc. of Haptics Symposium*, pages 263–270, 2002.

[van01] G. van den Bergen. Proximity queries and penetration depth computation on 3d game objects. *Game Developers Conference*, 2001.

[Whi94] D. J. Whitehouse. *Handbook of Surface Metrology*. Institute of Physics Publishing, Bristol, 1994.

[WM03] M. Wan and W. A. McNeely. Quasi-static approximation for 6 degrees-of-freedom haptic rendering. *Proc. of IEEE Visualization Conference*, pages 257–262, 2003.

[WS03] S. P. Walker and J. K. Salisbury. Large haptic topographic maps: Marsview and the proxy graph algorithm. In *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 83–92, 2003.

[Wu00] D. Wu. Penalty methods for contact resolution. *Game Developers Conference*, 2000.

[YSLM04] S. Yoon, B. Salomon, M. C. Lin, and D. Manocha. Fast collision detection between massive models using dynamic simplification. In *Proc. of Eurographics Symposium on Geometry Processing*, 2004.

[ZS95] C. Zilles and J. K. Salisbury. A constraint-based god object method for haptics display. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995.

[ZSS97] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proc. of ACM SIGGRAPH*, pages 259–268, 1997.